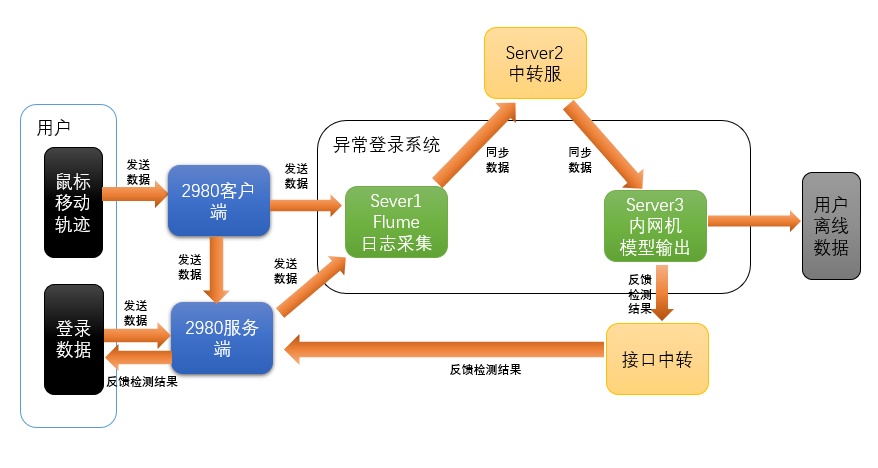
登录异常检测整体进展情况

# 一．登录异常检测系统整体结构

## 1.1系统结构

整个用户登录异常检测系统的结构如下图所示。异常登录系统利用Flume作为日志采集Server1分别采集用户的鼠标移动轨迹的数据（通过2980客户端接收）和用户的登录数据（通过2980的服务端接收），通过Server2中转服同步数据到内网机Server3，并在内网机搭建模型，将接收到的数据通过模型输出反馈的检测结果到接口中转，并由其发送检测结果数据到2980的服务端，最终由2980的服务端发送异常检测结果反馈给用户。



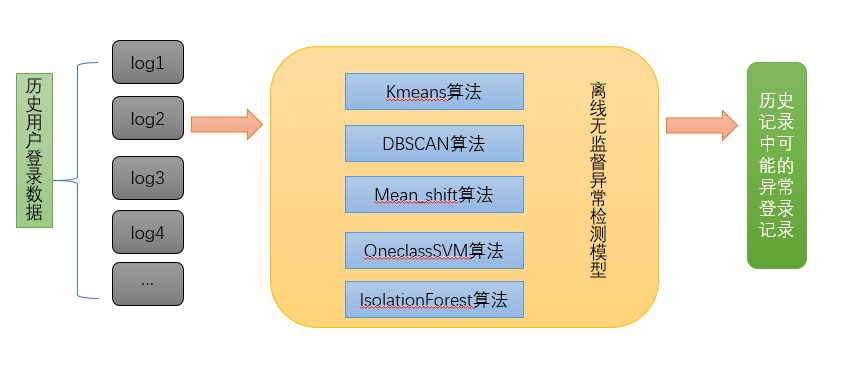
图一：异常检测系统结构图

用户数据一共分为两部分，具体如下表所示。

|  |  |
| --- | --- |
| 用户数据 |  |
| 用户的登录数据（服务端数据） | **用户的鼠标移动轨迹（客户端数据）** |
| 用户ID | 鼠标移动轨迹 |
| 登录失败类型编号 | 鼠标点击 |
| 登录是否成功 | 屏幕宽度 |
| 登录类型 | 屏幕高度 |
|  | 用户代理 |
|  | 鼠标移动开始时间 |
|  | 鼠标移动结束时间 |
|  | 纬度 |
|  | 经度 |
|  | 城市 |
|  | 地区 |

