第四届中国软件开源创新大赛-开源项目创新赛自由组

恶意流量智能识别分类与预测系统

作品类别：系统软件

网络安全创领者

不忘初心，牢记使命

为国家网络安全事业持续赋能

目录

1. 目标问题与意义价值

1.1 背景与问题

1.1.1 国内外网络安全研究现状

随着人工智能、大数据、5*G* 等新兴技术的发展，企业面临的威胁日益严重化。相关数据显示，在 2015 年至 2025 这十年间，网络攻击引发的全球潜在经济损失可能高达 2940 亿美元。网络风险的升级，让政府、企业和个人都对该风险愈加关注。

我国自 2017 年 6 月开始实行《网络安全法》。2019 年 5 月，我国发布了等级保护 2.0 国家标准，增加了个人信息保护、云计算扩展等要求。国家互联网应急中心发布的《2019 年上半年我国互联网网络安全态势》显示，2019 年上半年，我国互联网网络安全状况具有四大特点：个人信息和重要数据泄露风险严峻；多个高危漏洞曝出给我国网络安全造成严重安全隐患；针对我国重要网站的 *DDoS* 攻击事件高发；利用钓鱼邮件发起有针对性的攻击频发。国家互联网应急中心从恶意程序、漏洞隐患、移动互联网安全、网站安全以及云平台安全、工业平台安全、互联网金融安全等方面，对我国互联网网络安全环境开展宏观监测。数据显示，与 2018 年上半年数据比较，2019 年上半年我国境内通用型“零日”漏洞收录数量，涉及关键信息基础设施的事件型漏洞通报数量，遭篡改、植入后门、仿冒网站数量等有所上升，其他各类监测数据有所降低或基本持平。

2018 年，世界经济论坛发布的《2018 年全球风险报告》中首次将网络攻击纳入全球风险前五名，成为 2018 年全球第三大风险因素。2019 年，随着当前生产和生活对网络信息平台依赖性的增强，网络攻击事件的数量仍将不断增多，影响范围也将更加广泛。全球网络对抗态势进一步升级。网络空间已成为各国争夺的重要战略空间，各国采取多种措施不断谋求增强网络防御和对抗能力。1)美国 2019 年 7 月发布的《2019 年国防授权法案》和 9 月发布的《国防部网络安全战略》在顶层规划中强化网络对抗战略意图。2)2019 年 8 月，日本防卫部门宣布将组建专门部队保护国防通信网络免受攻击以及 10 月北约提出将成立网络指挥部，以全面及时掌握网络空间状况。3)北约举行“锁定盾牌”网络战演习推动国际合作。4）韩国国防部不断深化网络武器研发。

随着相关国家网络空间政策的调整以及网络军事力量建设加速，网络空间争夺或将掀起新高潮。各国将更加重视数据安全，治理数据已成为国家重要战略资源和生产要素，针对数据的网络攻击以及数据滥用问题日趋严重，提升数据安全治理水平刻不容缓。2018 年 5 月，欧盟《通用数据保护条例》（*GDPR*）正式生效，欧盟国家参照 *GDPR* 研究制定国内数据保护相关规定，非欧盟国家，如阿根廷、巴西、伊朗、印度、泰国等国也调整其数据保护法规与 *GDPR* 保持一致。同年 10 月，欧盟议会通过《欧盟非个人数据自由流动条例》，消除欧盟成员国数据本地化的限制。因此，对恶意流量进行检测已经成为研究热点。

1.1.2 恶意流量检测的背景与问题

随着以5*G*技术为代表的网络技术的快速发展,互联网已经在工业、金融、教育等各个领域都广泛应用。然而,在互联网给我们的生活带来各种便利的同时,也带来了各种安全问题,各种特洛伊木马程序、蠕虫、病毒等恶意软件的数量和种类也在快速增多,为互联网用户的安全带来了更多的问题和隐患。

网络空间主权是国家主权在位于其领土之中的信息通信基础设施所承载的网络空间中的自然延伸，即对出现在该空间的信息通信技术活动及信息通信技术平台本身及其数据具有主权。在《2020 中国网络安全企业 100 强报告》中提到，未来十年，中国网络安全市场规模将增长十倍、网络安全会成为优先级最高的 *IT* 投资、网络安全将成为第一生产力。在《2020 年中国网络安全产业分析报告》中的提到了几组数据：2019 年我国网络安全市场规模约为 478 亿元，同比增长率为 21.5%，如今已进入稳健增长阶段。尽管受到新冠肺炎疫情和 *GDP* 增速放缓影响，但由于新基建（5*G*、物联网等）催生引领新需求并将形成可观的增量市场，预计未来三年将保持 15%以上增速，到 2022 年市场规模预计将达到 759 亿元。由此可见，网络安全已是未来的趋势所向。因此，利用一些技术手段来维护网络的安全，维护国家网络空间主权尤为必要。

在互联网发展之初,主要是为了增强通信能力,并没有过多的考虑安全问题。然而，在万物互联的时代，早期的很多网络协议因为缺乏安全防护，从而极易被黑客利用然后产生网络攻击行为。因此，为了保护普通用户在网络活动中的隐私信息,很多学者和公司已经逐步展开了加密传输的研究。*Netscape* 公司在 1994 年提出了套接字安全协议（*SecureSocketLayer*，*SSL*)协议,并紧接着提出 *SSL*2 协议，然而 *SSL*2 因讨论不足而存在严重的漏洞[1]。为了解决 *SSL2* 协议中存在的重放攻击,消息认证等问题,1995 年发布了 *SSL*3,并在协议中增添了 *ChangeCipherSpec* 等新内容。然后,*TransportLayerSecurity(TLS)* 工作组基于 *SSL*3 设计了 *TLS*,并发布了 *TLS*1.0、*TLS*1.1 和 *TLS*1.2,逐步修补了协议在设计和实现中先后发现的漏洞,并在 2016 年着手计划出台具有更好完整性的 *TLS*1.3[2]。*TLS* 的新版本 *TLSv*1.3 已于 2018 年 8 月发布[6]。与 *TLSv*1.2 相比，它引入了重大变化和性能改进。浏览器和网络服务器正在慢慢采用这一新版本:*Chrome*、*Firefox* 或 *Opera* 等主要浏览器已经支持它，如果网络服务器能够过渡，约 66%的用户将能够使用该版本。

现如今，大多数互联网流量都是用 *TLS* 协议进行加密的,*TLS* 协议已经作为一种工业标准大量应用于电子商务等安全应用。它的主要用途是加密超文本传输协议流量，并以此形成 *HTTPS* 协议,旨在为 *HTTP* 客户端和 *Web* 服务器之间创建一条安全的连接,来实现端到端的认证并保证数据传输的保密性和完整性。*Cisco* 的调查[3]显示,到 2019 年,80%的全球网络流量将被加密，企业、学校、政府和个人都受益于隐私加密，*TLS* 的使用在未来几年肯定会继续增长。然而，加密传输并不能保证安全，恶意软件开发者已经开始利用 *TLS* 来隐藏他们的恶意活动，从而逃避检测。*Cisco* 的报告显示，以某种方式使用 *TLS* 的恶意软件样本的百分比从 2015 年 8 月的 2.21%上升到 2017 年 5 月的 21.44%。在考虑到如今 *TLS* 已经变得无处不在的情况下，我们可以预估这一数字还会增长。因此,如何检测恶意 *TLS* 加密流量已成为恶意软件检测识别中的一个重要研究热点。

1.1.3 LightGBM 算法的背景

提升树是利用加法模型和前向分布算法实现学习的优化过程，它有一些高效的实现，如 *GBDT, XGBoost* 和 *pGBRT*，其中 *GBDT* 是通过损失函数的负梯度拟合残差，*XGBoost* 则是利用损失函数的二阶导展开式拟合残差。但是，当面对大量数据集和高维特征时，其扩展性和效率很难令人满意，最主要的原因是传统的 *boosting* 算法需要对每一个特征都扫描所有的样本数据来获得最优切分点，这个过程是非常耗时的。为了解决这种在大样本高纬度数据的环境下耗时的问题，我们采用了基于 *GBDT* 的另一种形式 *LightGBM*----基于直方图的切分点算法。

Boosting 是用一系列子模型的线性组合来完成学习任务的，它分为两种类型 *AdaBoosting* 和 *GradientBoosting*，*LightGBM* 属于 *GradientBoosti* 的一种。*GradientBoosting* 的思想是：一次性迭代变量，迭代过程中，逐一增加子模型，并且保证损失函数不断减小。

*GBDT(GradientBoostingDecisionTree)*是机器学习中一个长盛不衰的模型。*GBDT* 拥有着 *GradientBoosting* 和 *DecisionTree* 的功能共同特性，具有训练效果好、不易过拟合等优点。*GBDT* 的工具主要包括 *XGBoosting*、*Pgbrt*、*Sklearn*、*R.GBM* 等。*GBDT* 在工业界应用广泛，通常被用于点击率预测，搜索排序等任务。*GBDT* 也是各种数据挖掘竞赛的致命武器，据统计 *Kaggle* 上的比赛有一半以上的冠军方案都是基于 *GBDT*。

LightGBM 是 *GBDT* 的一种，被提出的主要原因是为了解决 *GBDT* 在海量数据遇到的问题，让 *GBDT* 可以更好更快地用于实践。*LightGBM* 中的决策树子模型是采用按叶子分裂的方法分裂节点的，因此它的计算代价比较小，也正是因为选择了这种分裂方式，需要控制树的深度和每个叶子节点的最小数据量，从而避免过拟合现象的发生。*LightGBM* 选择了基于 *Histogram* 的决策树算法，将特征值分为很多个小“桶”，进而在这些“桶”上寻找分裂，这样可以减小储存成本和计算成本。另外，类别特征的处理，也使得 *LightGBM* 在特定数据下有比较好的提升。*LightGBM* 分为三类：特征并行、数据并行和投票并行。特征并行运用在特征较多的场景，数据并行应用在数据量较大的场景，投票并行应用在特征和投票都比较多的场景。

LightGBM 算法自 2016 年发布以来，已经广泛运用于大数据机器学习领域，与之前的 *XGBoosting* 并称为当今机器学习的“倚天屠龙”。公开数据的实验表明 *LightGBM* 能在学习效率和准确率上都表现出比其他已有 *Boosting* 工具更好的表现，相比 *XGBoosting* 速度更快，内存占用更少，准确率更高。此外，实验也表明 *LightGBM* 通过使用多台机器进行特定设定的训练，它能取得线性加速效果。因此，总结该算法的优点显著体现在如下五个方面：①更快的训练速度；②更低的内存消耗；③更好的模型精度；④支持并行学习；⑤可以快速处理海量数据。将性能优良的 *LightGBM* 算法运用于恶意流量的识别，其可靠性和灵活性将大大促进相关领域的长足发展。

目前，知网上基于 *TSL* 协议进行恶意流量检测的文献很少，虽然已经有了基于深度学习进行恶意流量分类[4-5]，但是监督学习更适合我们的初始目标，因为监督学习通过获取标签集对恶意流量进行识别、分类，可以更好地理解所获得的结果，从而帮助分析师了解恶意软件流量与正常流量的确切区别。虽然监督机器学习方法已经广泛应用于异常

检测，但是，关于 *TLS* 异常检测的文献并不多，此外，对比多种算法模型发现 *LightGBM* 算法在该方面准确度最高。因此，基于 *LightGBM* 算法的 *TLS* 恶意流量识别、分类与预测系统具有广大的研究前景。更重要的是目前市面上还没有较为成型的集恶意流量识别分类和预测功能一体化的系统。

1.2 应用领域与应用价值

全球互联网走向全面加密时代已经是大势所趋。但在加密访问可保障通讯安全的情况下，绝大多数网络设备对网络攻击、恶意软件等加密流量却无能为力。攻击者利用 *SSL* 加密通道完成恶意软件载荷和漏洞利用的投递和分发，以及受感染主机与命令和控制(*C&C*)服务器之间的通信，这就造成了对此类恶意网络流量无法侦测，导致遗漏重要信息。

Gartner 预测：2019 年，超过 80%的企业网络流量将被加密，届时加密的流量中将隐藏超过一半以上的网络恶意软件。识别恶意流量的传统方法是解密流量，并使用诸如新一代防火墙等设备查看流量。但这种方法耗时较长，破坏了加密技术解决数据隐私的初衷，且需要在网络中添加额外的设备。同时不能对无法获取秘钥的加密流量进行解密及检测。针对上述情况，怎样在加密流量中检查到恶意通讯数据就成为摆在人们面前地难题。面对这一严峻形势，世界一流的安全厂商如思科、*Redware*，以及一些安全初创公司，纷纷开始研究针对加密流量的安全检测与防护手段，并先后发布了相应产品。虽然国内安全初创公司观成科技宣布正式发布有效针对恶意加密流量的 *AI* 检测引擎，但在国内，这一新型检测技术尚属稀缺。

因此，本文开发基于 *LightGBM* 算法的 *TLS* 恶意流量识别、分类与预测系统在国内会有很大的市场。本文提出的系统基于人工智能用于恶意流量安全检测将是一种新技术手段，即以人工智能技术赋能恶意流量检测，通过本次实验成功验证了基于 *LightGBM* 算法的 *TLS* 恶意流量模型在对恶意 *TLS* 网络流分类的结果准确率、召回率和 *f*1 分数均达到 99%以上，充分展现了本文提出的算法模型具有高度的可行性和良好的应用前景。

1.3 目标与基本功能

本文主要的目标是：

(1) 对加密流量实现恶意检测,能识别不同网络场景中新的恶意加密流量，并对恶意流量进行分类,给出分类的结果和时间,从而有利于相应公司采取相应的措施；

(2) 利用现有的日志流量数据预测未来遇到的恶意流量类型以及发生的时间，并提出建议措施。

(3) 开发恶意流量识别、分类与预测系统，实现流量捕获、流量预处理、特征提取、分类器训练、恶意流量检测等功能。

本文的剩余部分组织如下，

第 2 部分总体概述了作品解决问题的主要设计思路与技术路线，并简要概述了 *TLS*

协议分析、恶意软件如何利用 *TLS*、针对此类恶意软件的检测技术的当前状态、恶意流量和良性流量特征差异分析（流量元数据差异、*TLS* 参数差异、证书差异）。

第 3 部分详细介绍了作品的基本思想和实现，包括流量捕获、流量预处理、相关特征的选择、分类模型的介绍和模型验证方法。

第 4 部分说明作品实际运行的情况与运行效果。首先，分析了数据集的来源、模型评估方法、模型在数据集上的性能测试结果,给出了我们所使用的数据集以及在这些数据集上进行的实验；最后，简单介绍了开发系统的应用，即 *HTML*、*css*、*js*、*jquery*、*python*、*React* 框架、*express* 框架的应用。本文使用以上语言及框架开发了基于 *LightGBM* 算法的恶意流量识别、分类与预测系统。系统的集成功能有：流量捕获、流量预处理、特征提取、分类器训练、恶意流量分类与预测等。

第 5 部分说明了作品在创意、技术、应用等方面的创新与特色点。

第 6 部分对本文所做的工作进行了总结、展望，以及介绍了团队成员。

附件为恶意流量 15 个类别对应的名称以及识别次数以及界面具体图例介绍

1.4 具体计划安排

为了保证项目能够如期完成，并达到期望的效果，因此，我们制定了项目具体计划安排。虽然在执行过程中会因难点问题或者相关软件的使用而影响项目进度，但是与计划表前后相差不大.最终，我们如期完成了项目。具体如图 1 所示，

