基于机器学习的流量分析识别系统

设计文档

一、 所在系列及赛项

B-EP2

二、 目标问题与意义价值

目前，各种类型的流量充满了网络空间，其中包含了正常上网业务，具有 CVE 编号的网络攻击、计算机病毒（邮件病毒、木马、蠕虫、勒索软件）等等。如何开发出一个网络安全系统，能够有效抵对流量数据进行检测、分析与分类，并实时识别出网络上的恶意攻击行为成为现阶段面临的一个挑战。目前在工业界，已经已经有了基于网络端口映射的流量分类识别方法和基于有效载荷分析的流量分类识别方法等。但这些方法面临两个都面临了准确性和可靠性低的问题。

我们提出将网络系统与机器学习相结合，利用 CyberFlood 产生不同种类的业务流量和恶意流量，通过 Python 对流量进行抓取并以 URL 的形式存储在 MySQL 数据库中，然后通过 Python 将流量数据从 MySQL 数据库中导出并进行特征提取和特征选择，然后对集成学习模型进行训练，最终得到可以对不同种类流量进行分类的集成学习模型。该模型可以在以 Bootstrap 前端框架和 PHP 语言搭建的 Web 服务中进行线上推断。我们基于机器学习技术进行流量分析与分类，同时具有高准确性和高实时性特点，可以解决传统安全系统所面临的的挑战。

目前我们的系统版本为 V1.0，可以识别出业务流量、恶意软件流量和网络攻击流量这三种流量类型，并进行动态流量数据的可视化呈现。

三、 设计思路与方案

3.1 目标

(1) 对 CyberFlood 产生的流量数据进行抓取、分析与分类。
(2) 实时识别出网络上的正常业务流量、网络攻击流量，恶意软件流量。
(3) 实现一个可交互的网络流量监测系统，并且能对流量数据可视化。

3.2 系统架构

系统架构大致分为以下几个部分：
(1) 利用 CyberFlood 产生不同种类的流量
(2) 利用 Python 抓取流量
(3) 将流量以 URL 的形式存储在 MySQL 数据库中
(4) 数据预处理
(5) 特征选择
(6) 使用集成学习模型得到初步结果
(7) 集成学习模型准确度评估
(8) 模型调试调优
(9) 最终结果可视化呈现

我们的系统分为离线训练部分和线上推断部分。

离线训练部分如下：首先，通过组委会提供的 CyberFlood 工具，生成指定类型的正常业务流量与网络攻击行为的流量，之后利用 Python 将流量抓取后以 URL 的形式存入数据库，接着基于 Python 对数据进行数据预处理与特征选择，然后使用带标签的流量数据对集成学习模型进行训练和交叉验证，然后对模型进行准确的评估，通过不断地调试调优改进模型。

线上推断部分如下：通过组委会提供的 CyberFlood 工具，生成随机的带有正常与网络攻击行为的流量，之后利用 Python 将流量抓取后以 URL 的形式存入数据库，接着基于 Python 对数据进行数据预处理与特征选择，然后集成学习模型对流量数据进行分析与分类，最终 Python 程序将模型的推断结果存入数据库中，PHP 将数据从数据库中导出并将结果通过前端可视化呈现。

