|  |
| --- |
| **机器学习** |
| **博世生产线性能** |
| 主 研 人：李琦  参 研 人：李琦 |
| 审 核 人：  方 向：工业算法案例研究  版 本 号：A |
|  |

**版本更新**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **更新描述** | **作者** |
| 2018/10/31 | A | 初稿 | 李琦 |

**选题表格**

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 竞赛名 | 竞赛背景描述（50字以内） | 类型（分类/回归） |
| 2019/02/18 | 博世生产线性能  （kaggle） | 博世公司为确保生产配方的质量，在其装配线的每一步都记录数据。竞赛要求通过对这些大量数据进行分析，来预测内部故障。 | 分类 |

目录

[1. 背景描述](#_Toc22191_WPSOffice_Level1) [4](#_Toc22191_WPSOffice_Level1)

[1.1 竞赛赛题描述](#_Toc32618_WPSOffice_Level2) [4](#_Toc32618_WPSOffice_Level2)

[1.2 评估指标描述](#_Toc15613_WPSOffice_Level2) [4](#_Toc15613_WPSOffice_Level2)

[2. 数据来源及描述性统计分析](#_Toc32618_WPSOffice_Level1) [5](#_Toc32618_WPSOffice_Level1)

[2.1 大赛数据来源](#_Toc9038_WPSOffice_Level2) [5](#_Toc9038_WPSOffice_Level2)

[2.2 数据的描述性统计](#_Toc2474_WPSOffice_Level2) [5](#_Toc2474_WPSOffice_Level2)

[2.2.1 数据基本情况描述：](#_Toc32618_WPSOffice_Level3) [5](#_Toc32618_WPSOffice_Level3)

[2.2.2 数据字段介绍：](#_Toc15613_WPSOffice_Level3) [5](#_Toc15613_WPSOffice_Level3)

[2.2.3 数据描述性统计](#_Toc9038_WPSOffice_Level3) [7](#_Toc9038_WPSOffice_Level3)

[3. 优秀算法思路](#_Toc15613_WPSOffice_Level1) [12](#_Toc15613_WPSOffice_Level1)

[3.1 方案一](#_Toc13077_WPSOffice_Level2) [12](#_Toc13077_WPSOffice_Level2)

[3.1.1 方案一数据预处理及特征工程部分方案](#_Toc2474_WPSOffice_Level3) [13](#_Toc2474_WPSOffice_Level3)

[3.1.2 方案一模型设计、建立部分方案](#_Toc26457_WPSOffice_Level3) [16](#_Toc26457_WPSOffice_Level3)

[3.1.3 方案一结果、排名等](#_Toc17910_WPSOffice_Level3) [16](#_Toc17910_WPSOffice_Level3)

[3.1.4 方案一算法流程图](#_Toc1360_WPSOffice_Level3) [17](#_Toc1360_WPSOffice_Level3)

[3.2 方案二（第三名方案） 18](#_Toc7344_WPSOffice_Level2)

[3.2.1 方案二数据预处理及特征工程部分方案](#_Toc12066_WPSOffice_Level3) [18](#_Toc12066_WPSOffice_Level3)

[3.2.2 方案二模型设计、建立部分方案](#_Toc19896_WPSOffice_Level3) [20](#_Toc19896_WPSOffice_Level3)

[3.2.3 方案二结果、排名等](#_Toc25281_WPSOffice_Level3) [22](#_Toc25281_WPSOffice_Level3)

[3.2.4 方案二算法流程图](#_Toc17014_WPSOffice_Level3) [22](#_Toc17014_WPSOffice_Level3)

[3.3 方案三（第八名方案）](#_Toc11592_WPSOffice_Level2) [23](#_Toc11592_WPSOffice_Level2)

[3.3.1 方案三数据预处理及特征工程部分方案](#_Toc13144_WPSOffice_Level3) [23](#_Toc13144_WPSOffice_Level3)

[3.3.2 方案三模型设计、建立部分方案](#_Toc11108_WPSOffice_Level3) [26](#_Toc11108_WPSOffice_Level3)