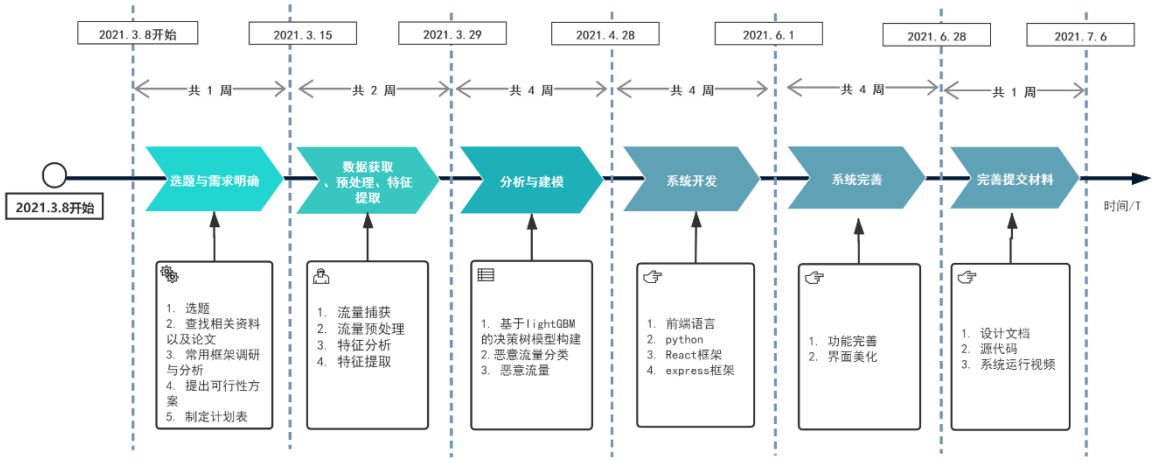

图 1 项目具体计划安排

1.5 开发工具与技术

工具：*Anaconda*、*Jupyter*、*Pycharm*、*Wireshark*、*python*、*React* 框架、*express* 框架

技术：相关性分析、*LightGBM*

2. 设计思路与方案

2.1 恶意流量监测思路与方案

我们的目的是对恶意流量进行正确识别。基本思想是通过监控和捕获网络加密流量，对原始流量进行预处理；然后提取流量中的比较明显的流量元数据特征、*TLS* 特征、证书特征等，构建特征因子集合；最后，训练机器学习分类模型,并利用该训练模型对其他加密流量进行识别与预测。整体检测流程如图 2 所示。

在图 2 中，我们可以看到整个测试流程包括流量捕获、流量预处理、特征提取以及模型训练等四个部分。

(1) 流量捕获部分主要是采用工具 *wireshark* 抓包工具来完成采集,为了得到训练阶段加密的恶意流量及正常流量,流量捕获阶段将在沙箱中运行恶意软件和正常软件生成流量数据;本文后续的工作中,*wireshark* 会提供了一个接口进行实时抓包，从而实现实时分析数据并识别恶意流量的目的。

(2) 流量预处理阶段主要完成对流量的清洗,过滤未加密流量和不完整的会话,生成可用于流量检测的会话信息;

(3) 特征提取阶段主要是根据需求,提取相关会话的统计特征和平台特征信息,形成训练的特征因子集合;

(4) 模型训练阶段主要是构建恶意加密流量分析的模型。

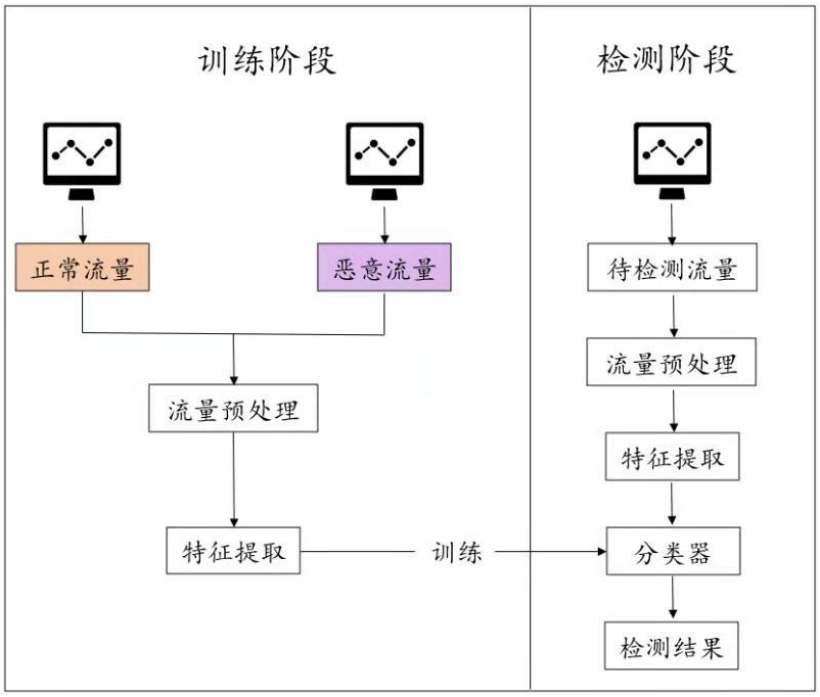

图 2 恶意流量检测流程图

接下来主要概述了流量来源、*TLS* 协议分析、恶意软件如何利用 *TLS*、针对此类恶

意软件的检测技术的当前状态、恶意流量和良性流量特征差异分析（流量元数据差异、*TLS* 参数差异、证书差异）

2.2 TLS 协议分析

TLS 协议是一种加密协议，其主要目标是“在两个通信应用程序之间提供隐私和数据完整性”。*TLS* 的第一个版本于 1999 年 1 月发布，以取代现在已被否决的 *SSL* 协议。截至 2019 年 5 月，*TLS* 最广泛的版本是 2008 年 8 月发布的 *TLSv*1.2。

TLS 协议位于应用层之下，传输层之上。如今，*TLS* 的主要用途是加密超文本传输协议流量，并以此形成 *HTTPS* 协议。据谷歌[6]称，2019 年 4 月，在美国，90%的访问页面都是通过 *HTTPS* 加载的。然而 *TLS* 的使用并不仅限于 *HTTP*，任何应用层协议理论上都可以利用 *TLS*。例如，*SMTP* 协议和 *TLS* 一起构成了保护电子邮件的 *SMTPS*。*TLS* 的新版本 *TLSv*1.3 已于 2018 年 8 月发布[7]。与 *TLSv*1.2 相比，它引入了重大变化和性能改进。浏览器和网络服务器正在慢慢采用这一新版本:*Chrome*、*Firefox* 或 *Opera* 等主要浏览器已经支持它，如果网络服务器能够过渡，约 66%的用户将能够使用该版本。在接下来的章节中，我们将简要介绍 *TLSv*1.2 协议、*TLSv*1.3 引入的主要变化以及 *TLS* 证书的介绍。

2.2.1 TLSv1.2

*TLSv*1.2 是目前最常用的 *TLS* 版本:95%的网络服务器支持它[3]。它的主要目标是为不安全的 *MD*5 和 *SHA*1 算法提供新的替代方案。

TLS 会话是在客户端和服务器之间的两次消息往返中建立的，如图 3 所示。

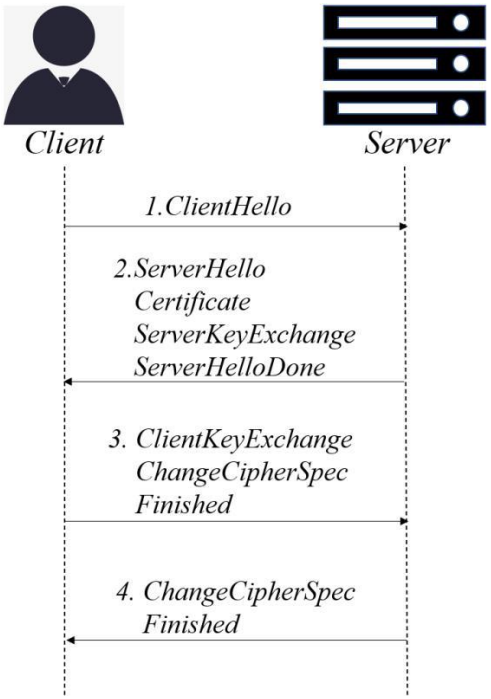

图 3 *TLSv*1.2 完整握手过程

1. 客户端发送一个 *ClientHello* 消息。此消息包含:

- 客户端支持的密码套件列表;
- 压缩方法列表;
- 客户端用来向服务器传达附加信息的扩展列表(如指定目标主机名、客户端支持的椭圆曲线等)。

2. 服务器以几条消息响应:

(a) *SeverHello*:包含服务器选择的密码套件和压缩方法。

(b) *Certificate*:包含一系列证明公钥所有权的顶级域名证书。

(c) *ServerKeyExchange*:包含允许客户端传递预设密码的信息。

(d) *ServerHelloDone*:通知客户端，在握手的第一部分，服务器已经发送完消息。3.客户端回复:

(a)*ClientKeyExchange*:包含允许服务器计算最终对称会话密钥的信息。

(b)*ChangeCipherSpec:*通知服务器所有后续消息都将使用会话密钥加密。

(c)*Finished:*通知服务器客户端 *TLS* 握手成功。

4.最后，服务器结束协商:

(a)*ChangeCipherSpec:*通知客户端所有后续消息都将使用会话密钥加密。

(b)*Finished*:通知客户端 *TLS* 握手对于服务器是成功的。

直到 *ChangeCipherSpec* 之前的所有消息都以明文形式发送，只有当双方共享对称密钥之后，才能实现加密。

2.2.2 TLSv1.3

*TLSv*1.3 是 *TLS* 的新版本，于 2018 年 8 月发布，主要的目的在于提高安全性和速度。主要变化是:

- 删除过时的密码和散列算法:*SHA1，MD5，DES...*
- 握手所需的消息数量已经减少。
- *SeverHello* 之后的所有消息现在都已加密。

握手只需要 3 组消息，如图 4 所示:

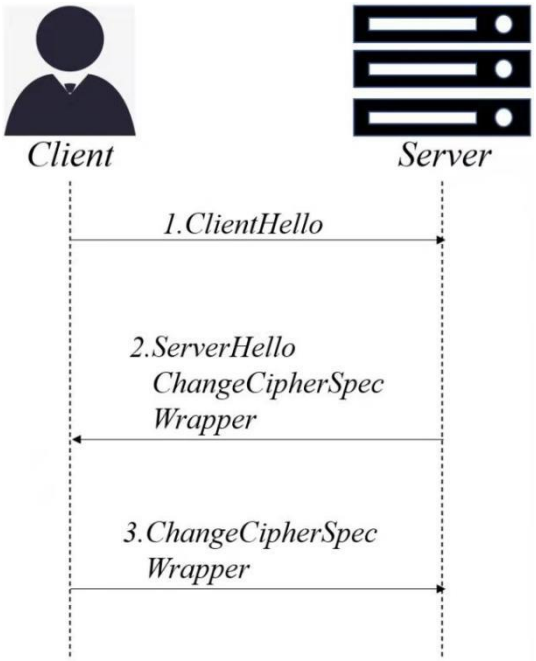

图 4 *TLSv*1.3 会话复用过程

1. *ClientHello* 消息与 *TLSv*1.2 中的消息相同，但也带有一个公钥列表。
2. 服务器会回复几条消息:

(a) *SeverHello* 包含与 *TLSv*1.2 中相同的信息以及公钥。从客户端的公钥和服务器的公钥中，导出一个新的共享密钥，该密钥仅用于加密握手的其余部分。

(b) *ChangeCipherSpec* 与 *TLSv*1.2 中的目的相同

(c) *Wrape*r 包含 *TLSv*1.2 中也存在的服务器消息:已完成、证书等。

3. 客户端以下列方式结束握手:

(a) *ChangeCipherSpec*，其目的与 *TLSv*1.2 中的相同。

(b) *Wraper*，其包含 *TLSv*1.2 中也存在的客户端消息:完成的可选消息...

需要注意的是，在 *ChangeCipherSpec* 之后，包括服务器证书在内的所有数据包都是加密的。这可以防止窃听者仅仅通过查看证书主题来识别与服务器相关联的主机名。此外，*TLSv*1.3 还添加了一个新的扩展:加密服务器名称指示(*EncryptedServerNameIndication*，*ESNI)*，它是 *TLSv*1.2 中 *SNI* 扩展的加密版本，主要的功能是客户端使用它来指示它们试图连接到哪个特定的主机名。

2.2.3 TLS 服务器证书

服务器证书是将公钥绑定到服务器的电子文档，由称为证书颁发机构的实体进行数字签名。在 *TLS* 握手期间接收服务器证书的客户端必须验证该证书是否有效，以及是否已由可信机构签名。如果证书是由一个未知的组织签署的，客户端必须在证书链中向上

移动，直到找到一个来自可信机构的证书，该机构通过构造来验证证书链中它下面的所有证书。当服务器的合法性最终得到验证时，服务器的公钥可以用来加密和共享对称会话密钥。

TLS 证书一般遵循 *X*.509 标准[5]。它们包含几个字段，其中一些是可选的。最著名的有:

- 颁发者(*Issuer*):验证服务器合法性并颁发证书的实体。
- 有效性(*Validity*):包含两个子字段，日期从证书有效日期到证书过期日期。
- 主题(*Subject*):证书的受益人。
- 主题公钥信息(*SubjectPublicKeyInfo*):包含两个子字段，指示服务器公钥算法和公钥本身。
- 扩展(*Extensions*):包含几个说明如何使用证书的字段以及关于证书的附加信息。
- 证书签名算法(*CertificateSignatureAlgorithm*)和证书签名值(*CertificateSignatureValue*):签名算法和证书主体颁发者的签名。
- 指纹(*Fingerprint*):不是证书的实际部分,而是整个证书的散列

然而，证书并不都有相同的验证级别级别。通常验证级别可以分为四类:

(1) 自签名(*Self-Signed,SS*):这是最低级别的验证，因为这意味着证书已由证书接收方自己备份。通常，这种证书根本不能提供密钥所有权的证据，浏览器会显示警告。

(2) 领域验证(*DomainValidation,DV*):认证中心只检查了公钥和域名之间的连接，没有进一步的身份检查。

(3) 组织验证(*OrganizationValidation,OV*):证书颁发机构在颁发证书之前已经检查了域名和组织的一些信息。

(4) 扩展验证(*ExtendedValidation,EV)*:“最强”证书，其中证书颁发机构除了验证域名之外，还对组织进行了彻底的调查.