整体系统架构如图 1 所示。

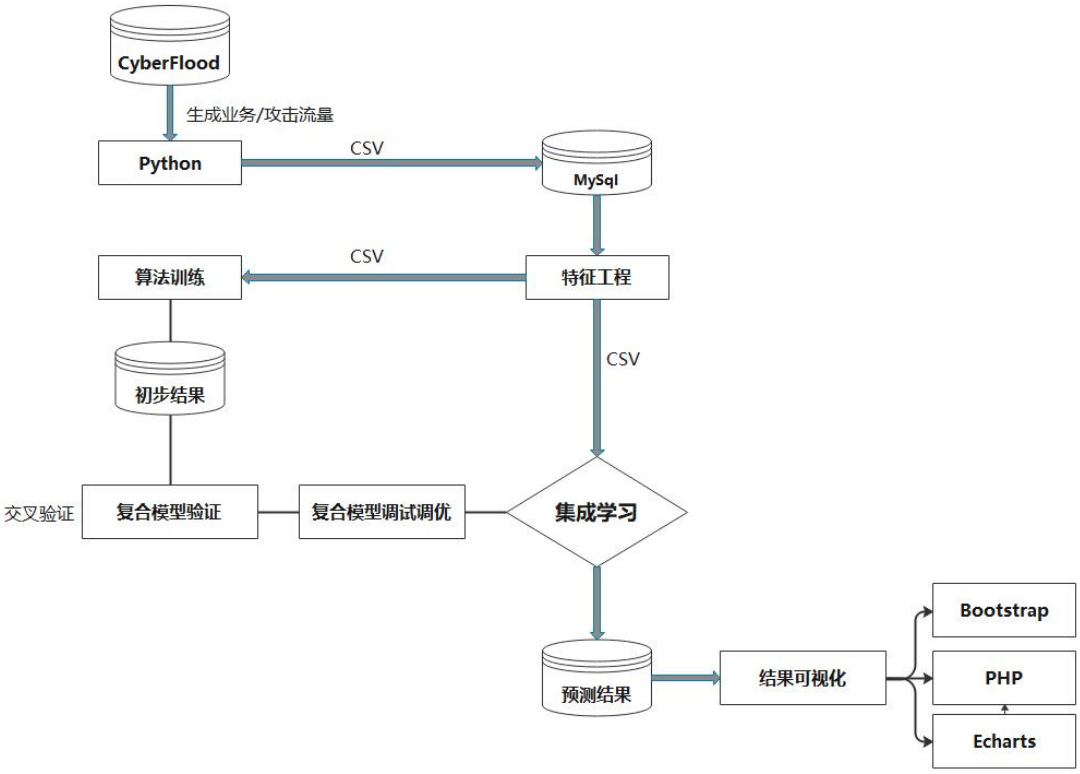


图 1 系统架构

四、 实现

4.1 数据采集

数据集通过组委会提供的 CyberFlood 工具，生成正常业务流量与网络攻击行为的流量，再使用 Python 对指定的网络端口抓取流量。数据采集流程如图 2 所示。

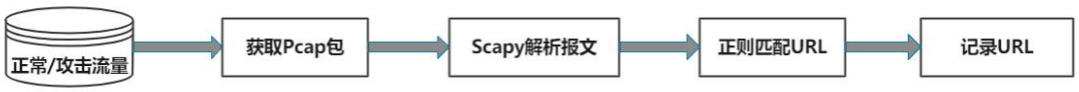


图 2 数据采集流程

4.2 流量存储

业务类型或攻击类型流量都以 CSV 文件格式保存在 MySQL 数据库中，其中每条流量记录的包括 ID 字段，URL 字段与时间戳字段。原始流量数据存储如表 1 所示。

ID	URL	Time
1	GET /rtv.png?plf=3	2020-7-29 16:44:00
2	GET /default.aspx?wa=wsignin1.0	2020-7-29 19:50:00
3	GET /nagiosql/config/content.css	2020-7-30 20:13:00
4	GET /Interop.MessengerAPI.dll	2020-7-30 13:05:00

表 1 原始流量数据存储

后期通过模型推断，完成分类后的流量则存储如表 2 所示，可通过 PHP 读取该表数据从而完成前端动态流量可视化呈现。其中 0 标签表示为业务流量，1 标签表示为网络攻击流量，2 标签表示为恶意软件流量。

ID	Label	Time
1	0	2020-7-29 16:44:00
2	0	2020-7-29 19:50:00
3	1	2020-7-30 20:13:00
4	2	2020-7-30 13:05:00

表 2 流量预测结果存储

4.3 数据预处理

该部分由数据导入、特征提取三部分组成。先使用 Python 的 PyMySQL 库将数据从 MySQL 中提取出来，以 CSV 表格数据形式存储，然后使用 Python 的 Pandas 库读取 CSV 格式的数据表格，以 DataFrame 形式存于内存，之后利用文本处理中常用的技术“单词向量空间模型”(word vector space model)由字符型的 URL 构建数值特征。数据预处理流程如图 3 所示。