[3.2.3 方案三结果、排名等 2](#_Toc13077_WPSOffice_Level3)8

[3.2.4 方案三算法流程图 2](#_Toc7344_WPSOffice_Level3)9

[4. 算法比较](#_Toc9038_WPSOffice_Level1) [30](#_Toc9038_WPSOffice_Level1)

[表4-1 算法比较](#_Toc23114_WPSOffice_Level2) [30](#_Toc23114_WPSOffice_Level2)

[5. 总结与展望](#_Toc2474_WPSOffice_Level1) [31](#_Toc2474_WPSOffice_Level1)

[5.1 总结](#_Toc26871_WPSOffice_Level2) [31](#_Toc26871_WPSOffice_Level2)

[5.2 建模思路](#_Toc12195_WPSOffice_Level2) [31](#_Toc12195_WPSOffice_Level2)

1. 背景描述

一个好的巧克力蛋奶酥是美味和精致的。但是，准备工作是一项挑战。当你从炉子里拿出一个令人失望的甜点时，你本能地回溯你的步骤，以确定你出错的地方。

[博世](http://www.bosch.com/en/com/home/index.php" \t "https://www.kaggle.com/c/_blank)是世界领先的制造公司之一，必须确保其生产的先进机械部件的配方具有最高的质量和安全标准。做法的一部分是在制造过程中逐步监控其各个部分，并应用高级分析来改进这些制造流程。这将使博世能够以最低成本为最终用户提供优质产品。

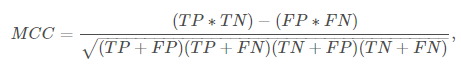
**1.1 竞赛赛题描述**

博世在其每条装配线的每一步都记录数据，再应用高级分析来改进这些制造流程。然而，数据的复杂性和生产线的复杂性给当前方法带来了问题。

在本次比赛中，挑战者需使用数千次测量和测试的结果来预测内部故障，并对装配线上的每个部件进行测试，预测哪些部件将无法通过质量检测。

**1.2 评估指标描述**

参赛作品将根据马修斯相关系数(MCC)对预测结果和观测结果进行评估 。



其中TP是真正例的数量，TN是真负例的数量，FP是假正例的数量，FN是假负例的数量。

1. 数据来源及描述性统计分析

**2.1 大赛数据来源**

大赛数据提供方博世公司，为大赛提供真实生产数据，提供工艺专家的专业指导，从软硬件环境诸多方面提供大赛支撑。

[data\bosch-production-line-performance.zip](data/bosch-production-line-performance.zip)

**2.2 数据的描述性统计**

**2.2.1 数据基本情况描述：**

本次比赛的数据代表了零件在博世生产线上的测量结果。每个部分都有唯一的ID。目标是预测哪些部件将无法通过质量检测（由'response'= 1表示）。

数据集包含大量匿名特征。功能根据约定命名，该约定告诉您生产线，生产线上的工作站和功能编号。例如，L3\_S36\_F3939是在第3行，站36处测量的特征，并且是特征号3939。

由于数据集的太大，我们将文件按其包含的特征类型分开：数字，分类，最后是具有日期特征的文件。日期特征提供了每次测量时间的时间点。每个日期列以与前一个功能号对应的数字结尾。例如，L0\_S0\_D1的值是L0\_S0\_F0的取值时间。

除了作为Kaggle有史以来最大的数据集之一（就特征数量而言）之外，本次比赛的基本事实也非常不平衡。总之，这两个属性预计会使这成为一个具有挑战性的问题。

**2.2.2 数据字段介绍：**

**train\_numeric.csv** - 数字特征的训练集（包含'Response'变量）

**表2-1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **var1** | **ID** | 离散 | 0% |
| **var2~var969** | **匿名特征** | 连续 |  |
| **var970** | **响应** | 离散 | 0% |

**test\_numeric.csv** - 数字特征的测试集（不包含'Response'变量）

**表2-2**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **var1** | **ID** | 离散 | 0% |
| **var2~var969** | **匿名特征** | 连续 |  |