2.3 恶意软件对 TLS 的使用

利用 *TLS* 的方法有很多，我们在这里介绍一些恶意软件使用的涉及 *TLS* 加密的技术。

- 有效载荷存放(*Payloaddeposit*):*TLS* 可能会被用于隐藏机器感染。例如，员工可能在使用 *TLS* 访问恶意网页时，免驱动下载的恶意软件就会在用户不知情的情况下自行安装。恶意软件会因为 *TLS* 的加密而逃避员工公司设置的基本有效负载检查。
- 数据泄露(*Dataexfiltration*):加密可用于防止敏感数据的泄露:密码、*cookies*、公司

数据等。然而，由 *TLS* 封装到攻击者服务器端口 443(通常的目的端口)的简单开机自检请求不会被内部防火墙阻止，从而可能会泄漏宝贵的信息。

- 命令和控制(*CommandandControl)*:恶意软件的开发者越来越多地利用 TLS 来混淆受感染机器获取 *C&C* 命令的过程，原因在于加密允许绕过对有效负载的检查，包括端口的 *TLS* 流量。

- 网络钓鱼*(Phishing)*:对于钓鱼开发来说,欺骗用户和窃取敏感信息的所有手段都值得钓鱼开发者探索。根据钓鱼网站 2018 年 11 月发布的一项调查[6]，80%的用户认为绿色挂锁意味着网站是安全的或可信任的，然而 49%的钓鱼网站似乎在使用 *HTTPS* 从而骗取用户的信任。

接下来，我们将介绍两个众所周知的使用 *TLS* 的恶意软件。

(1) *TorrentLocker*

TorrentLocker 使用 *AES* 加密受害者的磁盘,然后它提示目标用户支付大约 500 美元来解锁该磁盘。通常利用闪存和互联网浏览器漏洞的本地化垃圾邮件活动、依靠 *TSL* 的开机自检请求和免驱动下载来感染其他用户。

(2) *Dridex*

Dridex 是 2014 年首次出现的金融特洛伊木马，目的是窃取个人的银行凭证。通常通过垃圾邮件传播，发送包含恶意宏的 *Word* 文档等来感染其他用户。

2.4 基于 TLS 的异常检测的发展

越来越多的组织和公司意识到这些恶意软件带来的威胁。*Ponemon* 研究所发布的报告中指出[7], 68%的公司都担心加密的恶意软件通信会绕过网络保护从而使公司成为加密攻击的目标。这里我们概述了检测此类恶意软件现有的解决方案,以及它们的优缺点。

1. 传统检测平台

解密平台是企业环境中处理 *TLS* 通信的最常见方式，主要功能是拦截加密流量、动态解密流量、检查数据包的内容。然而，它有以下的缺点:

- 网络性能较差，相比于其他技术，平均性能下降 92%，延迟增加 672%。这是传统检测平台不断截获和解密数据包造成的结果。此外，这也是很多公司放弃该平台的主要原因之一[7]。

- 不符合当今某些国家的隐私标准，严重的甚至可能侵犯员工的权利。员工的银行凭证和电子邮件等敏感信息必须避免以明文形式暴露其内容，从这一层面来说增加了传统检测平台的复杂性。

- 平台必须跟踪解密数据包所需的所有 *TLS* 证书和密钥，这再次增加了此类平台的整体复杂性。

即使公司可以基于这种技术从有效载荷的现有检测方法中获得收益，但是改善性能

的高成本使得这种解决方案不太实用。

2. 证书分析

由于证书在握手过程中以明文形式发送，因此对网络服务器发送的证书进行分析有助于检测钓鱼网站或异常连接。当检测到自签名服务器证书或客户端证书字段中的异常字符串时，会发出数据泄露的警报。

基于证书分析最简单的方法是维护一个已知的恶意服务器的证书黑名单，并根据该列表检查所有传入的证书。然而，该方法具有所有基于黑名单的方法共有的缺点，即管理负担。简单来说，就是黑名单必须由可信的人定期更新，若更新不及时会发生合法网站被阻止的错误。

3. 基于 JA3 的 TLS 指纹识别

*JA*3 是一个从 *Salesforce* 到指纹 *TLS* 会话的项目，它可从 *ClientHello* 消息中提取五个值:*SSL* 版本、密码套件列表、*TLS* 扩展的长度、椭圆曲线组和椭圆曲线点格式，然后将值连接起来，用 *MD*5 对结果字符串进行哈希运算，以获得最终的 *JA*3 指纹。

恶意软件在通过 *TLS* 与他们的 *C&C* 服务器通信时经常使用自定义参数，从而产生了唯一的 *JA*3 指纹。*JA*3 的另一个好处是，由于它只从传输层提取信息，所以它不受针对互联网层的传统规避方法的影响。然而，客户端应用程序在使用相同的公共库或操作平台套接字时会导致了共享的 *JA3* 变散列，并不能阻止攻击者修改 *ClientHello* 的值。尽管如此，*JA*3 仍然是一个有用的元数据，尤其是当与其他方法结合时，会对恶意软件检测有很大帮助。

2.5 流量特征分析

大部分的恶意软件并不都是从零开始编写,而是对已有的恶意软件进行代码复用和修改，从而生成的新型恶意软件。恶意家族是指功能、行为类似的恶意软件，其会调用相同或相似的函数,执行类似的平台行为或者网络行为。通常由恶意软件和良性软件的特征的明显差异来进行特征选择。这部分主要讨论了恶意软件和良性软件之间的区别特征。并选择其中比较稳定的、不易受时间和场景影响的特征作为后续使用的分类特征，从而实现较为准确的对恶意软件家族进行归类。图 3-5 皆来自于 *Cisco* 报告[3]

2.5.1 流量元数据差异

收集的恶意通信平均比良性通信持续时间长，主要的原因在于 *Lastline* 的员工资源在使用 *TLS* 来加载网页和下载资源时会导致短暂的会话，而恶意软件可能会连续发送和接收大量数据。如图 5 所示

收集的恶意通信相比于良性通信字节熵差异较大，通常来说，高熵是使用强加密的结果，在良性通信中，会有多次短暂的会话，且 *TLS* 握手没有加密的比例在良性 *TLS* 会话的占比较大，所以良性会话的平均熵较低。如图 6 所示

收集的恶意通信相比于良性通信目的端口较多。因为通常情况下，良性通信的客户

端并不会向其他端口发起 *TLS* 会话；如图 7 所示

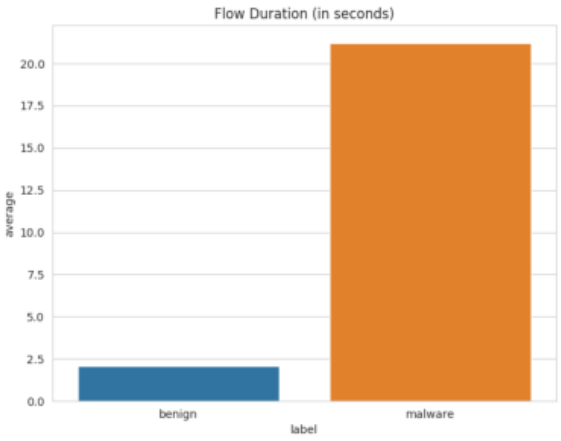

图 5 流量历时

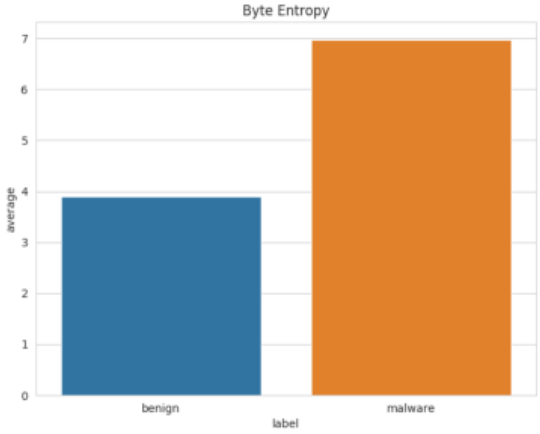

图 6 字节熵

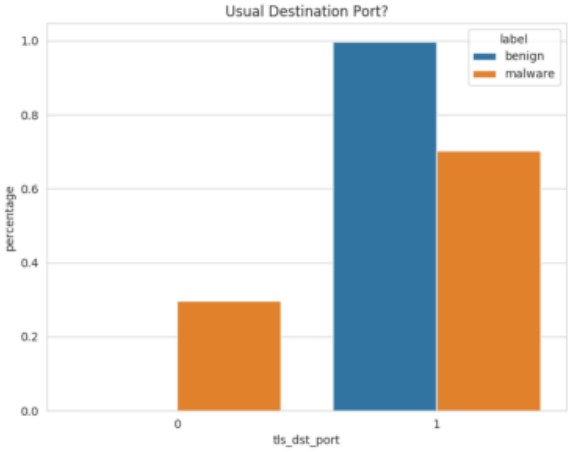

图 7 目的端口

2.5.2 TLS 参数差异

通常可从客户提供的密码套件和扩展列表进行鉴别。经常被恶意软件应用的五个密码套件的如下：

- *TLS_RSA_WITH_RC4_128_MD5*(4or0x0004)
- *TLS_RSA_WITH_RC4_128_SHA*(5or0x0005)
- *TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA*(19or0x0013)
- *TLS_DHE_DSS_WITH_AES_128_CBC_SHA*(50or0x0032)
- *TLS_DHE_DSS_WITH_AES_256_CBC_SHA*(56or0x0038)

通常来说，恶意软件开发者缺乏对强有力选择的认知，缺乏在恶意软件发布后用现代密码套件远程更新它们的机制，缺乏对密码套件的强度的持续关注。对于扩展列表来说，

恶意软件倾向于使用更少的扩展，恶意软件平均提供 3 个扩展，而正常流中有 10 个扩展。

2.5.3 证书差异

证书包含许多参数，这些参数对于两种类型的流量来说是完全不同的。恶意软件更倾向于依赖自签名证书，更有可能使用比合法更长的有效期，因为知名来源的合法证书价格昂贵。良性流量更倾向于较短的期限，因为平台管理员会自动更新程序，有效的限制了证书泄露造成的损失。

3.方案实现

我们的目的是对恶意流量进行正确识别。基本思想是通过监控和捕获网络加密流量，对原始流量进行预处理；然后提取流量中的比较明显的流量元数据特征、*TLS* 特征、证书特征等，构建特征因子集合；最后，训练机器学习分类模型,并利用该训练模型对其他加密流量进行识别与预测。本章主要分为流量捕获、流量预处理、特征降维、模型训练、模型验证几个部分。

3.1 流量捕获

为了获得训练用的纯净加密流量和检测阶段的流量数据,构建了如图 8 所示的流量捕获模型。在图 8 中,正常流量的获取通过在监控计算机上运行 *wireshark* 等工具捕获访问正常加密网站或运行正常软件产生的流量来获得,或者通过监控较为干净的网络环境流量来获得,并通过白名单过滤获得白名单中的会话作为正常流量。恶意流量的获取采用沙箱方式,在沙箱中运行恶意软件,保存其运行期间产生的流量,然后过滤掉沙箱间通信流量及平台白流量,将剩余的流量作为恶意流量。

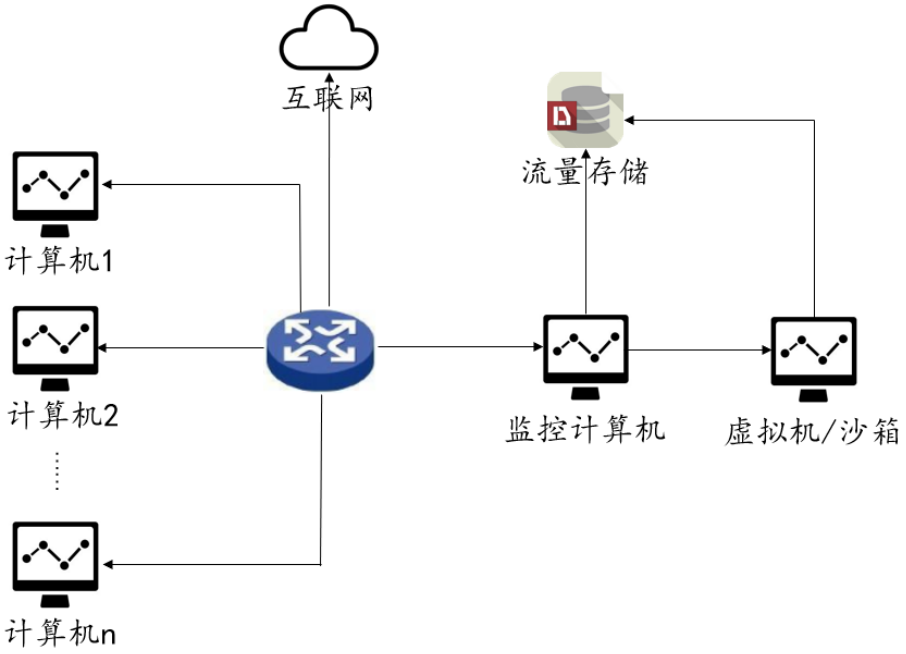

图 8 流量采集模型

3.2 流量预处理

一般来说，原始数据会呈现出一定的问题，如缺失、重复、冗余、歧义等，这些都是需要进行清洗或者整理，以达到算法需要的质量要求，使得数据更符合构建模型的要求。许多数据的异常是由生产过程不平稳引起的，而不是本身的错误。

异常值是指一组测定值中与平均值的偏差超过两倍标准差的测定值，一般异常值采用直接删除的方法处理即可。特殊情况下保留部分异常值可以更加精准地分析异常运行

过程，及早地为操控人员提供决策支持。

缺失值是指是指粗糙数据中由于缺少信息而造成的数据的聚类、分组或截断。在建模的过程中会丢失大量信息，模型所表现出的不稳定性会更加显著，所蕴含的规律更难把握，从而导致不可靠的输出。缺失值的处理一般有不处理、删除有缺失值的记录和插值处理。直接删除有确实值得记录比较简单，但是会丢失隐藏在已被删除记录中的信息。但是数据量较大的案例中，直接删除极小部分的缺失值影响不大。一般缺失数据主要分为两类，一类为全为 0 的数据，另一类为单个数据缺失。对于第一种情况，可直接进行删除处理；对于单个数据缺失的情况，可进行插值处理。

对于本项目来说，抓包获得的流量数据实际上是个有很多问题的数据集，我们通过 *python* 对抓包的流量数据表“*malware_lastline.csv*”进行预处理从而变成了规整的数据，保存在表“*class_num_antiy.csv*”中。对加密流量进行预处理的步骤如下:

(1) 过滤未加密的流量,保留使用 *TLS* 协议的流量;

(2) 过滤会话,从混杂的包中提取会话,过滤未完成完整握手过程和未传输加密数据的会话。通过观察会话中是否有 *ClientHello* 消息和 *ChangeCipherSpec* 消息,来判断握手是否完成;通过观察会话中是否有 *ApplicationData* 消息,来判断会话是否传输了加密数据。

(3) 过滤重传包、确认包及传输丢失的坏包,以避免对分类造成影响

3.3 特征降维

提取每个加密会话有关的流量元特征、*TLS* 握手特征、证书特征等,形成特征向量,作为恶意流量识别的输入。因此，本文在分析恶意流量和正常流量区别的基础上，进行特征降维，提取了其中能明显区别正常流量和恶意流量的特征因子集合。

3.3.1 文本数据处理

原数据是存在多种数据类型的，而其中字符型的数据不能直接用于训练模型，所以我们对其进行硬编码，并将其保存为 *json* 格式的文件便于查找字符值和数字的对应关系，具体保存在'*col_dict.json*'文件中。在文件中，我们可以清晰的显示出该条目下的字符型数据同数字的映射关系，我们将字段名映射为‘-1’便于按字段查找。*Json* 格式保存格式: [[["*family*", −1], ["*bankerx*", 1], ["*unclassified*", 2], ["*bunitu*", 3], ["*upatre*", 4]]] 等。

3.3.2 数据探索

我们利用 *Describe*()函数，可探索数据的均值、标准差、最小值、最大值、分数位。部分结果如表 1 所示。

表 1 数据的均值、标准差、最小值、最大值、分数位

seg_route_to	*Bytes_in*	*Bytes_out*	*validity*	*family*
count	333	333	333	333
mean	16361	4348	210	4
std	21427	10191	321	2
min	106	353	5	1
0.25	1451	849	30	2
0.5	3888	953	83	4
0.75	26011	1011	182	6
max	65349	63837	1185	8

由表 1 我们发现某些字段中存在较为极端的大值与小值，因此在此处我们采用标准化来处理数据.