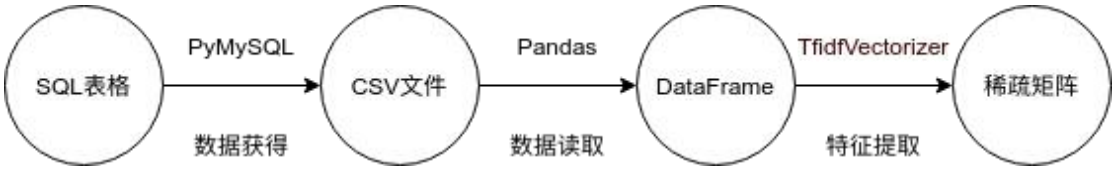


图 3 数据预处理

（1）数据获得

使用 Python 的 PyMySQL 库将数据从 MySQL 中提取出来，以 CSV 表格数据形式存储。

（2）数据读取

然后使用 Python 的 Pandas 库读取 CSV 格式的数据表格，将m条n维的流量记录以 DataFrame 格式存于m行n列的矩阵X中。

（3）特征提取

由于流量数据为 URL 格式，读取后得到的是一个文本数据，设使用正则表达式解析预处理和分词之后的流量数据为单词-文本矩阵，m条流量的 URL 数据对应m条文本，每条文本的向量空间为单词向量空间(word vector space)，即对每一条流量 URL 文本，用一个向量表示该文本的“语义”，向量的每一维对一个一个单词，其数值为该单词在该文本中出现的权值。

形式化的定义如下，给定一个含有m条文本的集合$D = \{d_1, d_2, \ldots, d_n\}$，以及在所有文本中出现的$n$个单词的集合$W = \{w_1, w_2, \ldots, w_n\}$。将单词在文本中出现的数据用一个单词－文本矩阵(word-document matrix)表示，记作X

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

这是一个$m \times n$矩阵，元素X_{ij}表示单词w_j在文本d_i中出现的频数或权值。由于单词的种类很多，而每个文本中出现单词的种类通常较少，所以单词－文本矩阵是一个稀疏矩阵。

权值我们用单词频率－逆文本频率（term frequency-inverse document frequency,TF-INDF）表示，其定义是

$$\mathrm{TFIDF}_{ij} = \frac{\mathrm{tf}_{ij}}{\mathrm{tf}_{i\cdot}} \log \frac{\mathrm{df}}{\mathrm{df}_j}, i = i, 2, \ldots, m; j = i, 2, \ldots, n$$

式中tf_{ij}是单词w_j在文本d_i中出现的频数，$\mathrm{tf}_{i\cdot}$是文本d_i中出现的所有单词的频数之和，df_j是含有单词w_j的文本数，df是文本集合D的全部文本数。也就是说一个单词在一个 URL 中出现的频数越高，这个单词在这个 URL 中的重要度就越高；一个单词在整个 URL 集合中出现的 URL 条数越少，这个单词就越能表示其所在的 URL 的特点，重要度就越高；一个单词在一条 URL 中的 TF-IDF 是两种重要度的积，表示综合重要度。

单词向量空间模型直接使用单词－文本矩阵的信息。单词－文本矩阵的第 i 行向量x_i表示 URL d_i

$$x_i = [x_{i1}, x_{i2}, \ldots, x_{im}], i = 1,2, \ldots, n$$

其中x_{ij}是单词w_j在 URL d_i中的权值，$i = 1,2, \ldots, m$，权值越大，该单词在该 URL 中的重要程度就越高。这时矩阵X也可以写作$X = \begin{bmatrix} x1 \\ \vdots \\ xm \end{bmatrix}$。

4.4 特征选择

对于数据预处理后的的特征选择部分，采用 XGBoost 模型对特征进行选择。该算法为基于决策树的梯度提升算法，可以自动地获取特征的重要性，从而有效地进行特征的筛选。

一般来说，特征的重要性表示这个特征在构建提升树的作用。如果一个特征在所有树中作为划分属性的次数越多，那么该特征就越重要。通过每个属性分割点改进性能度量的量来计算单个决策树的重要性，并由节点负责的观察数量加权。性能度量可以是用于选择分裂点的纯度(基尼指数)或另一个更具体的误差函数。最后，在模型中的所有决策树中对要素重要性进行平均。最终得到每个特征的重要性,之后可以特征排序或者进行相互比较。

我们已经在数据读取流程中拆分出训练集和验证集，然后在训练集上训练 XGBoost 模型，用验证集来验证模型的准确率。此外，基于训练 XGBoost 得到的 feature_impoerance,通过 SelectFromModel 库函数进行特征选择，并比较不同特征重要性阈值下的准确率。