**train\_categorical.csv** - 分类特征的训练集

**表2-3**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **var1** | **ID** | 离散 | 0% |
| **var2~var** | **匿名特征** | 连续 |  |
| **var** | **响应** | 离散 | 0% |

**test\_categorical.csv** - 分类特征的测试集

**表2-4**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **var1** | **ID** | 离散 | 0% |
| **var2~var** | **匿名特征** | 连续 |  |

**train\_date.csv** - 日期特征的训练集

**表2-5**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **var1** | **ID** | 离散 | 0% |
| **var2~var** | **匿名特征** | 连续 | 80% |
| **var3** | **响应** | 离散 | 0% |

**test\_date.csv** - 日期功能的测试集

**表2-6**

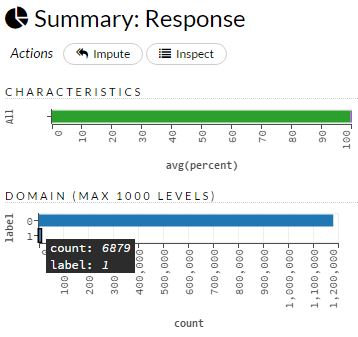
|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **var1** | **ID** | 离散 | 0% |
| **var2~var** | **匿名特征** | 连续 |  |

**2.2.3 数据描述性统计**

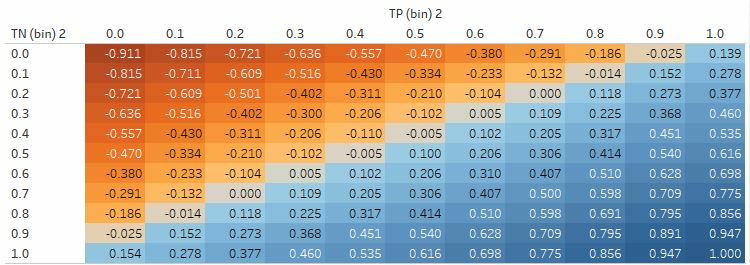
### 快速分析标签：

### 我们正在使用1：172的不平衡比率（1个正数，172个负数）：如果仅预测负面情况，则准确度为99.41％

1. 1183747行
2. 1176868个负面标签
3. 6879个正面标签



度量的演变如下(真正vs真负- FP/FN):

*<https://public.tableau.com/views/KaggleMCCMetric/Binned0_01?:embed=n&:display_count=no&:toolbar=no>*

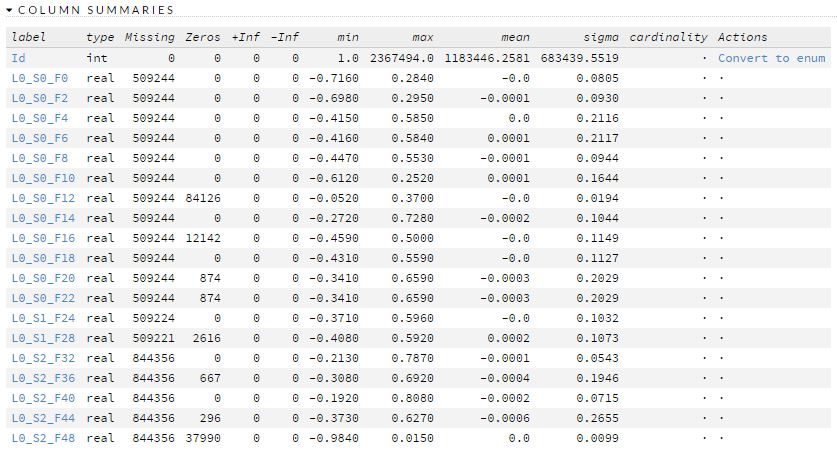
### train\_numeric.csv

1. 大小：[1183747 x 969]（969个特征，1183747观测）
2. 缺失的值似乎没有随机丢失。有关详细信息，请参阅

<https://www.kaggle.com/blobs/download/forum-message-attachment-files/4690/Missing_Train_Numeric.htm>