3.3.3 数据标准化

在统计学中，归一化用于创建统计数据的移位和缩放版本，其主要目的是消除不同量纲对数据产生的影响。如果不同维度的特征值差距较大或者分布不够均匀，则会导致在模型的训练过程中，训练时间较长，并且模型的识别效果也会受此影响。这时，采用归一化处理可以很好的解决这一问题。我们选择的是 *min-max* 归一化方法对原始的完整数据进行线性变换，将其映射到[0,1]，其归一化公式为：

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

3.3.4 相关性分析

通常情况下，针对特定目标的算法处理过程并不会用到原始的所有数据，必然存在一些与目标无关的数据。因此，不加选择地全部利用，不仅会造成时间和精力的浪费，而且还可能影响到最终的准确率。因此，对与目标相关的数据集进行有选择的提取，可减轻项目的工作量并提高模型的准确性和目标性。

我们采用皮尔逊相关系数来进行相关性的分析。皮尔逊相关系数又称“皮尔逊积矩相关系数”，对两个定距变量的关系强度的测量，*Pearson* 相关系数公式如下：

$$\rho_{X,Y} = corr(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu X)(Y - \mu Y)]}{\sigma_X \sigma_Y} \tag{2}$$

由公式可知，*Pearson* 相关系数是用协方差除以两个变量的标准差得到的，虽然协方差能反映两个随机变量的相关程度（协方差大于 0 的时候表示两者正相关，小于 0 的时候表示两者负相关），但是协方差值的大小并不能很好地度量两个随机变量的关联程度，为了更好的度量两个随机变量的相关程度，引入了 *Pearson* 相关系数，其在协方差的基础上除以了两个随机变量的标准差，容易得出，*Pearson* 是一个介于-1 和 1 之间的

值，当两个变量的线性关系增强时，相关系数趋于 1 或-1；当一个变量增大，另一个变量也增大时，表明它们之间是正相关的，相关系数大于 0；如果一个变量增大，另一个变量却减小，表明它们之间是负相关的，相关系数小于 0；如果相关系数等于 0，表明它们之间不存在线性相关关系。

我们利用皮尔逊相关系数对数据中的每个字段与目标字段计算其相关性，然后按由大到小的顺序用图像显示出来，如图 9 所示。由于图像中过于密集，不便于查看相关性。为了更加直观的显示变量之间的相关关系，因此接下来我们将相关性的绝对值按从大到小进行排序后，生成了变量间的热力图，如图 10 所示。

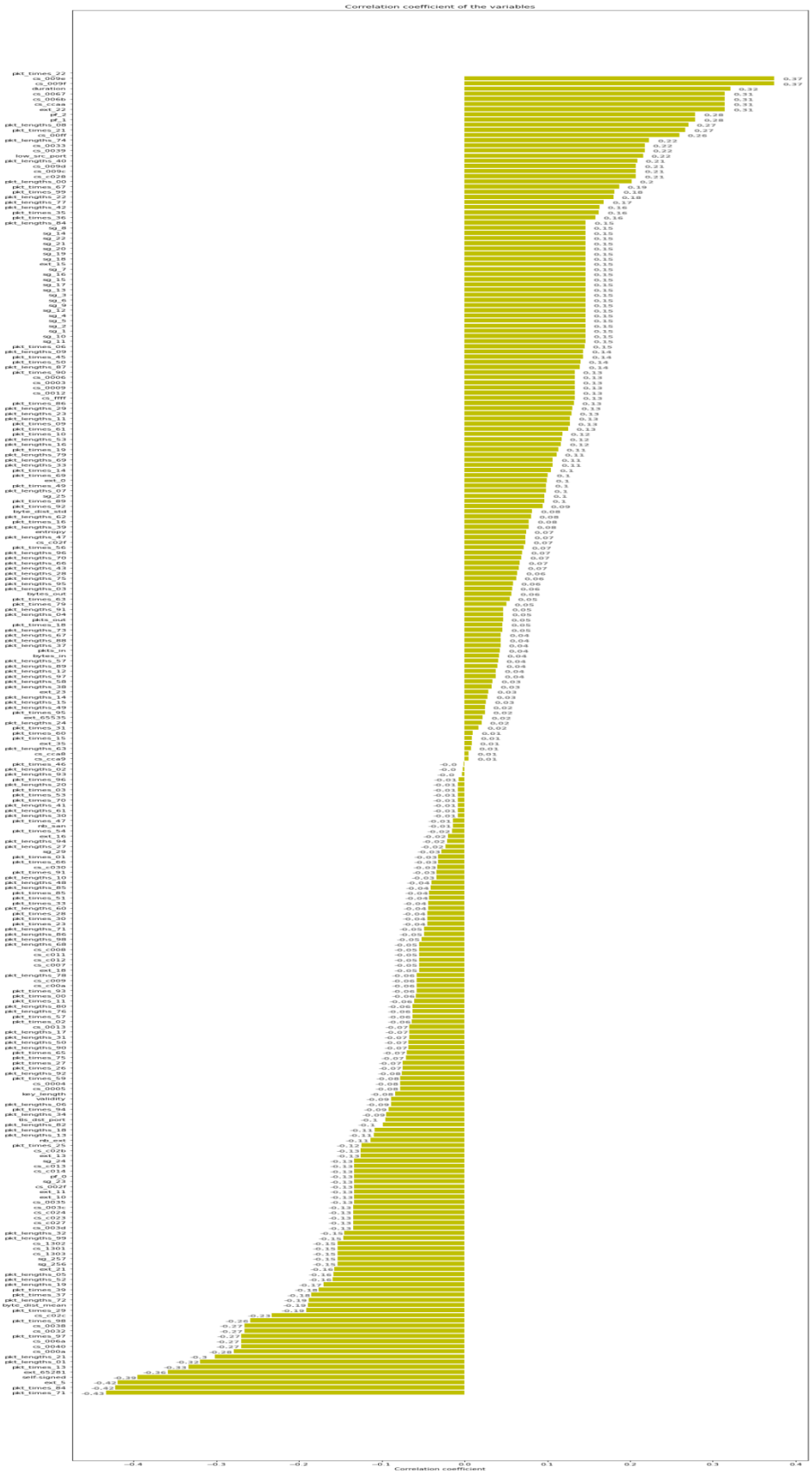

图 9 变量与目标之间的相关关系

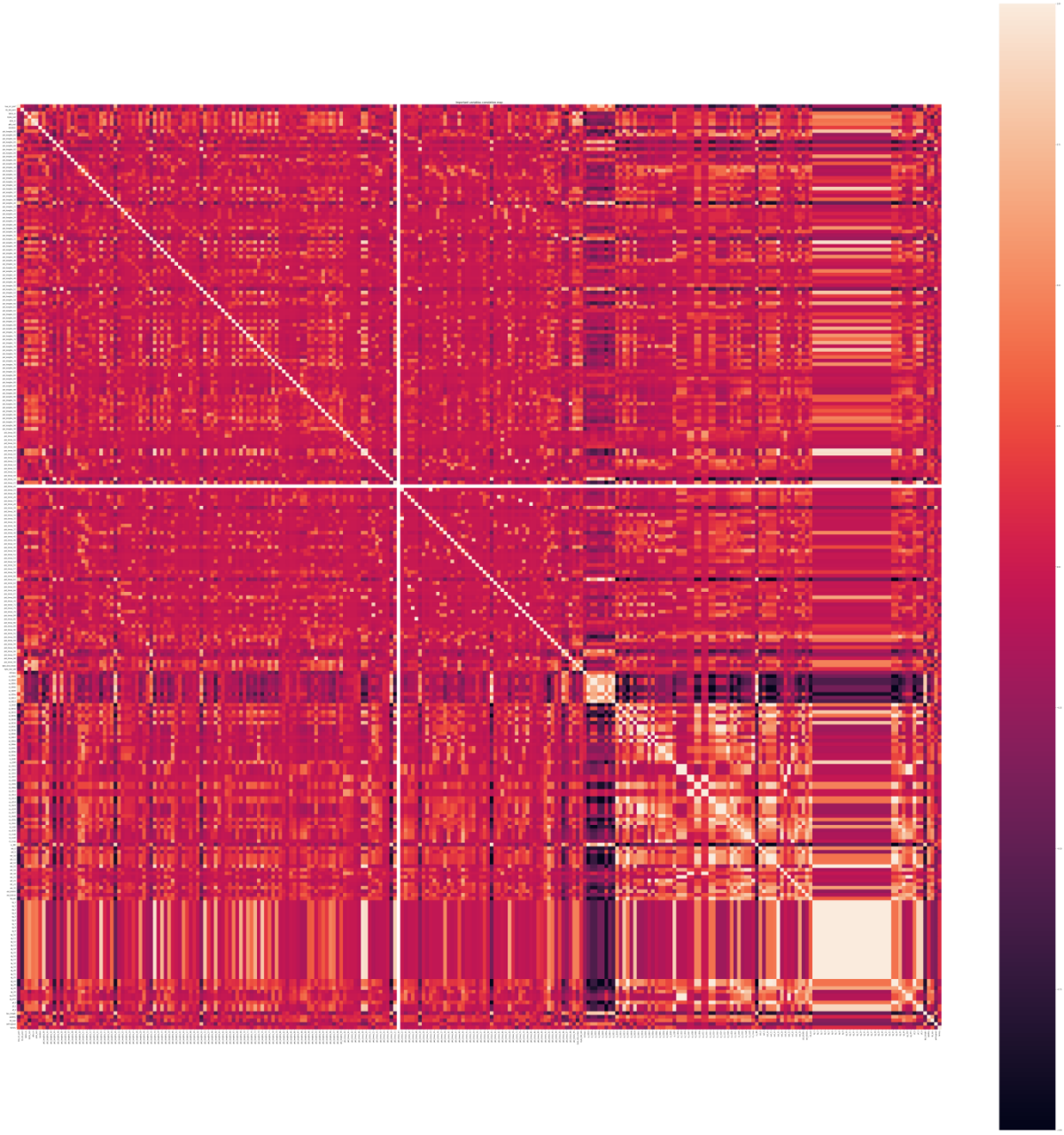

图 10 变量间的热力图

3.3.5 特征降维

在本案例中给出了多个特征因子，通常来说，这些变量都不是各自独立的，它们之间往往存在着某种联系，并且相互影响，相互制约。由于变量之间具有一定的相关性并且样本数据过于庞大，所以会导致数据分析和解决问题的复杂程度增大，未经处理的数据维度通常很高，并且存在很多冗余的属性特征，无法获得数据的真实特征。因此，我们需要对原始数据进行降维从而简化问题。

原来共有 421 个特征因子，本文在分析恶意流量和正常流量区别的基础上，通过数据预处理和特征降维后，提取了 298 个特征因子。我们将提取的特征因子集合保存在表

“*out-train-nor5-298d.csv*”中。

类似的，恶意流量家族中一共是 44 种恶意流量类型，共计 16279 条，通过数据标准化、相关性分析进行特征提取后，利用 macro 评价指标保留了数量较多的前 15 个类别。如图 11 所示。因为这 15 个类别几乎占了总的流量的 99%以上，所以其余类别可以忽略不计。最终用恶意家族的这 15 个类别进行多类别的分类问题。恶意家族的 15 个类别如下所示：

["*miuref*",1],["*hancitor*",2],["*dreambot*",3],["*emotet*",4],["*icedid*",5],["*troldesh*",6],["*vawtrak*",7],["*zeus*",8],["*trickbot*",9],["rig",10],["*zeuspanda*",11],["*ursnif*",12],["*unclassified*",13],["*gootkit*",14],["*dridex*",15]

	precision	recall	f1-score	support
1	0.9130	0.9438	0.9282	89
2	0.9873	0.9957	0.9915	234
3	1.0000	0.9706	0.9851	34
4	0.9832	0.9590	0.9710	122
5	1.0000	0.8710	0.9310	31
6	1.0000	1.0000	1.0000	101
7	0.9844	1.0000	0.9921	63
8	0.9881	1.0000	0.9940	83
9	0.9945	1.0000	0.9972	1438
10	0.9200	0.8519	0.8846	27
11	0.9811	0.9811	0.9811	53
12	1.0000	0.9583	0.9787	72
13	1.0000	0.9870	0.9935	77
14	0.9394	0.9118	0.9254	34
15	0.9986	0.9986	0.9986	705
accuracy			0.9908	3163
macro avg	0.9793	0.9619	0.9701	3163
weighted avg	0.9909	0.9908	0.9908	3163

图 11 恶意家族占比 99%以上的 15 个类别的评分

3.4 模型训练

异常检测平台的目标是发现偏离基线的行为，即“正常”(最好是良性)行为。但是首先，必须建立一个正常行为的模型。主要应用机器学习进行恶意流量检测，这里有两种方法:

1.监督学习。

在这种情况下，我们可以访问一组 n 个观察值，每个观察值有 p 个特征的数据。此外，可以利用相应的 n 个响应，来训练和验证模型，从而预测新的观测结果。

2.无监督学习

在这种情况下，我们只能利用其特征的观察集，发现观察结果之间有趣的关系，从而将它们分为子群体。

本文将重点关注监督学习，监督学习更适合我们的初始目标，也就是通过获取标签

集对恶意流量进行识别、分类。此外，监督学习可以更好地理解所获得的结果，从而帮助分析师了解恶意软件流量与正常流量的确切区别。虽然监督机器学习方法已经广泛应用于异常检测，但是，关于 *TLS* 异常检测的文献并不多，因此，基于监督学习的 *TLS* 恶意流量检测具有广大的研究前景。接下来我们将进行 *LightGBM 相关理论介绍。*

提升树是利用加法模型和前向分布算法实现学习的优化过程，它有一些高效的实现，如 *GBDT, XGBoost* 和 *pGBRT*，其中 *GBDT* 是通过损失函数的负梯度拟合残差，*XGBoost* 则是利用损失函数的二阶导展开式拟合残差。但是，当面对大量数据集和高维特征时，其扩展性和效率很难令人满意，最主要的原因是传统的 *boosting* 算法需要对每一个特征都扫描所有的样本数据来获得最优切分点，这个过程是非常耗时的。为了解决这种在大样本高纬度数据的环境下耗时的问题，我们采用了基于 *GBDT* 的另一种形式 *LightGBM*----基于直方图的切分点算法。

3.4.1 GradientBoosting

Boosting 是用一系列子模型的线性组合来完成学习任务的，它分为两种类型 *AdaBoosting* 和 *GradientBoosting*，*LightGBM* 属于 *GradientBoosting* 的一种。*GradientBoosting* 的思想是：一次性迭代变量，迭代过程中，逐一增加子模型，并且保证损失函数不断减小。假设 $f_i(X)$ 为子模型，复合模型为：

$$F_m(X) = \partial_0 f_0(X) + \partial_1 f_1(X) + \cdots + \partial_m f_m(X) \tag{3}$$

损失函数为 *L[Fm(X),Y]*,每一次加入新的子模型后，使得损失函数不断朝着信息含量次高的变量的梯度减小：

$$L[Fm(X),Y] < L[Fm\text{-}1(X),Y] \tag{4}$$

3.4.2 决策树

决策树(*DecisionTree*)是一种分类和回归的方法，实际研究中大多用于分类。决策树的结构呈树形结构，大多运用的是二权树，在每一个叶子节点上，根据某一判断条件，输出“符合条件”和“不符合条件”两类，不断重复向下输出，如图 12。可以把决策树理解成众多 *if-then* 规则的集合，也可以认为是定义在特定空间与类空间上的条件概率分布。决策树的创建包括 3 个主要步骤：特征选择、决策树的生成和决策树的修剪，该方法具有可读性高、分类速度快的优点。

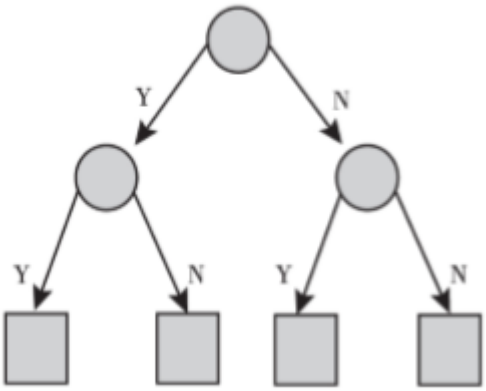

图 12　决策树结构

决策树的分裂方法分为两类，一类是按叶子分裂的学习方法(*Leaf-wiseLearning*)；另一类是按层分裂的学习方法(*Level-wiseLearning*)。按叶子分裂的学习方法是指在分裂的过程中要不断地寻找分裂后收益最大的节点，对其进行进一步的分裂，其他非收益最大化的结点不再继续分裂，以这样的规则生长这棵树。这样做的优点是可以使算法更加快速有效；缺点是会忽略掉那些被舍弃的叶子上的信息，导致分裂结果不够细化。图 13 描述的就是按叶子分裂的过程。