我们在实验中发现随着特征重要性阈值的增加，选择特征数量的减少，模型的准确率也在下降。

于是，我们认为必须在模型复杂度（特征数量）和准确率做一个权衡，但是有些情况，特征数量的减少反而会是准确率升高，因为这些被剔除特征是噪声。我们最终的特征选择流程如图 4 所示。

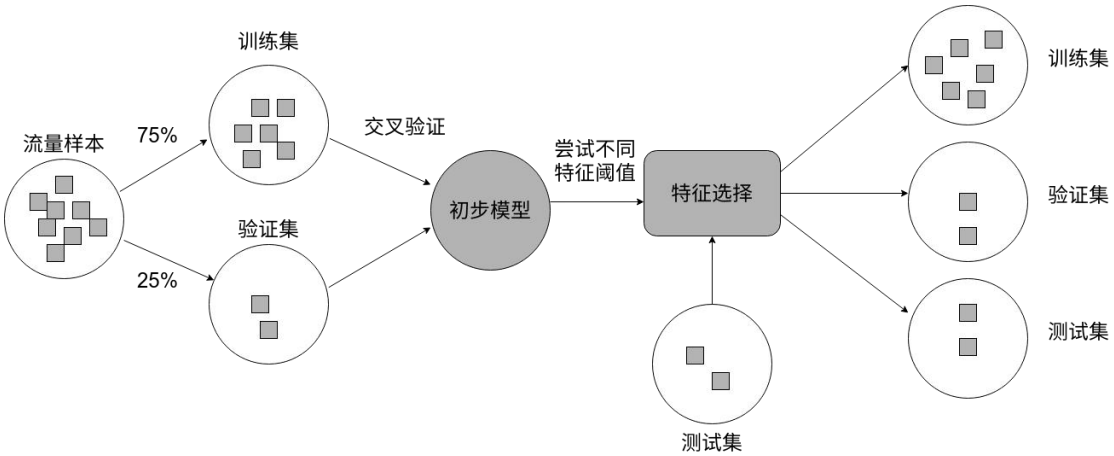


图 4 特征选择

4.5 模型训练

机器学习的流量分类研究所面临的两大挑战.特征偏置是指一些数据流统计特征在提高部分应用识别准确率的同时也降低了另外一部分应用识别的准确率.

类别不平衡是指机器学习流量分类器对样本数较少的应用识别的准确率较低.为解决上述问题,通过多模型融合（集成学习）的手段可以有效解决上述问题，

将 URL 的文本特征建模为行向量作为特征取值，列向量为不同 URL 请求的矩阵。

我们在模型上采用模型融合方法。常见的模型融合方法有一下三种：Bagging、Boosting、Stacking。

(1) Bagging

Bagging 将多个模型，也就是多个基学习器的预测结果进行简单的加权平均或者投票。它的好处是可以并行地训练基学习器。Random Forest 就用到了 Bagging 的思想。

(2) Boosting

Boosting 的思想有为每个基学习器是在上一个基学习器学习的基础上，对上一个基学习器的错误进行弥补。AdaBoost，Gradient Boost 就用到了这种思想。

(3) Stacking

Stacking 是用新的次学习器去学习如何组合上一层的基学习器。如果把 Bagging 看作是多个基分类器的线性组合，那么 Stacking 就是多个基分类器的非线性组合。Stacking 可以将学习器一层一层地堆砌起来，形成一个网状的结构。

相比来说 Stacking 的融合框架相对前面的二者来说在精度上确实有一定的提升，所以在下面的模型融合上，我们也使用 Stacking 方法。

考虑到我们的模型要处理大规模稀疏矩阵，而 XGBoost 模型对稀疏矩阵中大量出现的 0 采取了缺失值处理方式大大节省了内存空间，加快了模型训练速度。故我们的第一层的基分类器和第二层的组合分类器都采用 XGBoost 模型。

XGBoost 处理稀疏矩阵的原理为:对稀疏矩阵中大量出现的 0 采取了缺失值处理，即没有缺失的值上分裂，然后把缺失值分别带入每个树节点的左节点算一下分裂后的增益，再带入右节点算一下分裂后的增益然后去其中大的一个作为最终的分裂方案。如果训练中没有数据缺失，预测时出现了数据缺失，则默认被分类到右节点。