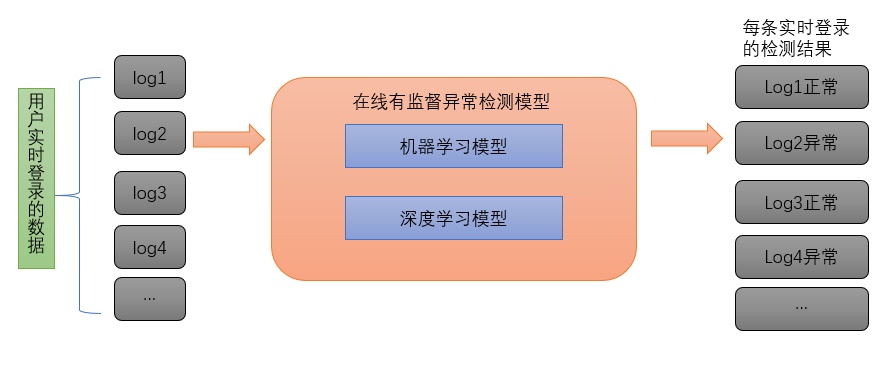
## 1.2 模型结构

异常检测模型一共分为2个模型：离线无监督模型和在线有监督模型。离线模型的具体结构如下图二所示。结合大量用户登录的历史数据和一些聚类算法和异常检测算法构建离线模型，输出在历史记录数据中一些可能的异常登录记录。



图二：离线无监督异常检测模型

而在线学习模型是首先利用历史数据（有标签）结合机器学习算法和深度学习算法搭建的，能够对用户实时登录数据进行实时分类。通过输入在线的用户登录数据，模型判断该条用户登录数据是否是异常。具体流程图如下所示。



图三：在线有监督异常检测模型

# 二. 项目进展

## 2.1 系统搭建情况

### 2.1.1 flume日志采集

（1）在测试服上利用flume+kafka搭建了日志采集系统，实验了从2980客户端和服务端发送外测数据到测试服上，能够实现接收和离线到本地。

（3）编写了2980客户端和服务端的API，完成外测和对接。

（2）编写了正式服flume配置文件。

## 2.2 深度学习模型实验

由于目前我们还没有2980的日志登录数据，就在网上找到了2017年中国高校计算机大赛的初赛数据集（鼠标移动轨迹数据3000条样本） 作为我们实验的数据。

### 2.2.1 特征工程

（1）每条数据中包含n和坐标点（x,y,t），所有数据中n <= 300的，不够300的以0为填充值，并记录当前有用信息的长度sequence\_lenth = n。

（2）对于移动轨迹的每个坐标数据（x,y,t）构建其特征状态量，一共构建了7个特征状态，如下表所示：

|  |  |  |
| --- | --- | --- |
| 特征 | 表示 | 说明 |
| 当前点的速度 | Velocity | 对于首个点而言速度为0，其他坐标点速度等于两个点之间的距离除以两个相邻点的差值。 |
| 当前点与前一个点的距离的差值 | Distance\_diff\_ex | 计算当前点和前一个点的距离，首个点为0 |
| 当前点与其一个点的时间差值 | Time\_diff\_ex | 计算两个相邻点时间t的插值，首个点为0 |
| 当前点的加速度 | Acceleration | 用与前一个点的速度的差值除以时间的差值，首个点为0 |
| 当前点与其一个点的角度的差值 | Theta\_diff\_ex | 利用与前一个点的y方向的差值除以x方向的差值，利用反三角函数求弧度差值 |
| 当前点与目标点的距离 | Distance\_diff\_ta | 计算当前点与最终目标点的距离的差值 |
| 当前点与目标点角度差值 | Theta\_diff\_ta | 计算当前点与最终目标点的角度差值 |