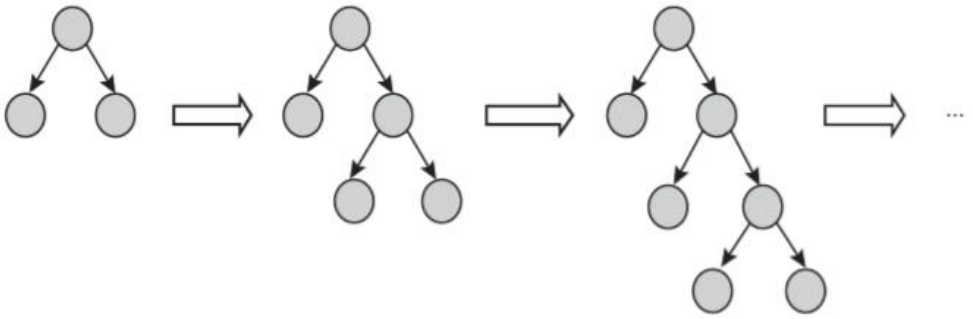

图 13 按叶子分裂的决策树学习过程

按层分裂的学习方法与按叶子分裂的学习方法不同，如图 14，它不需要挑选收益最大化的节点，每一层的每一个结点都要进行分裂，也就是说每次迭代都要遍历整个训练数据的所有数据。优点是每一层的叶子可以并行完成，具有天然的并行性；缺点是这样会产生很多没有必要的分裂，需要更多的计算成本，同时，也会占用较大的运行内存。

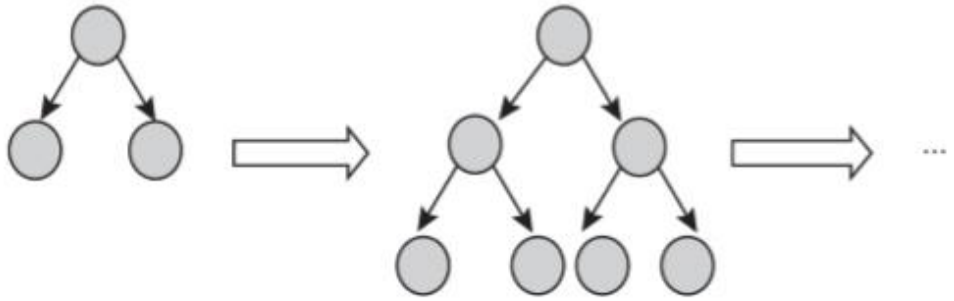

图 14 按层分裂的决策树学习过程

3.4.3 GBDT

*GBDT(GradientBoostingDecisionTree)*是机器学习中一个长盛不衰的模型，事实上：

GBDT＝GradientBoosting+DecisionTree

即若 *GradientBoosting* 中的每一个子模型都是一个 *DecisionTree*，这样的模型就是 *GBDT*。*GBDT* 拥有着 *GradientBoosting* 和 *DecisionTree* 的功能共同特性，具有训练效果好、不易过拟合等优点。*GBDT* 的工具主要包括 *XGBoosting*、*Pgbrt*、*Sklearn*、*R.GBM* 等。*GBDT* 在工业界应用广泛，通常被用于点击率预测，搜索排序等任务。*GBDT* 也是各种数据挖掘竞赛的致命武器，据统计 *Kaggle* 上的比赛有一半以上的冠军方案都是基于 *GBDT*。

3.4.4 LightGBM

LightGBM 是 *GBDT* 的一种，被提出的主要原因是为了解决 *GBDT* 在海量数据遇到的问题，让 *GBDT* 可以更好更快地用于实践。

LightGBM 中的决策树子模型是采用按叶子分裂的方法分裂节点的，因此它的计算代价比较小，也正是因为选择了这种分裂方式，需要控制树的深度和每个叶子节点的最小数据量，从而避免过拟合现象的发生。*LightGBM* 选择了基于 *Histogram* 的决策树算法，将特征值分为很多个小“桶”，进而在这些“桶”上寻找分裂，这样可以减小储存成本和计算成本。另外，类别特征的处理，也使得 *LightGBM* 在特定数据下有比较好的提升。*LightGBM* 分为三类：特征并行、数据并行和投票并行。特征并行运用在特征较多的场景，数据并行应用在数据量较大的场景，投票并行应用在特征和投票都比较多的场景。

LightGBM 通过表 2 几个主要的参数实现算法控制与参数优化：

表 2 *LightGBM* 的参数说明

参数	特点
metric	*mae*：平均绝对误差 *mse*：均方误差 *binary_logloss*：二元分类的损失 *multi_logloss*：多元分类的损失
objective	默认值为 *regression*, 其他的选择有： *regression,fair,poisson,quantile,quantile_l2,binary,multiclass,lambdarank* 等
boosting_type	*Gbdt*：传统的梯度提升决策树 *Rf*：*RandomForest*(随机森林) *Dart*：*DropoutsmeetMultipleAdditiveRegressionTrees* *Goss*：*Gradient-basedOne-SideSampling*(基于梯度的单侧采样)
learning_rate	若在最近的 *early_stopping_round* 回合中没有提高，模型将停止训练默认值为 0.1
is_unbalance	用默认值为 *false*，用于 *binary* 分类，如果培训数据集不平衡，则设置为 *true*
num_class	默认值为 1，用于 *multiclass* 分类。
nthread	*LightGBM* 的线程数 为了更快的速度, 将此设置为真正的 *CPU* 内核数, 而不是线程的数量
verbosity	默认值为 1
num_iterations	*Boosting* 的迭代次数。*LightGBM* 对于 *multiclass* 问题设置 *num_class*num_iterations* 棵树

4. 应用效果

4.1 流量描述

大量相关的数据是实现有监督分类器的基础。本节介绍了数据来自何处和何时。

收集的良性流量来自中型和面向信息技术的办公室之类的企业环境。目前网上的资源大多侧重于明文协议，如 *HTTP* 或 *DNS*，而且这些资源的捕获文件中良性和恶意软件流之间的区别通常是模糊的。因此，我们选择将恶意软件在捕获其流量之前放在隔离的沙箱中，仅保留提及感染后加密流量的捕获文件。恶意软件数据集已 *GoogleDrive(https://drive.google.com/open?id=1TfRz6q65wPaiuB4D9qmyfCxoJ8zEBUQY)* 上公开。它由三个主要来源构成:

- *malware-traffic-analysis.net*:收集了从 2014 年 1 月到 2019 年 6 月包含 *TLS* 的所有捕获文件。
- S*tratosphereIPS*:提供了几种长期捕获 *D*1 使用 *TLS* 进行通信的恶意软件.
- *Lastline*:保存网络传感器检测到的恶意软件的非常短的捕获文件。保留从 2019 年 4 月 24 日到 2019 年 5 月 7 日的两周内的 *TLS* 恶意软件中捕获的所有信息。

4.2 评价指标

4.2.1 评估函数

- 机器学习问题之中，通常需要建立模型来解决具体问题，我们通常采用一些评价指标来评估模型的泛化能力，比如准确率、精确率、召回率、*F*1 值、*Micro-F*1、*Macro-F*1 等指标。
- 混淆矩阵

我们用一个例子来解释混淆矩阵。假如现在有一个二分类问题，那么预测结果和实际结果两两结合会出现如表 8 所示的四种情况。由于用数字 1、0 表示不太方便阅读，我们转换一下，用 *T(True)*代表正确、*F(False)*代表错误、*P(Positive)*代表 1、*N(Negative)* 代表 0。先看预测结果(*P|N*)，然后再针对实际结果对比预测结果，给出判断结果*(T|F)*。按照上面逻辑，重新分配后为如表 9 所示。在表 9 中，*TP*、*FP*、*FN*、*TN* 可以理解为:

- *TP*：预测为 1，实际为 1，预测正确。
- *FP*：预测为 1，实际为 0，预测错误。
- *FN*：预测为 0，实际为 1，预测错误。
- *TN*：预测为 0，实际为 0，预测正确。

表 3 基于二分类问题的混淆矩阵_1

		实际结果	
		1	0
预测结果	1	11	10
	0	01	00

表 4 基于二分类问题的混淆矩阵_2

		实际结果	
		1	0
预测结果	1	*TP*	*FP*
	0	*FN*	*TN*

- 准确率

准确率*(Accuracy)*即预测正确的结果占总样本的百分比，表达式为：准确率*=(TP+TN)/(TP+TN+FP+FN)*。

虽然准确率能够判断总的正确率，但是在样本不均衡的情况下，并不能作为很好的指标来衡量结果，所以，我们需要寻找新的指标来评价模型的优劣。

- 精确率

精确率(*Precision*)是针对预测结果而言的，其含义是在被所有预测为正的样本中实际为正样本的概率，表达式为：精确率*=TP/(TP+FP)*.

精确率和准确率看上去有些类似，但是却是两个完全不同的概念。精确率代表对正样本结果中的预测准确程度，准确率则代表整体的预测准确程度，包括正样本和负样本。

- 召回率

召回率(*Recall*)是针对原样本而言的，其含义是在实际为正的样本中被预测为正样本的概率，表达式为：召回率(*Recall*)*=TP/(TP+FN)*.

- *F1* 分数

在实际生产中，我们希望精确率和召回率都很高，但实际上上述两个指标是矛盾体，无法做到双高。因此，提出了新的指标 *F*1 分数(*F1-Score*)。其同时考虑精确率和召回率，让两者同时达到最高，取得平衡。表达式为：*F*1 分数=(2*精确率*召回率)/(精确率+召回率).

- *Micro-F*1

统计各个类别的 *TP*、*FP*、*FN*、*TN*，加和构成新的 *TP*、*FP*、*FN*、*TN*，然后计算 *Micro-Precision* 和 *Micro-Recall*，得到 *Micro-F*1。具体的说，统计出来各个类别的混淆矩阵，然后把混淆矩阵“相加”起来，得到一个多类别的混淆矩阵，然后再计算 *F1score*

- *Macro-F*1

我感觉更常用的是 *Macro-F1*。统计各个类别的 *TP*、*FP*、*FN*、*TN*，分别计算各自的 *Precision* 和 *Recall*，得到各自的 *F*1 值，然后取平均值得到 *Macro-F*1。

从上面二者计算方式上可以看出，*Macro-F*1 平等地看待各个类别，它的值会受

到稀有类别的影响；而 *Micro-F*1 则更容易受到常见类别的影响。

4.2.2 损失函数

KS(Kolmogorov-Smirnov)：*KS* 用于模型风险区分能力进行评估，指标衡量的是好坏样本累计分部之间的差值。好坏样本累计差异越大，*KS* 指标越大，那么模型的风险区分能力越强。具体公式如下所示：

$$ks = max(\frac{Cum.B_i}{Bad_{total}} - \frac{Cum.G_i}{Good_{total}}) \tag{5}$$

损失函数（*LossFunction*）：是定义在单个样本上的，是指一个样本的误差。其意义是预测值和目标值直接的差距。损失函数越小，表示预测值更接近目标值，神经网络越收敛。

Loss 曲线的评价标准如下：

- 学习率过高，一开始 *loss* 曲线会下降很快，然后不会再下降，表明陷入了局部极小值，很容易出现参数爆炸现象。
- 学习率过低，*loss* 曲线表现出线性(下降缓慢)，极其耗时；
- 学习率恰好，*loss* 曲线会平滑下降。

4.3 LightGBM 算法应用

4.3.1 数据集划分

划分训练集、验证集、测试集。按照发布比例划分，训练集和验证集的比例为 3:1，训练集和测试集的比例为 4:1。其中，*x_train* 表示训练集数据，*x_dev* 表示验证集数据，*x_test* 表示测试集数据，*y_train* 表示训练集标签，*y_dev* 表示验证集标签，*y_test* 表示测试集标签。

4.3.2 模型训练

表 5 *LightGBM* 算法的参数值

参数	初始值	参数	初始值
boosting_type	gbdt	*nthread*	4
objective	multiclass	*learning_rate*	0.1
is_unbalance	True	*verbosity*	-1
metric	multi_logloss	*num_iterations*	150000
num_class	class_num		

模型的参数值具体设置如表 5 所示。

将经过调优的参数代入到模型进行训练，模型结果如图 15 所示。从图 15 中，我们可以观察到，随着迭代次数的增加，训练集和验证集的 *multi_logloss* 和 *multi_error* 值越来越低，且越趋于稳定。

```
[1]     training's multi_logloss: 0.864777      training's multi_error: 0.339709        valid_1's multi_loglo
ss: 0.885329    valid_1's multi_error: 0.340816
Training until validation scores don't improve for 10 rounds
[2]     training's multi_logloss: 0.676577      training's multi_error: 0.0377015       valid_1's multi_loglo
ss: 0.702669    valid_1's multi_error: 0.042681
[3]     training's multi_logloss: 0.544783      training's multi_error: 0.0237907       valid_1's multi_loglo
ss: 0.573952    valid_1's multi_error: 0.0300348
[4]     training's multi_logloss: 0.443992      training's multi_error: 0.0178628       valid_1's multi_loglo
ss: 0.47467     valid_1's multi_error: 0.0256086
[5]     training's multi_logloss: 0.364475      training's multi_error: 0.0139899       valid_1's multi_loglo
ss: 0.396777    valid_1's multi_error: 0.0227632
[6]     training's multi_logloss: 0.300577      training's multi_error: 0.00987986      valid_1's multi_loglo
ss: 0.335028    valid_1's multi_error: 0.0199178
[7]     training's multi_logloss: 0.248888      training's multi_error: 0.00703446      valid_1's multi_loglo
ss: 0.284454    valid_1's multi_error: 0.0180209
[32]    training's multi_logloss: 0.0029154     training's multi_error: 0       valid_1's multi_logloss: 0.04
10391   valid_1's multi_error: 0.00822004
[33]    training's multi_logloss: 0.00245863    training's multi_error: 0       valid_1's multi_logloss: 0.04
06358   valid_1's multi_error: 0.00822004
[34]    training's multi_logloss: 0.00207631    training's multi_error: 0       valid_1's multi_logloss: 0.04
0279    valid_1's multi_error: 0.00790389
[35]    training's multi_logloss: 0.00174466    training's multi_error: 0       valid_1's multi_logloss: 0.04
01138   valid_1's multi_error: 0.00790389
[36]    training's multi_logloss: 0.0014676     training's multi_error: 0       valid_1's multi_logloss: 0.03
94677   valid_1's multi_error: 0.00790389
[37]    training's multi_logloss: 0.00123554    training's multi_error: 0       valid_1's multi_logloss: 0.03
92281   valid_1's multi_error: 0.00790389
[38]    training's multi_logloss: 0.00103884    training's multi_error: 0       valid_1's multi_logloss: 0.03
9211    valid_1's multi_error: 0.00790389
[39]    training's multi_logloss: 0.000872528   training's multi_error: 0       valid_1's multi_logloss: 0.03
88819   valid_1's multi_error: 0.00822004
Early stopping, best iteration is:
[29]    training's multi_logloss: 0.00489597    training's multi_error: 0       valid_1's multi_logloss: 0.04
26103   valid_1's multi_error: 0.00790389
```

图 15 基于 *LightGBM* 算法的训练结果

4.3.3 模型评估

图 16 是 *multi_error* 图，我们可以看出随着迭代次数的增加，训练集和验证集的 *multi_error* 曲线陡然下降，并在迭代 10 次时逐渐趋于平稳，说明模型的性能较好。

图 17 是 *multi_logloss* 图，我们可以看出随着迭代次数的增加，训练集和验证集的 *multi_logloss* 曲线逐渐下降，并在迭代 15 次时趋于平稳，说明模型的性能较好。