1. 稀疏的
2. 包含我们需要预测的标签

潜在峰值：

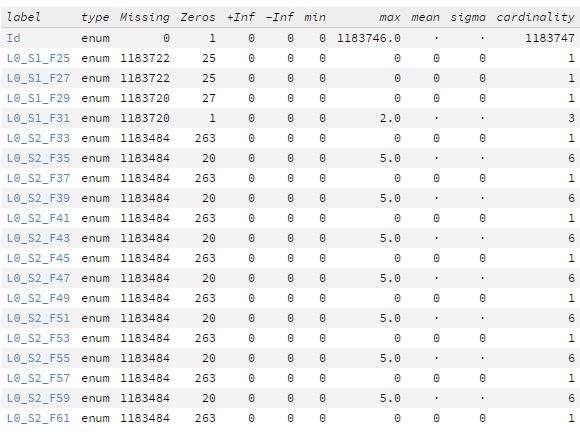


### train\_categorical.csv

1. 矩阵大小：[1183747 x 2141]（2141个特征，1183747观测）
2. 缺失的值似乎没有随机丢失。有关详细信息，请参阅<https://www.kaggle.com/blobs/download/forum-message-attachment-files/4691/Missing_Train_Categorical.htm>

（加载数据集时似乎存在导入错误，需要检查但这些缺失值计数根本不可靠）。

1. 许多列有1183747个缺失值或接近它（这意味着可以删除很多）。
2. 非常稀疏
3. 很多只有0
4. L1-S24-F7XX到L1-S24-F9XX（用数字代替XX）似乎连接（相同的零）
5. 很多特征没有内容（只是稀疏，见下文）