### 2.2.2 模型开发环境

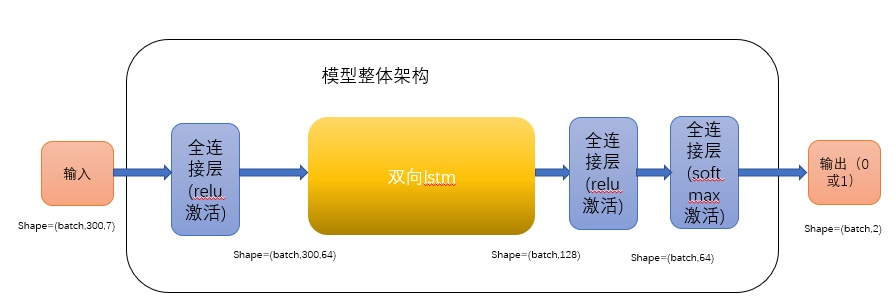
（1）开发语言：python3.5

（2）深度学习框架：Tensorflow1.12.0

（3）编程环境：Windows10

### 2.2.3 深度学习模型搭建过程

（1）模型架构如下图一所示。

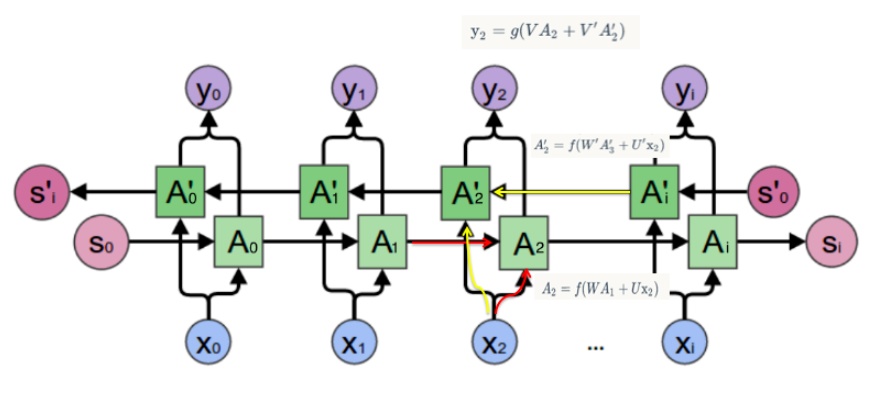


图四：模型整体架构

深度学习模型整体架构如上图二所示，将每条鼠标移动轨迹数据经过特征工程之后输入到模型中，首先经过第一层全连接层，然后经过双向LSTM层抽取时序数据的历史特征，接着利用2层全连接层解决模型异或和非线性问题，最终输出该条轨迹数据是否是正常还是异常。

（2）模型选型

之所有选择双向lstm的原因：RNN和LSTM都只能依据之前时刻的时序信息来预测下一时刻的输出，但在有些问题中，当前时刻的输出不仅和之前的状态有关，还可能和未来的状态有关系。而我们的数据集某一个点的状态量不仅和前一个状态量相关还和后一个状态量相关，因此选用双向lstm层。双向lstm层的细节如下图三所示。



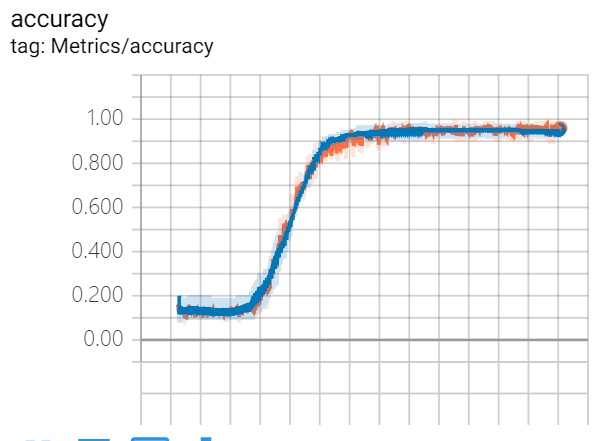
图五：双向LSTM层的具体细节

（4）模型损失的定义

这里考虑到了之前数据集出现的数据不均衡的问题，选用了来自ICCV2017年收录的一篇文章《Focal Loss for Dense Object Detection》，负样本数量太大，占总的loss的大部分，而且多是容易分类的，因此使得模型的优化方向并不是我们所希望的那样，针对样本不均衡，作者提出了一种新的损失函数（focal loss），这个函数可以通过减少易分类样本的权重，使得模型在训练时更专注于难分类的样本，从而达到解决数据不均衡给模型带来分类困难的问题。