我们将 *multi_error* 和 *multi_logloss* 保存在“*evals_result.json*”文件中。

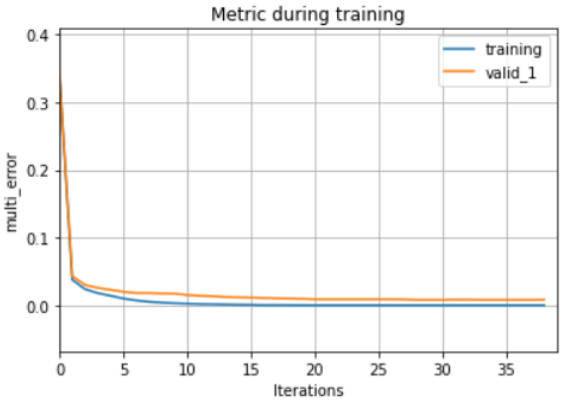

图 16 multi_error 图

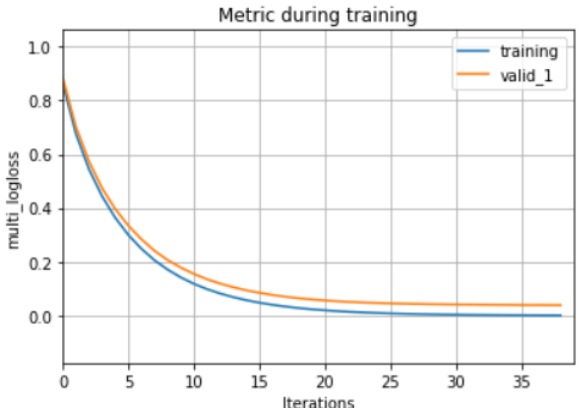

图 17 *multi_logloss* 图

4.4 预测结果

- 数据预处理

我们将测试集与训练集经过相同的数据预处理。

- 特征降维过程

在分析恶意流量和正常流量区别的基础上，进行特征降维，提取了其中能明显区别正常流量和恶意流量的特征因子集合。

- 模型预测结果

将处理过的测试集代入所训练好的 *LightGBM* 算法中，验证结果如图 18 所示，我们将具体数值保存在文件”*confu.csv*”中。从图 18 我们可知准确率为 0.9921，精确率为 0.9921，召回率为 0.9921，*f*1 分数为 0.9921，恶意家族的 15 个分类的平均多分类精确值为 0.9797，平均多分类召回率为 0.9683，平均多分类 *f*1 分数为 0.9735，以上数据说明恶意流量的多分类的结果较好。为了更直观的查看混淆矩阵，我们将其可视化，如图 19 所示。

图 20 是基于 *LightGBM* 算法的测试数据评估的热力图，可以更简洁明了的看出基于 *LightGBM* 算法的模型分类效果较好。我们具体数值如表 6 所示，也可以在文件”*result_df2.csv*”中找到。从表 6 我们可以看到，恶意流量类别 4,6,10,13,14 的分类效果是最高的，精确率、召回率和 *f*1 分数都达到了 100%。

预测的最终结果如图 21 所示，也可在文件“*pre_result.json*”中找到。从图 21 我们可以看到，在恶意流量类别中，*icedid*、*zeus_panda*、*trickbot* 是出现频率最高的。

```
metrics.accuracy_score:
0.9920961112867531
metrics.confusion_matrix:
[[ 83   0   1   0   0   0   1   0   1   0   0   1   2   0
    0]
 [  0  26   0   0   0   0   0   0   0   0   4   1   0   0
    0]
 [  0   0  34   0   0   0   0   0   0   0   0   0   0   0
    0]
 [  0   0   0  72   0   0   0   0   0   0   0   0   0   0
    0]
 [  0   1   0   0  76   0   0   0   0   0   0   0   0   0
    0]
 [  0   0   0   0   0 101   0   0   0   0   0   0   0   0
    0]
 [  1   0   0   0   0   0 120   0   0   0   0   0   0   0
    1]
 [  0   0   0   0   0   0   0  25   0   0   0   2   0   0
    0]
 [  0   0   0   0   0   0   0   0  30   0   0   0   4   0
    0]
 [  0   0   0   0   0   0   0   0   1 704   0   0   0   0
    0]
 [  0   0   0   0   0   0   0   0   1   0 233   0   0   0
    0]
 [  0   0   0   0   0   0   0   0   0   0   0  83   0   0
    0]
 [  0   0   0   0   0   0   0   0   1   0   1   0 1436   0
    0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  63
    0]
 [  1   0   0   0   0   0   0   0   0   0   0   0   0   0
   52]]

metrics.precision_score:
0.9920961112867531
metrics.recall_score:
0.9920961112867531
metrics.f1_score:
0.9920961112867531
```

	precision	recall	f1-score	support
1	0.9765	0.9326	0.9540	89
2	0.9630	0.8387	0.8966	31
3	0.9714	1.0000	0.9855	34
4	1.0000	1.0000	1.0000	72
5	1.0000	0.9870	0.9935	77
6	1.0000	1.0000	1.0000	101
7	0.9917	0.9836	0.9877	122
8	1.0000	0.9259	0.9615	27
9	0.8824	0.8824	0.8824	34
10	1.0000	0.9986	0.9993	705
11	0.9790	0.9957	0.9873	234
12	0.9540	1.0000	0.9765	83
13	0.9958	0.9986	0.9972	1438
14	1.0000	1.0000	1.0000	63
15	0.9811	0.9811	0.9811	53
accuracy			0.9921	3163
macro avg	0.9797	0.9683	0.9735	3163
weighted avg	0.9921	0.9921	0.9920	3163

图 18 基于 *LightGBM* 算法的测试数据的评估

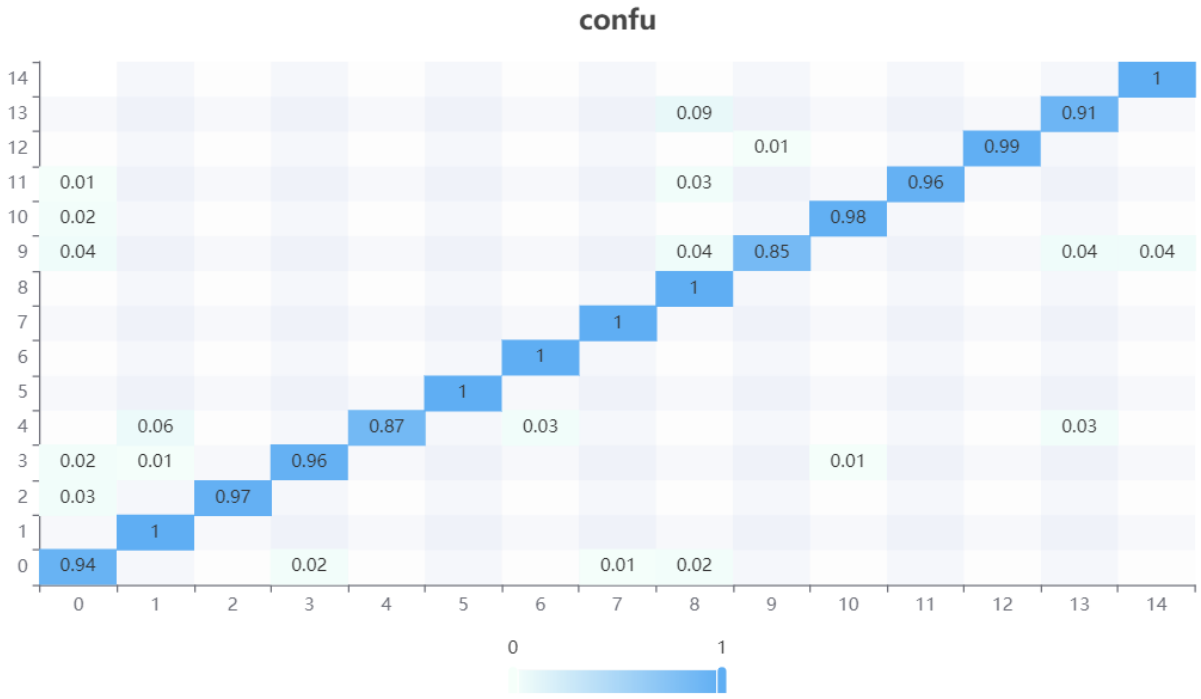

图 19 混淆矩阵

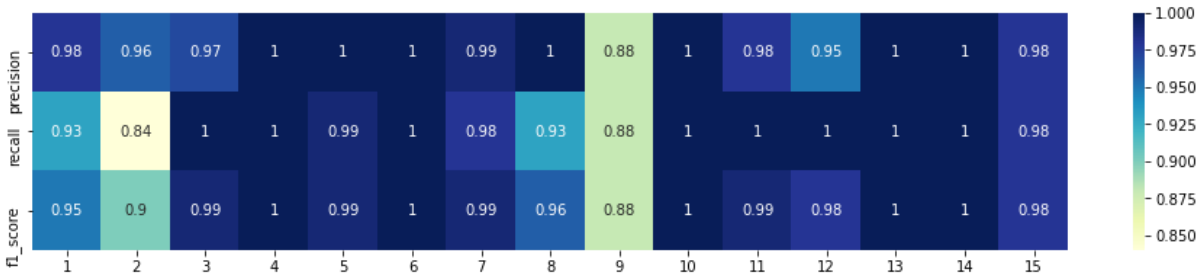

图 20 基于 *LightGBM* 算法的测试数据评估的热力图

表 6 恶意流量多分类的结果评估表

评估类别	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
precision	0.98	0.96	0.97	1	1	1	0.99	1	0.88	1	0.98	0.95	1	1	0.98
recall	0.93	0.84	1	1	0.99	1	0.98	0.93	0.88	1	1	1	1	1	0.98
f1_score	0.95	0.9	0.9	1	0.99	1	0.99	0.96	0.88	1	0.99	0.98	1	1	0.98

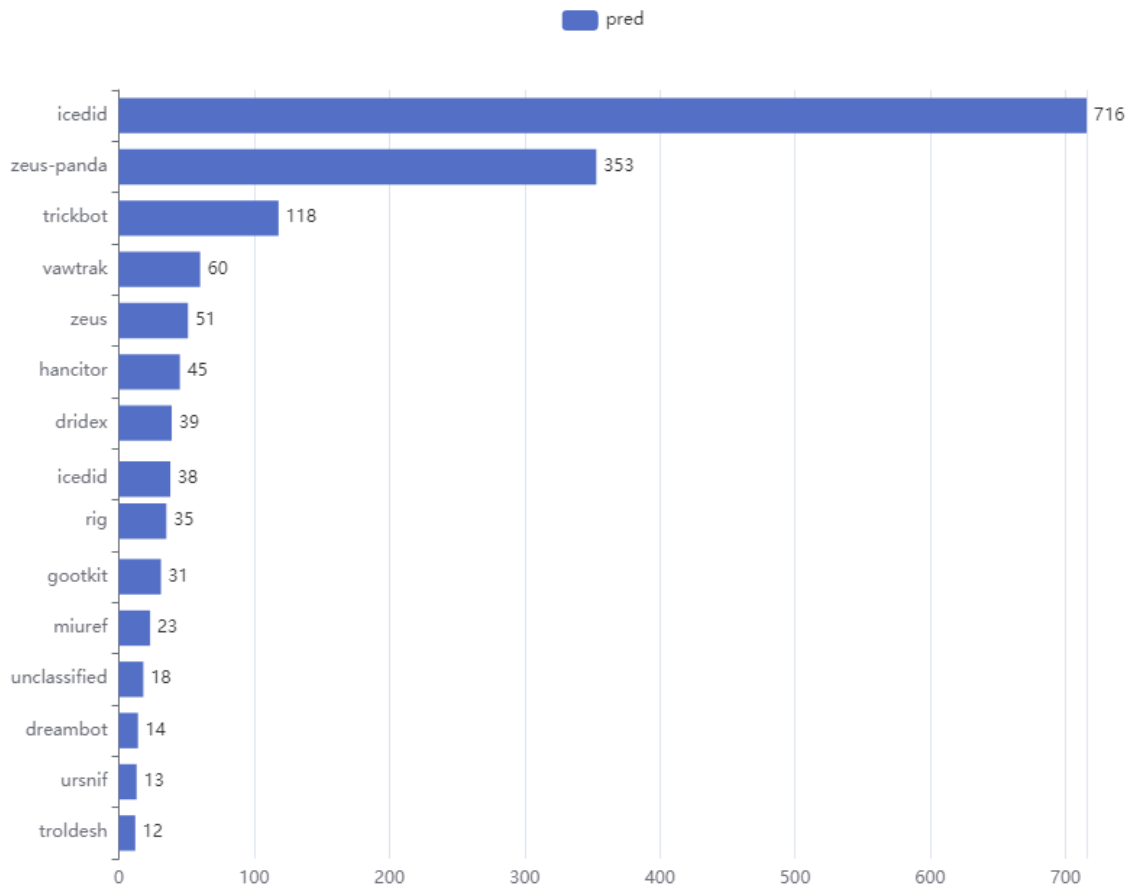

图 21 预测结果

4.5 智能识别预测恶意流量系统开发

基于 *HTML*、*css*、*js*、*jquery*、*python* 和 *React* 框架、*express* 框架完成恶意流量检测平台的架构设计，*LightGBM* 算法完成恶意流量的检测任务。

4.5.1 系统架构设计

智能识别预测恶意流量系统的架构设计分为应用层、嵌入层、业务层和数据层。

Web 应用层:在本项目中 *web* 应用层主要是用户操作的浏览器。用户在浏览器的页面上输入 *localhost* 和端口号，从而获取需要的数据和页面。

嵌入层：快捷处理图层样式、*html* 等静态文件的请求，并将后台对请求的响应发送到 *web* 应用层。此外，利用 *js* 渲染，把从后台传到 *js* 的数据展示在 *web* 界面。

业务层:业务层主要包括系统的功能模块和实现业务的前后端交互服务。系统功能模块包括流量捕获、流量预处理、特征提取、分类器训练、恶意流量检测。服务后台建立业务对应的应用，完成业务逻辑控制处理以及前端与后端的交互功能。

数据层:存储系统操作人员手动上传的数据,并存储训练节点、检测节点的相关信息。

4.5.2 功能设计

智能识别预测恶意流量系统功能包括登录功能、分类模型训练功能、恶意流量分类功能、预测模型训练功能及恶意流量预测功能。接下来我们首先介绍每个功能的流程，然后再具体描述相关界面。

登录功能：步骤如下，登录/注册➞登录➞登陆成功。具体如图 22-23 所示。

分类模型训练功能：步骤如下，训练模型➞选择训练集➞"恶意流量分类任务"按钮➞模型训练效果。

恶意流量分类功能：步骤如下，模型训练效果界面点击“下一步”按钮➞选择测试集➞恶意流量分类效果。

预测模型训练功能：步骤如下，训练模型➞选择训练集➞"恶意流量预测任务"按钮➞模型训练效果。

恶意流量预测功能：步骤如下，模型训练效果界面点击“下一步”按钮➞选择测试集➞恶意流量预测效果。

图 24 是训练模型的界面图，在此界面中，我们可以选择测试集或者训练集来训练模型。

图 25 是选择训练集的界面图，可以在此界面中选择对应的数据集进行训练。

图 26 是训练集/测试集预览，可以看到该数据集的相关详细信息。

图 27 是恶意流量分类模型训练的效果图，左上角的饼图由训练数据集的恶意流量的家族名及每个恶意流量家族的样本数构成。中上的柱状图由训练数据集的恶意流量的家族名及每个恶意流量家族的样本数构成，展示恶意流量 15 个类别对应的名称以及统计的攻击次数，横轴表示恶意流量的类别，纵轴表示该类别的攻击次数。左下角为样本每个属性之间的热量相关图，中下的两个曲线图展示了分类模型的 *multi_logloss* 和 *multi_error*，右边的图为样本每个属性与“恶意流量家族”属性的相关性。

图 28 是恶意流量分类的结果图，图左的饼图由测试数据集的恶意流量的家族名及每个恶意流量家族的样本数构成，图右的柱状图由测试数据集的恶意流量的家族名及每个恶意流量家族的样本数构成。鼠标移动到对应位置即可看到对应数据。