这里我们使用了两层的模型融合，Level 1 使用了 4 个 XGBoost 分类器，Level 2 又使用了一个 XGBoost 分类器对第一层预测的结果作为特征对最终的结果进行预测。该分类器原理如图 5 所示。

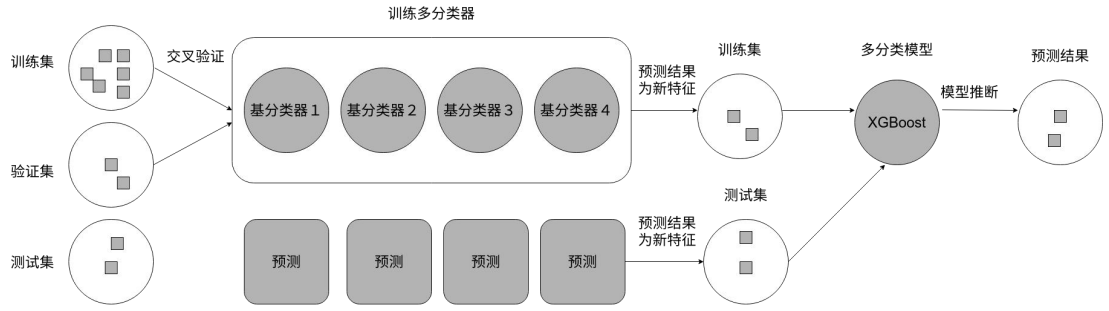


图 5 Stacking 模型融合思想

最终我们的恶意流量检测模型的训练流程如图 6 所示。

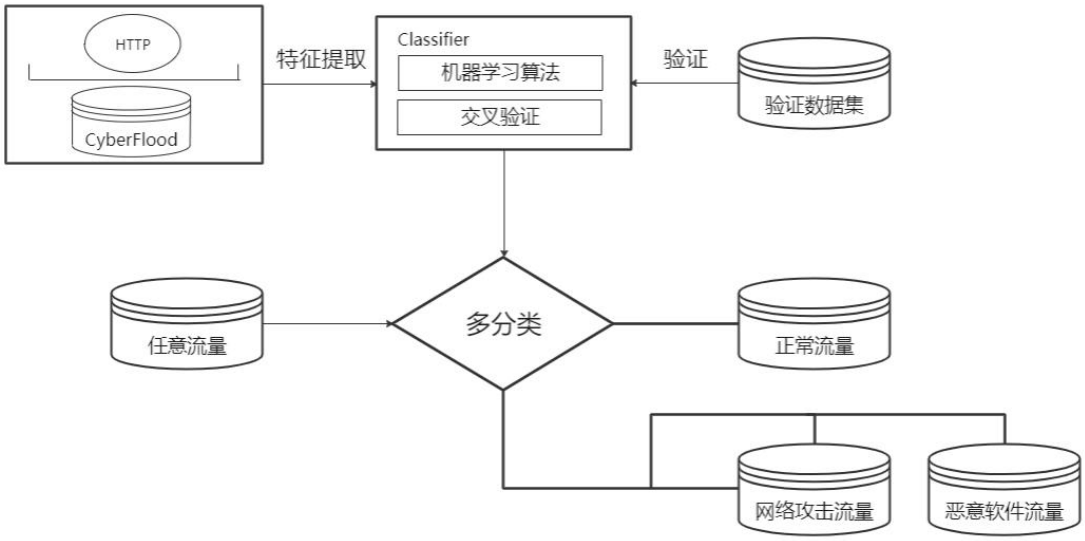


图 6 模型训练流程

为了避免测试的偶然性，本项目采用交叉验证方法，数据集分为验证集测试集，以验证集得到的正确率作为精度。

4.6 模型评价

我们评价分类器的标准包括整体准确率（OA）、查准率（P）、查全率(R)和综合评价(F1)。其中，查准率、查全率和综合评价代表分类器对每种类别的识别能力，整体准确率代表分类器整体的识别能力。

我们将类别为 K 定义为正例，类别为非 K 定义为负例。各种指标中相关参数如表 3 所示。

评价标准	参数意义
TP(true positives)	类别为 K 的流被判为类 K 的流个数
TN(true negatives)	类别非 K 被判为非 K 的流的个数
FP(false positives)	类别非 K 但判为 K 的流的个数
FN(false negatives)	类别为 K 但判为非 K 的流的个数