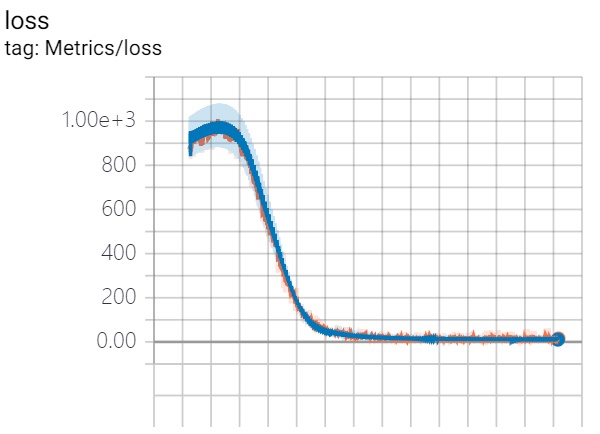
（5）模型在训练和验证集上表现的具体情况（loss和accuracy）

这里截取的图片是由模型在训练的时候由Tensorboard生成的曲线图。具体如下图四所示。



图六：模型在训练迭代中训练集和验证集上的准确率

在图一中，横坐标表示是模型迭代的步数，纵坐标表示的是模型的准确率，蓝色的线表示模型在验证集上的表现，红色表示模型在训练集上的表现。我们可以看到模型训练step=2.0k的时候，模型在训练集上已经表现为90%的准确性，验证集上的表现为89%，当step从3.5k到5.0k之间的时候，模型在训练集和验证集上的准确性趋于稳定，当模型step=5.0k的时候，模型的准确率在训练集上表现为96.3%，在验证集上表现为94%。说明模型在高精准度的情况下也不存在过拟合的现象，模型的泛化能力（算法对新鲜样本的适应能力）强。



图七：模型在训练迭代中训练集和验证集上的损失情况

图五表示的是模型在训练迭代中，随着迭代步数的增加，模型在训练集上从一开始训练和验证集上表现的1000的损失在迭代到1000步的时候，模型的损失陡降到了47左右，并且在稳定下来之后，最终模型在了训练集上loss=10，测试集上loss=12。

### 2.2.3 总结

本次模型的实验是基于2017年中国高校计算机大赛的初赛数据集（鼠标移动轨迹数据3000条样本）的，利用双向LSTM搭建深度学习模型，通过增加权重因子解决样本不均衡问题，并且在多次调参之后，决定了模型超参的大小，从而最终使得模型在迭代5k步之后，训练集上准确性的表现稳定在96.3%左右，验证集上准确性的表现稳定在94%，模型的泛化能力好，精确性高。

## 2.3 Xgboost模型实验

### 2.3.1 xgboost简介

xgboost是由CART树组成的tree ensemble

### 2.3.2 数据集

来自腾讯大数据挑战赛2017提供的3000条有标注的鼠标轨迹数据。

数据格式如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 解释 |
| a1 | bigint | 编号id |
| a2 | string | 鼠标移动轨迹(x,y,t) |
| a3 | string | 目标坐标(x,y) |
| label | string | 类别标签：1-正常轨迹，0-机器轨迹 |