图 29 是恶意流量预测模型训练的效果图，左上图为训练数据集的恶意流量家族时序图，左下图由样本每个属性之间相关性的热量图和预测模型的 *multi_logloss* 和 *multi_error* 组成，右图为样本每个属性与“恶意流量家族”属性的相关性。

图 30 是恶意流量预测的结果，纵轴代表预测的恶意流量家族的类别，可解释如下：["*ursnif*",1],["*trickbot*",2],["*miuref*",3],["*emotet*",4],["*gootkit*",5],["*dreambot*",6],["*dridex*",7],["*unclassified*",8],["*hancitor*",9],["*troldesh*",10],["*zeuspanda*",11]["*zeus*",12],["*rig*",13],["*icedid*",14],["*vawtrak*",15]，在这个功能中鼠标移动到指定时刻就能看到当前时刻的预测结果。

图 22 登录/注册

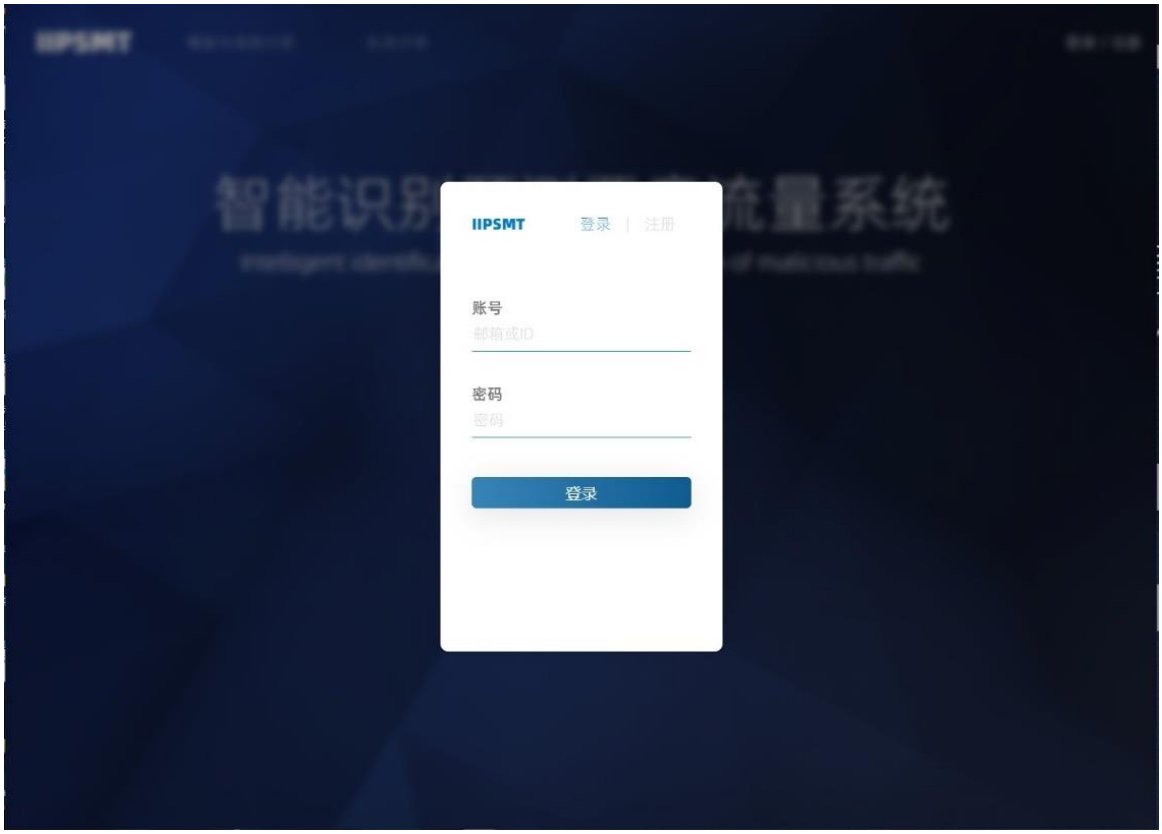

图 23 登录

图 24 训练模型

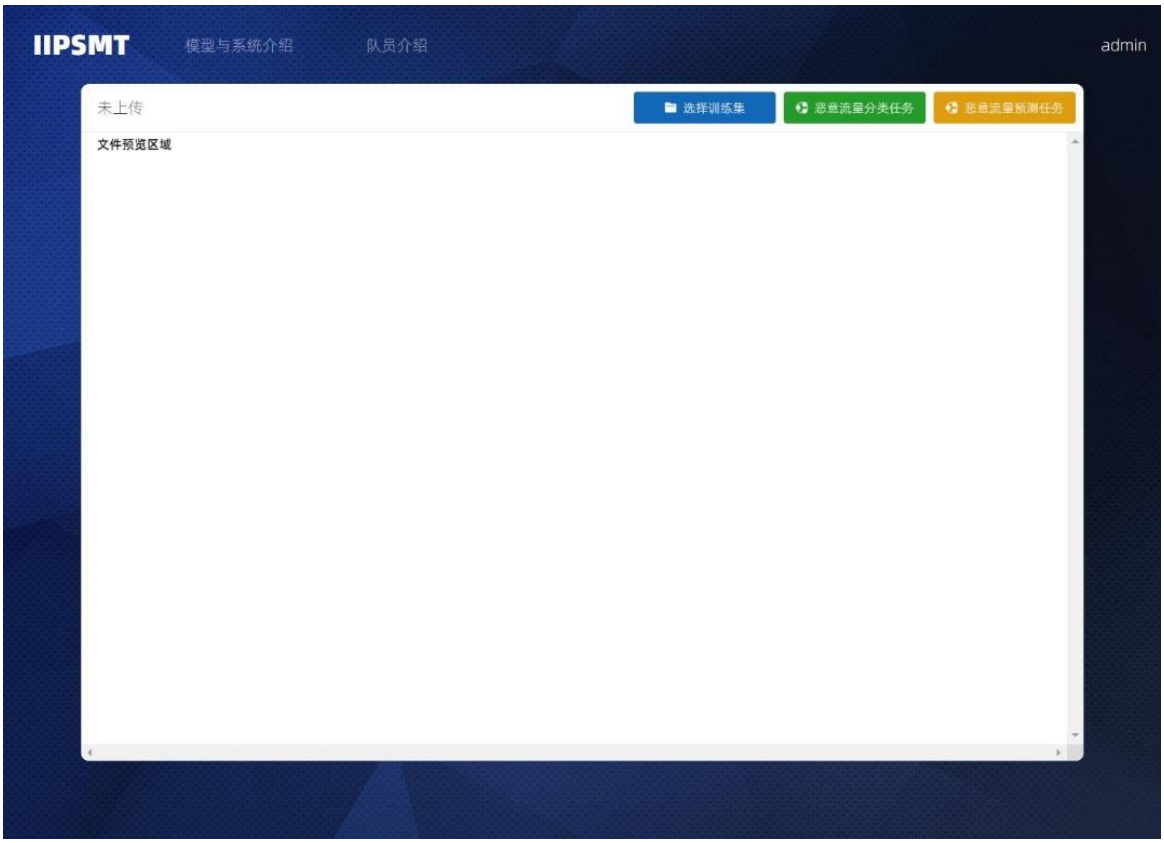

图 25 选择训练集

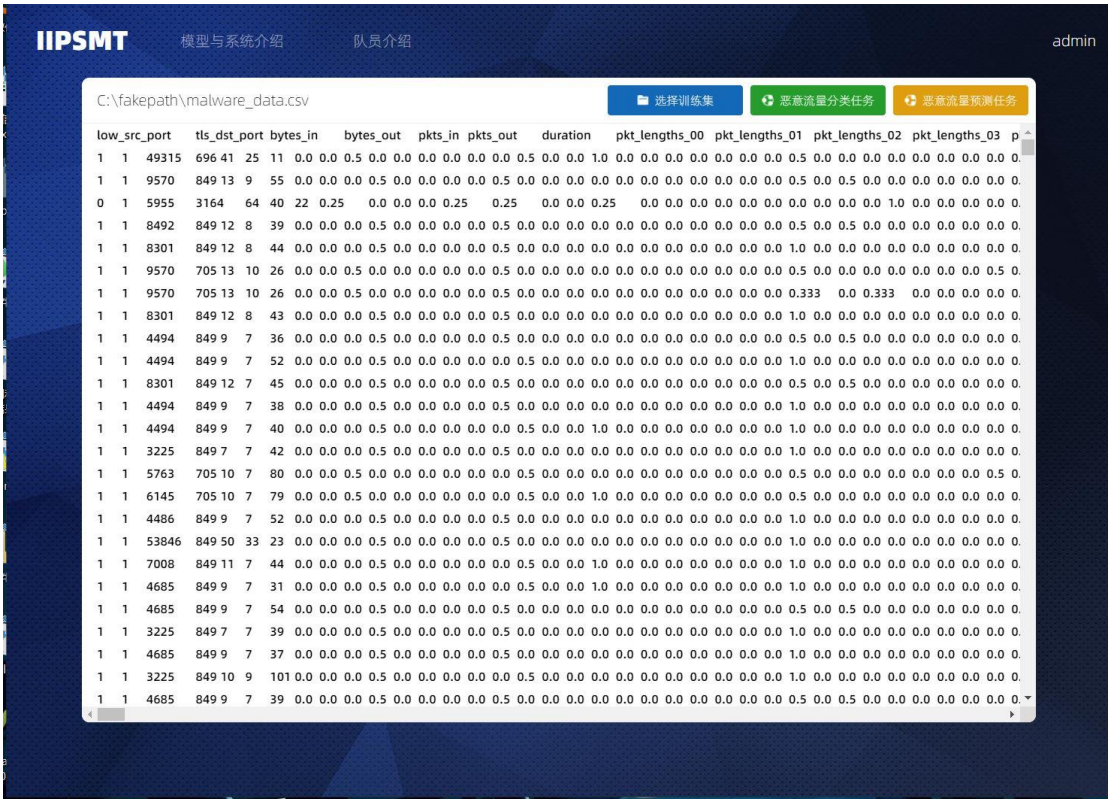

图 26 训练集/测试集详细预览

图 27 恶意流量分类模型训练效果图

图 28 恶意流量分类结果图

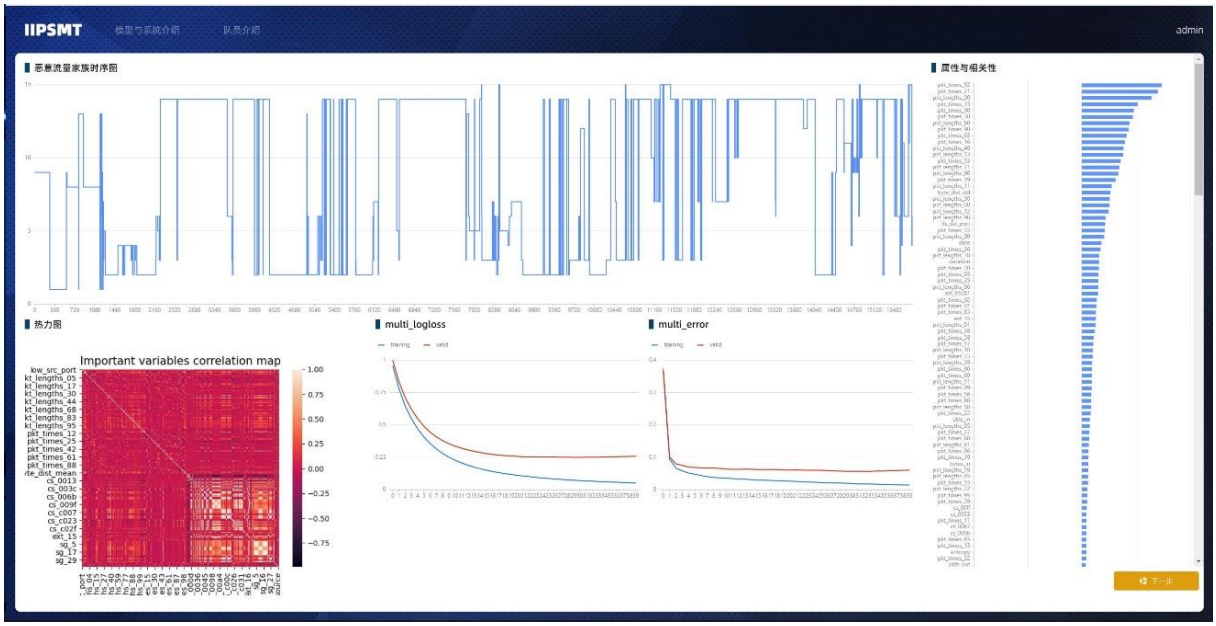

图 29 恶意流量预测模型训练效果图

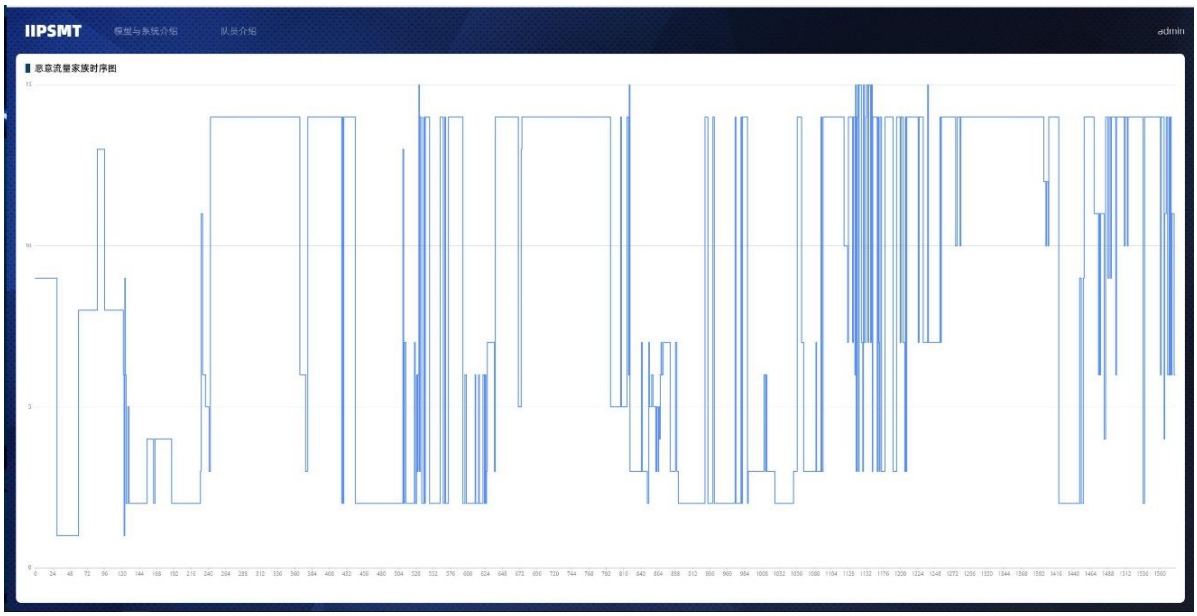

图 30 恶意流量预测结果图

5. 创新与特色

创意一：在网络安全领域融入了机器学习和人工智能、数据防泄漏等安全新理念、技术和方法。

创意二：通过用户电脑日志的流量数据，分析了恶意流量和正常流量的区别，提取了其中能明显区别正常流量和恶意流量的部分特征，包括流量元数据特征、*TLS* 握手特征和证书特征，用于恶意流量的识别、分类和预测。

创意三：基于相关性分析提取特征因子集合，既避免了信息重复又保证筛选出的指标对评价结果有显著影响，无形之中增加了模型的准确率。

创意四：充分考虑到传统的 *Boosting* 算法（如 *GBDT* 和 *XGBoost*）的不足，我们使用 *LightGBM* 构建决策树分析模型，利用上述明显区别正常流量恶意流量的特征因子集合，实现加密会话的快速精准的识别与分类，并获得较高的分类准确度效果。