表 3 评价标准

整体准确率（OA）的表达式如式(1)所示：

$$OA = \frac{\sum_{i=1}^{n}(TP_i + TN_i)}{\sum_{i=1}^{n}(TP_i + FP_i + TN_i + FP_i)} \tag{1}$$

查准率（P）和查全率(R)的定义式如式(2)和式(3)所示：

$$P = \frac{TP}{TP + FP} \tag{2}$$

$$P = \frac{TP}{TP + FN} \tag{3}$$

综合评价 F1-Score 的定义如式(4)所示：

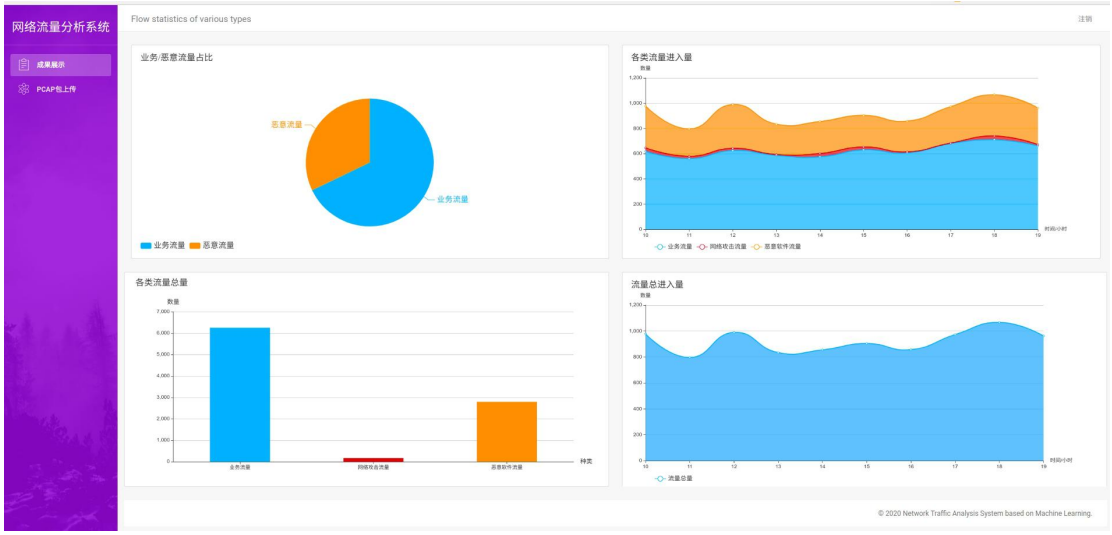
$$F_1 = \frac{2PR}{P + R} = \frac{2TP}{2TP + FP + FN} \tag{4}$$

目前我们模型的流量分类预测整体准确率达到了 80.5%，查准率为 86.2%，查全率为 79.3%，综合评价得分为 82.6%。

4.7 数据可视化

(1) 使用 Flask 结合 Bootstrap 框架实现前端页面
(2) 使用 Echarts 图标库来对数据提供直观、可交互数据可视化图表

五、 运行结果/效果



(1) 图 1：以饼状图的形式对恶意流量和业务流量进行显示，展示恶意流量和业务流量的分布。
(2) 图 2：以折线图的形式展现各流量的时间序列分布图，可以直观的看出业务流量、网络攻击流量、恶意软件流量在不同时间的动态变化趋势。
(3) 图 3：以条形图的形式统计单日内业务流量、网络攻击流量、恶意流量的数量分布。
(4) 图 4：以条形图的形式展现所有流量总和随时间变化的趋势。

六、 创新与特色

(1) 采用 Python 的 scapy 库抓取 CyberFlood 产生的流量，实现构造数据包、

发送数据包、分析数据包的功能，更方便地进行网络编程。

(2) 特征提取方面,后利用文本处理中常用的技术“单词向量空间模型”(word vector space model)由字符型的 URL 构建数值特征，充分挖掘文本潜在语义；特征选择方面，使用梯度提升树模型自动地获取特征的重要性，通过尝试不同的重要性阈值，可以有效地进行特征的筛选。

(3) 采用 XGBoost 梯度提升模型，对稀疏矩阵进行加速处理，并大大降低模型的偏差；引入模型融合与集成学习，训练多个基分类器以降低模型的方差；通过 Stacking 方式对基分类器进行线性组合，提高模型预测的准确率。