其中前2600条为正样本，后400条为负样本。

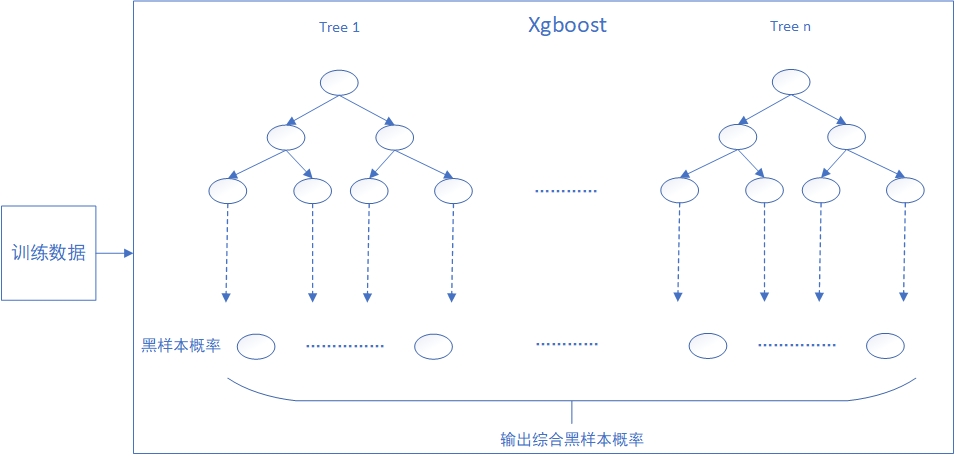
### 2.3.3 数据处理

提取特征供xgboost训练，总共使用了22个特征。

|  |  |  |
| --- | --- | --- |
| 特征编号 | 特征 | 解释 |
| 0 | first\_data\_x | x初始值 |
| 1 | first\_speed\_x | x方向初始速度 |
| 2 | first\_data\_y | y初始值 |
| 3 | first\_delt\_t | 时间差分初始值 |
| 4 | speed\_xstd\_laststd | x方向最后9个点速度标准差 |
| 5 | X\_max | x最大值 |
| 6 | get\_target | x最大值与目标值差 |
| 7 | dxspeed\_mean | x方向速度差分平均值 |
| 8 | dxspeed\_std | x方向速度差分标准差 |
| 9 | data\_x\_return | 是否有回勾 |
| 10 | data\_y\_unique\_value\_stdmean | y坐标唯一值的标准差比平均值 |
| 11 | time\_delta\_min | 时间差分最小值 |
| 12 | distance\_deltas\_max | 距离差分最大值 |
| 13 | median\_speed | x方向速度中位数 |
| 14 | y\_min | y最小值 |
| 15 | x\_min | x最小值 |
| 16 | xs\_delta\_var | x差分方差 |
| 17 | xs\_delta\_max | x差分最大值 |
| 18 | dxacc\_mean | x方向加速度平均值 |
| 19 | dxacc\_std | x方向加速度标准差 |
| 20 | xt\_angles\_max | xt夹角最大值 |
| 21 | xy\_angles\_kurtosis | xy夹角峰值 |

### 2.3.4 xgboost模型结构

模型结构如图八所示：



图八：xgboost结构示意图

模型输入经提取特征的训练数据，输出为黑样本的概率

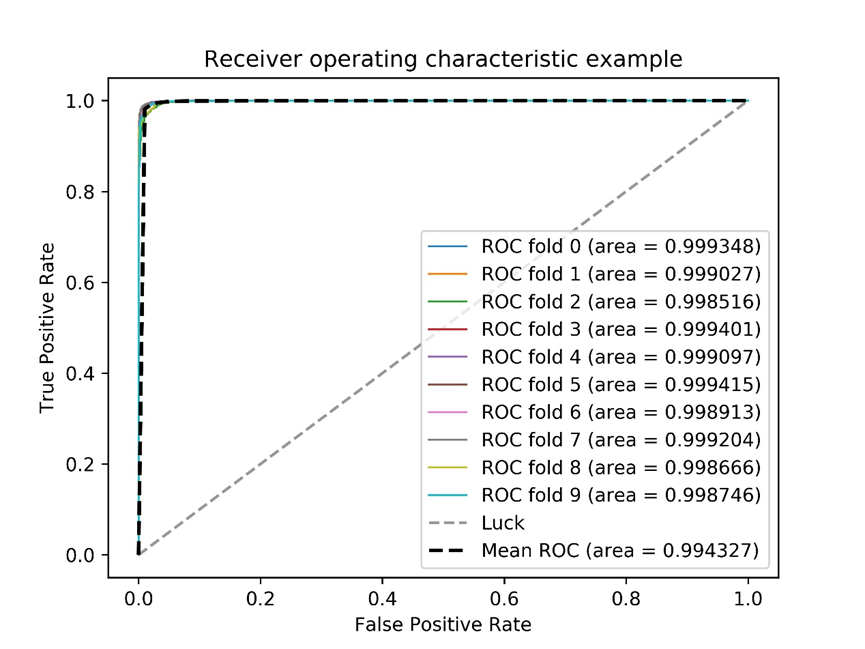
### 2.3.5 模型评估

对3000条训练数据，10次交叉验证（训练集：测试集=2：8），precision与recall如下表：

|  |  |  |
| --- | --- | --- |
| 交叉验证轮次 | precision | recall |
| 1 | 0.97125 | 0.97580 |
| 2 | 0.96053 | 0.98335 |
| 3 | 0.94239 | 0.98160 |
| 4 | 0.98556 | 0.95940 |
| 5 | 0.97836 | 0.97534 |
| 6 | 0.97544 | 0.96801 |
| 7 | 0.96380 | 0.97222 |
| 8 | 0.97108 | 0.97702 |
| 9 | 0.95611 | 0.97288 |
| 10 | 0.96428 | 0.98026 |