创意五：充分考虑了 *LightGBM* 与其他模型的优缺点，最终，选择预测准确率较好的 *LightGBM* 算法作为本文的恶意流量预测模型。

创意六：分析了 *LightGBM* 算法在不同时间流量数据的检测效果以及不同数据集下分类的效果,验证了该模型的稳定性、鲁棒性和分类的准确度。

创意七：开发了基于前端语言、*python* 和 *React* 框架、*express* 框架的恶意流量识别、分类与预测系统。

综上所述，本项目基于 *LightGBM* 算法对加密流量进行监控不但较为准确的识别了恶意流量，方便公司提出具体可行的方案；而且本项目提出的模型是开源框架，适用性和容错性好，可广泛应用于网络安全领域。

6. 结论、展望与团队风采

6.1 结论

近年来，随着大数据和人工智能的快速发展,互联网已经在工业、金融、教育等各个领域都广泛应用。为了更好的维护网络安全，我们提出了基于 *LightGBM* 算法的恶意流量的识别、分类与预测。首先，对抓包获得的流量数据集进行预处理；其次，在通过分析大量的正常和恶意加密流量基础上，从中提取出能明显区别正常流量和恶意流量的特征因子集合；最后，使用 *LightGBM* 算法对加密流量进行训练，利用训练的模型来对恶意流量进行检识别与预测。验证结果表明，基于 *LightGBM* 算法的恶意流量的识别与分类准确率较高,且在保持相当的分类准确性的基础上,具有更好的鲁棒性,适用性更广。此外，开发了基于前端语言、*python*、*React* 框架和 *express* 框架的智能识别预测恶意流量系统。

6.2 展望

虽然本文对恶意流量识别的研究上投入了很大精力。但鉴于时间有限、自身的研究能力和实践经验不足，以及对于安全行业的理解有限，本文仍存在着一定的局限性，需要未来进一步研究，主要归纳为以下几点:

第一，本文虽然基于 *wireshark* 捕获的流量进行了分析与处理，但是基于本文捕获的流量数据集可能与其他大学、家中或具有不同活动的公司产生的流量有很大的不同，可能会在一定程度上影响了模型的准确性。因此，在数据捕获方面在未来还需要进一步研究。

第二，本文虽然对恶意流量和良性流量的部分特征进行了分析与研究，较为准确的识别了恶意流量。但仅局限于本实验所应用的数据场景，基于 *LightGBM* 算法对其他加密流量识别的有效性仍需要进一步验证。

第三，由于目前市面上的仍然是使用 *TLSv*1.2 协议作为流量加密的传输协议,因此,本研究是基于 *TLSv*1.2 的分析与测试。然而，自从 2018 年 8 月以来，*TLSv*1.3 标准已经逐步推出与推广。因此，之后对于 *TLSv*1.3 协议的加密流量检测方法的研究将逐步成为研究热点。

6.3 团队风采

队长：曾荣获全国高校绿色计算大赛全国二等奖，第九届“华为杯”中国大学生智能设计竞赛全国二等奖，全国大学生数学建模竞赛省级三等奖，美国大学生数学建模三等奖,中国大学生计算机设计大赛中南赛区三等奖,全国大学生软件测试大赛省级二等奖,中国大学生服务外包创新创业大赛中部赛区三等奖，中国高校计算机大赛——网络技术挑战赛全国二等奖，全国大学生数字媒体科技作品竞赛全国一等奖，获得国家励志奖学金一次，校级一等奖学金多次，校级学科竞赛获奖十余项。发表 *EI* 国际会议论文两篇

（第一作者），申请专利两项、软件著作权三项，主持省级课题一项，参与省级课题一项、校级课题两项，并荣获校级优秀毕业生和湖南省优秀毕业生殊荣。为加固广交会信息系统、保障广交会的安全运行做出突出贡献。在第七届中国国际"互联网+"大学生创新创业大赛荣获校赛金奖和广东省金奖、全国铜奖，挑战杯全国大学生课外学术科技作品竞赛项目立项，在“青创杯”第八届广州青创大赛中获得总决赛优胜奖，并已成立公司一个。算法和开发功底过硬，熟练掌握 *Python*、*Matlab*、*SPSS*、*IDEA* 等软件，能熟练使用决策树、支持向量机、神经网络、随机森林等训练算法和粒子群、遗传算法等优化算法。

队员 1：中国大学生服务外包创新创业大赛中部赛区三等奖，为加固广交会信息系统、保障广交会的安全运行做出突出贡献。掌握 C/C++，*python,*了解 *java* 及 *Android*。熟练使用常见机器学习算法，熟练在自然语言处理方向的深度学习方法。对信息抽取及知识图谱构建有所心得。擅长使用 *keras* 架构，熟悉 *pytorch* 及 *tensorflow*。

队员 2：曾获全国高校绿色计算大赛全国二等奖，2019 中国高校计算机大赛微信小程序应用开发赛全国三等奖，获得国家励志奖学金一次，并荣获校级优秀毕业生殊荣。熟练掌握 *Python* 和 *HTML*，*css*，*JavaScript* 等前端开发语言，算法和开发能力较强，掌握进化算法和 *svm* 等机器学习算法。

队员 3：现担任“教师一站式成长与就业输送中心”项目技术部主管曾参与本校计算机科技研究院研究“面向动态几何智能构图的自然语言理解”；2018 年随本校交流团队参与美国华盛顿大学交流;获 *certificateofexcellence* 省级大学生程序设计竞赛 ACM 铜奖；中国大学生服务外包创新创业大赛中部赛区三等奖；技术开发专业技能过硬，掌握 C++，掌握 *Python*、*PythonPytorch*、*PythonFlask*，了解 *Java*、*C#*编程以及 *Android* 开发，了解 *HTML*、*JavaScript*、*CSS* 等，掌握机器学习 *KNN*、*SVM* 等 *ML* 基本算法，在团队中担任技术开发部主管，为项目网站、小程序等平台提供支持。

队员 4：曾荣获全国高校绿色计算大赛全国二等奖，2018 全国大学生软件测试大赛嵌入式测试个人赛省二等奖，中国大学生服务外包创新创业大赛中部赛区三等奖，校级数学建模一等奖；连续两年获得两次国家励志奖学金，陈琛奖学金和校一等奖学金，并荣获省级优秀毕业生殊荣。有数据清洗和数据可视化的相关经验，熟练掌握时间序列模型、相关进化算法、机器学习算法以及深度学习模型，并能根据具体场景进行应用。

总之，只要坚持，永远相信，幸运是留给努力的人。

参考文献

[1] *C.MeyerandJ.Schwenk.LessonslearnedfrompreviousSSL/TLSattacksabriefchronologyofattacksandweaknesses.IACRCryptologyePrintArchive,2013.*

[2] *E.Rescorla.TheTransportLayerSecurity(TLS)ProtocolVersion1.3,draft-ietf-tls-tls13-19.March2017*

[3] *EncryptedTrafficAnalytics.Whitepaper.Cisco,Jan.2019.*

[4] 潘嘉,翟江涛,刘伟伟.基于改进递归残差网络的恶意流量分类算法[J].计算机应用研究,2020,37(S2):227-229.

[5] *AbbasQamarandAbbasQamberandJinJesseS..MaliciousTrafficClassifierinandroidusingNeuralNetworks[J].JournalofPhysics:ConferenceSeries,2021,1732(1):012038-.*

[6] *Google.HTTPSencryptionontheweb.Jan.2019.*

[7] *InternetEngineeringTaskForce(IETF),ed.TheTLSProtocol–Version1.3.Aug.2018.*

[8] *SSLLabs.SSLPulse.Apr.2019.*

[9] *InternationalTelecommunicationsUnion(ITU),ed.InternetX.509PublicKeyInfrastructureCertificate.May2008*

[10] *BrianKrebs.HalfofallPhishingSitesNowHavethePadlock.Nov.2018.*

[11] *HiddenThreatsinEncryptedTraffic.Researchreport.PonemonInstitute,May2016.*

[12] 李慧慧,张士庚,宋虹,王伟平.结合多特征识别的恶意加密流量检测方法[J].信息安全学报,2021,6(02):129-142.

[13] 骆子铭,许书彬,刘晓东.基于机器学习的 TLS 恶意加密流量检测方案[J].网络与信息安全学报,2020,6(01):77-83.

[14] *DetectingMalwareinTLSTrafficProjectReport*

[15] 网络安全应急技术国家工程实验室发布的报告

附件

附件一

表 7 恶意流量 15 个类别对应的名称以及识别次数

	Maliciousstrafficcategory	***times***
1	*miuref*	92
2	*hancitor*	236
3	*dreambot*	33
4	*emotet*	119
5	*icedid*	27
6	*troldesh*	101
7	*vawtrak*	64
8	*zeus*	84
9	*trickbot*	1446
10	*rig*	25
11	*zeus-panda*	53
12	*ursnif*	69
13	*unclassified*	76
14	*gootkit*	33
15	*dridex*	705

附件二界面具体图例介绍

图 27 恶意流量分类模型训练效果图的具体图例（包括图 30-图 33）

图 30 左图为训练数据集的恶意流量的家族名及每个恶意流量家族的样本数构成的饼图，右图为训练数据集的恶意流量的家族名及每个恶意流量家族的样本数构成的柱状图

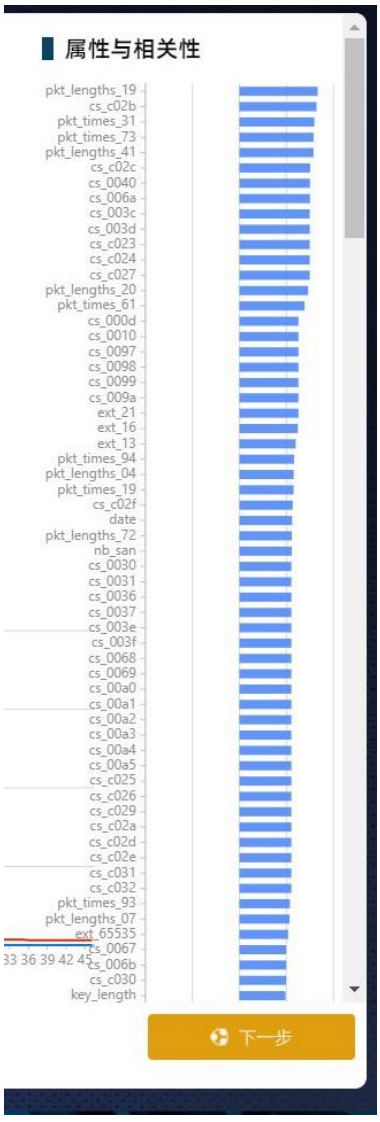

图 31 样本每个属性与“恶意流量家族”属性的相关性

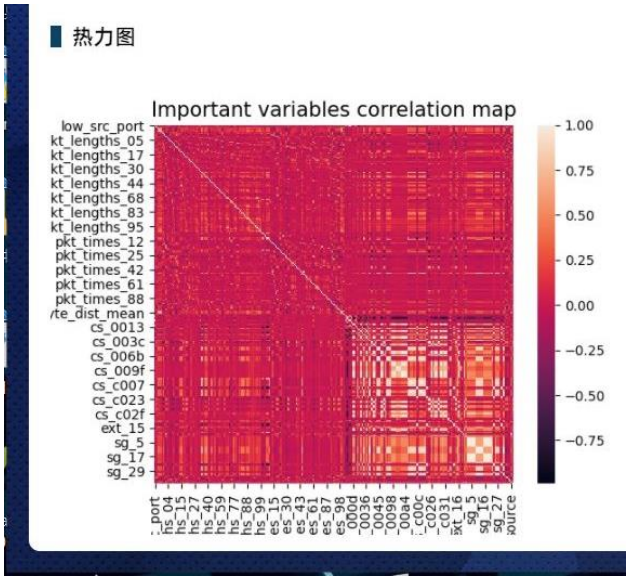

图 32 样本每个属性之间的相关性

图 33 分类模型的 *multi_logloss* 和 *multi_error*

图 29 恶意流量预测模型训练效果图的具体图例（包括图 34-37）

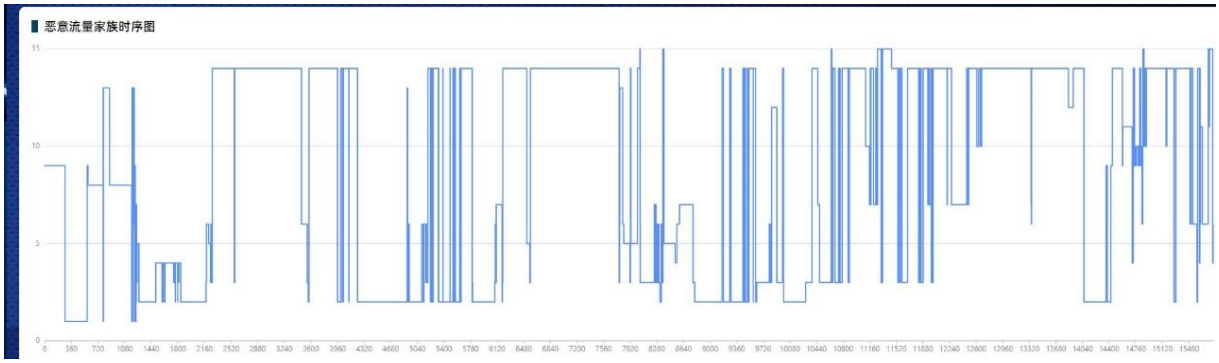

图 34 训练数据集的恶意流量家族时序图

图 35 样本每个属性与“恶意流量家族”属性的相关性

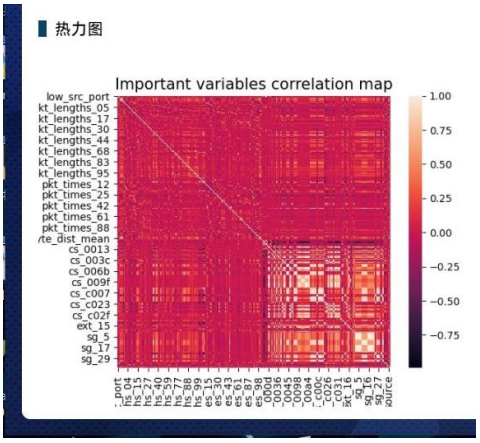

图 36 为样本每个属性之间的相关性

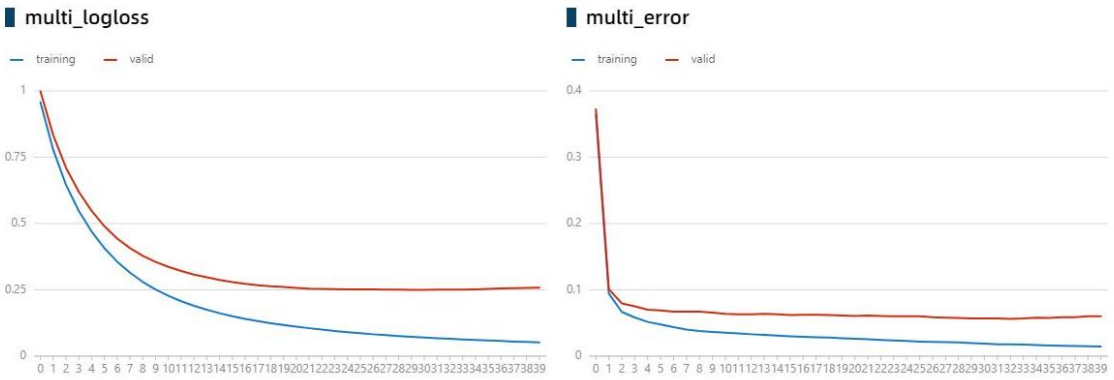

图 37 预测模型的 *multi_logloss* 和 *multi_error*