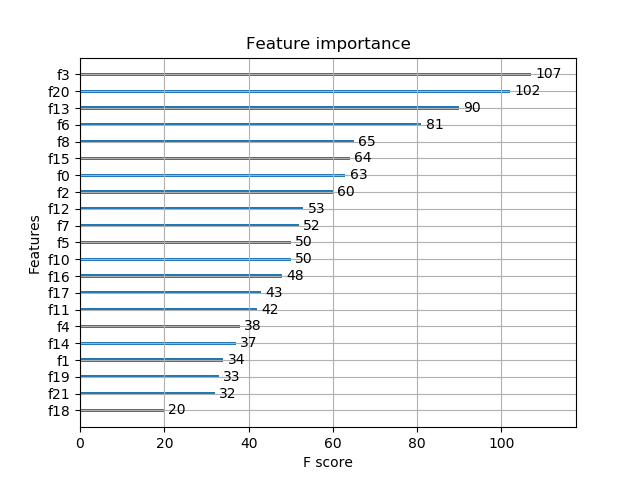
平均precision 0.97459，平均recall 0.9668。

ROC曲线如图七所示：



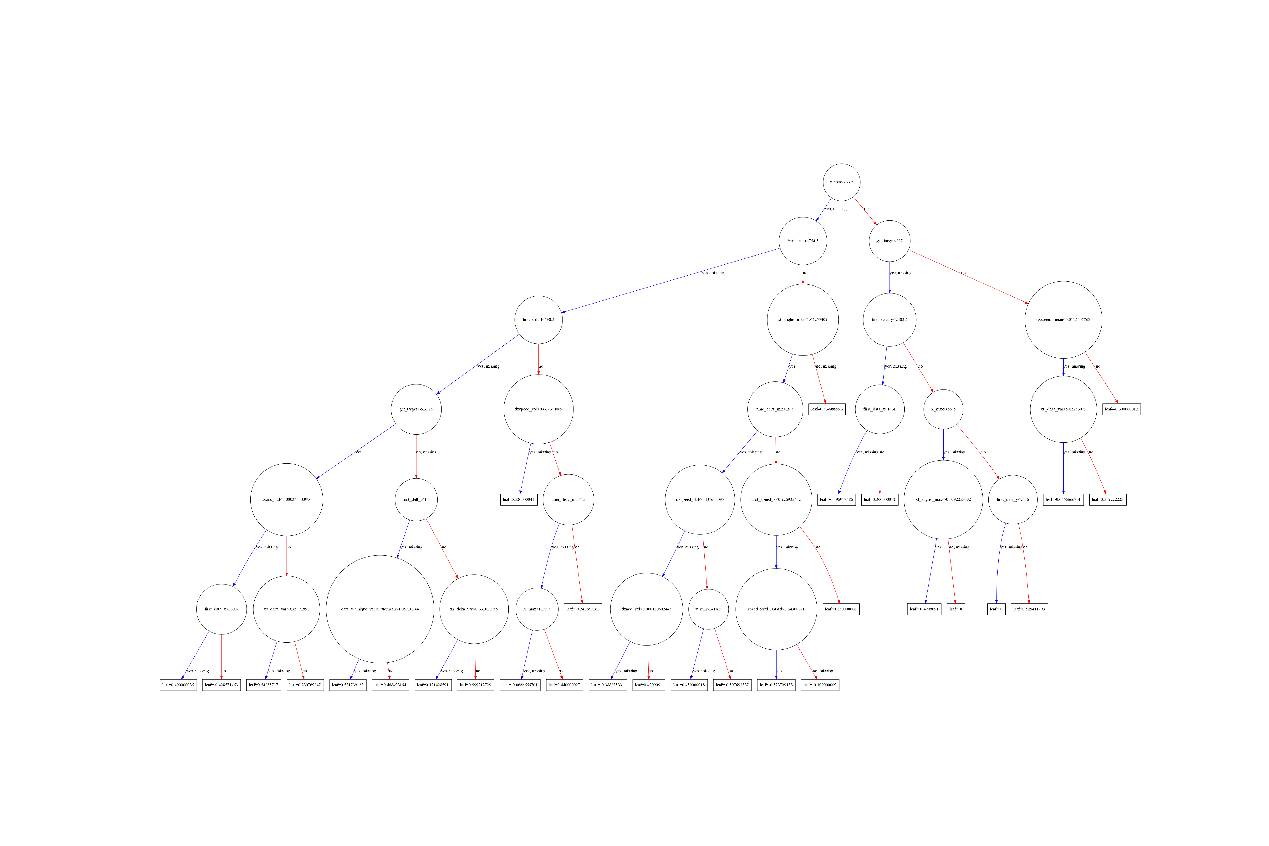
图九：10次交叉验证ROC曲线图

Xgboost特征重要性如图十所示：



图十：特征重要性

Xgboost树图如图十一所示：



图十一：Xgboost树图

### 2.3.6 总结

由于数据极度匮乏，只有3000条有标注数据且缺乏测试集，故重点在于提高模型的泛化能力，因此在划分数据集的时候将训练集与验证集比例设为2：8，并不同于一般监督学习验证集占比0.3左右。同样由于数据匮乏，为了提高对于大量的真实异常情况的覆盖能力，在本次xgboost模型的评估中，判黑阈值为0.35。最终10次交叉验证结果平均precision 0.97459，平均recall 0.9668，可以认为模型有较强的泛化能力，判别准确率高，覆盖广。

# 三．问题分析

## 3.1 数据问题

### 3.1.1 用户登录的真实数据没有标签

目前测试服上外测收集到的数据量极少，并且是2980那边模拟登录的数据，而这些历史数据是没有标签的，我们无法知道哪些登录数据是正常，哪些登录数据是异常的。也就是说模型无法通过有监督的方式进行在线学习。

而本次模型（在线有监督学习）的实验是在有标签的基础之上进行实验的，因此这是一个很大的问题。当然我们可以利用这些没有标签的数据通过无监督离线的方式找出样本中的离群点（异常点），但是我们想要进行在线实时给出模型的检测结果，必须要标签才能进行有监督的学习，完成实时检测的功能。

### 3.1.2 模型实验的数据和用户登录真实数据存在差异

系统从2980客户端收集到的数据中的鼠标移动轨迹和之前建模实验的数据一致，但是缺少最终目标点（target\_point）的数据，这样会让构建特征工程的时候少了2个关键维度。同时系统还收集到了其他的数据（比如用户的id，登录是否成功，登录类型，ip地址，登录失败类型的编号等等），而这些数据都是在之前建模实验的时候没有用到，这些信息和维度我们都没有实验过，因此模型的适用性在没有实验真实数据前还有待商榷。

## 3.2 异常检测系统问题

### 3.2.1 异常检测系统的实时问题

在图一中，由于系统通过flume采集到的数据是离线在外网服务器的本地的，离线日志定时打包成压缩文件，然后通过中转服将数据同步到内网机中，模型才能收到数据。而这段过程中耗时很长，无法做到实时反馈结果。

### 3.2.2 异常检测系统的实时处理数据效率问题

目前还没有找到合适的方式能力实时处理来自于2980客户端和服务端各自的数据，由于数据输入到在线学习模型之前，需要对数据做预处理才能够放进模型，得到检测结果，因此大批量处理数据和还要保证其处理效率这是一个问题。

# 四．待办事项

## 4.1 搭建正式服，接收用户登录数据

（1）目前测试过测试服的日志采集功能，还没有做压力测试。

（2）还未搭建正式服，没有针对正式服做压力测试。

（3）目前有模拟用户登录的外测数据，还没有用户的真实登录数据。

## 4.2 搭建异常检测模型

### 4.2.1 搭建离线异常检测模型

在没有得到真正的数据之前，我们无法利用这些历史数据进行分析和拟合模型（因为不同的样本不同的分布可能需要不同的聚类算法去拟合，因此我们无法利用网上的数据进行实验，这毫无意义），通过其各类特征的分布找到样本中的离群点（异常点），给出每个样本中是异常点的可能性。

我们需要搭建离线的异常检测模型是需要等到正式服收集到一定量的数据之后，我们才能对这些历史数据做清洗和预处理，从而搭建特征工程，利用无监督聚类算法构建离线的异常检测模型。通过该离线模型能输出历史登录数据中的异常点（离群点），给出历史数据中存在的可能是异常的登录记录。

### 4.2.2 搭建在线异常检测模型

在我们能够通过离线的异常检测模型输出历史数据中可能的异常点之后，还需要人工经验去判断这些可能的异常点是否是真的异常，从而能够拿到一定数量的带有异常和正常标签的数据，我们可以利用这些带由标签的数据来搭建在线有监督模型（利用之前用网上数据搭建的模型进行实验）。

## 4.3 测试系统中的数据流通性和实时性

（1）在搭建好异常检测系统之后，需要对数据在异常检测系统和其他服务器之间的流通性进行测试。

（2）在搭建好在线学习模型之后，对消息的实时性进行测试。