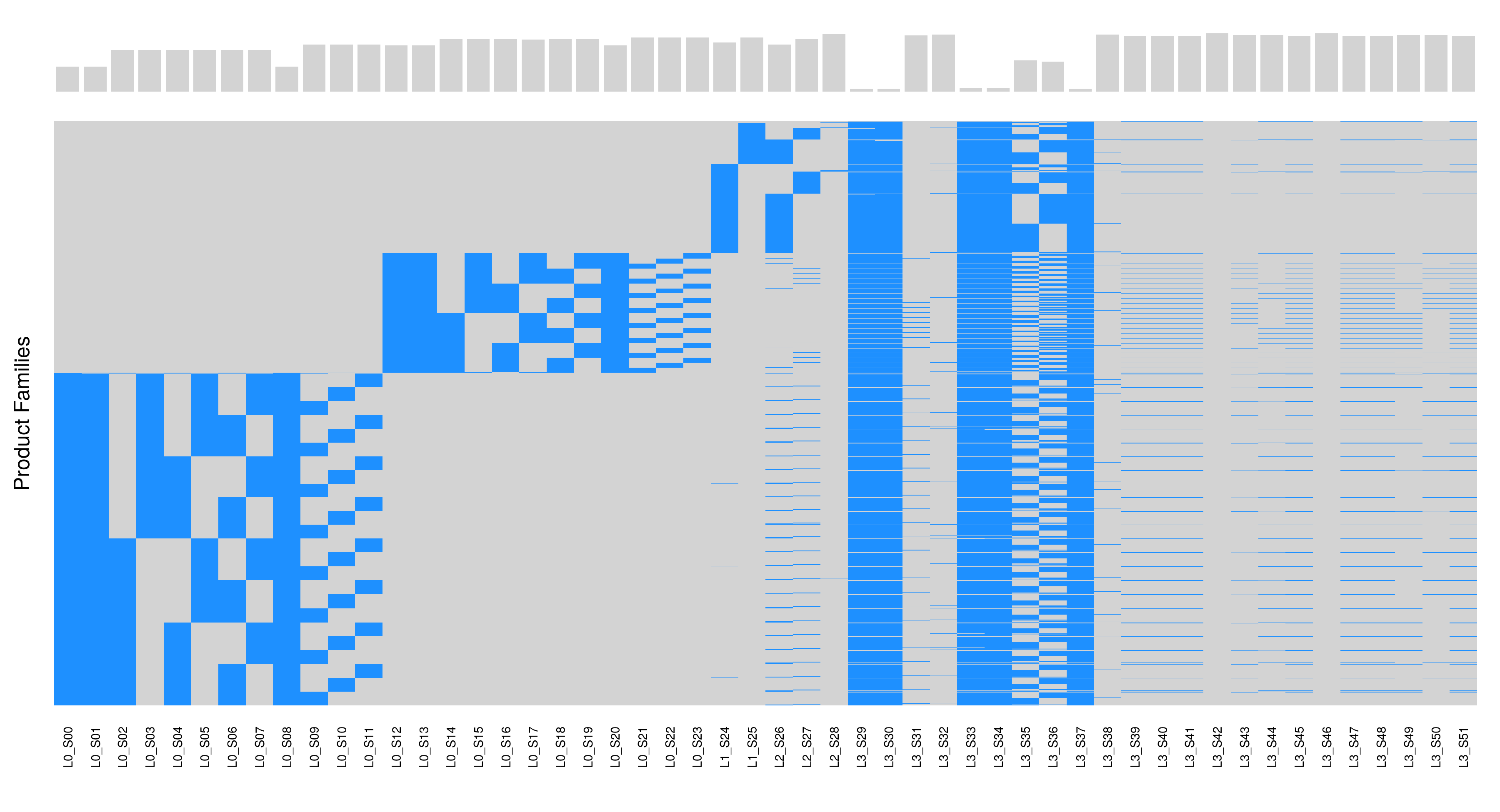
### train\_data.csv

1. 大小：[1183747 x 1157]（1157个特征，1183747个观测）
2. 缺失的值似乎没有随机丢失。
3. 稀疏的
4. 日期已经是数字，似乎按日“标准化”？



**生产地图：**

这是一张“生产地图”，了解零件如何通过工厂。每个部件接触哪些机器，哪个机器跳过等等。它给出了具有类似流路的部件的粗略分组。



以下是关于查看零件流的逻辑和代码的更多解释：

1. 目标是跟踪52个工位的每个部分，并了解制造流程。很多时候，质量是零件类型和使用的特定机器的函数。
2. 测量仅在二元意义上用于此处。如果零件在工作站（分类或数字特征）上有任何测量值，则该工作站对该零件进行操作。否则该部分没有通过该站。
3. 对具有相似制造模式的零件进行分组将有助于识别零件系列，并可能阐明工作站的性质。
4. R的VIM包装完成了绘图的所有繁重工作。通过对每个部分的NA进行计数和排序，它提供了很好的计数和非NA的排序。

关于图纸的一些注意事项：

1. 显然，x轴代表每个站。y轴表示一堆部件，按相似的NA模式排序。
2. 块的高度表示组的相对频率。
3. 顶部的条形显示每个工作站的NAs百分比。

我们可以从图片中看到一些东西：

1. 有3个基本的部分组。经过L0（S0-S11）的，经过L0（S12-23）的，和经过L1和L2的。所有部件都通过L3。
2. L0（S0-S11）和L0（S12-S23）具有相似的图案，因此它们可以是具有不同特性的类似部件类型。例如，常规尺寸的零件可能会通过一组机器，而较少数量的大型零件会通过另一组。
3. 每个主要群体内都有互补站。例如，S09-S11。这些可能是3个拥有相同设备的工作站，或者他们可以使用不同的设备来处理部件选项。鉴于量相等，我认为在这种情况下它们更可能是同一事物的3个实例。
4. 一切都贯穿L3。大多数部件经过相同的5个站，然后有许多低容量的站。这可能是终点线，几乎所有东西都被去毛刺，喷砂，涂漆等等，而少数零件可以获得某种定制效果。或者这可能是一个装配步骤？

代码链接：<https://github.com/JohnM-TX/Kaggle-Bosch>

**3. 优秀算法思路**

**3.1 方案一**

**3.1.1** 方案一数据预处理及特征工程部分方案

**1. 数据探索：**

对于如何处理数以千计的匿名特征，通过对数字特征图，统计数据，站到站转换概率，每个站的相关矩阵等进行数据发现。

由于存储器限制，在开始时需要一些额外的努力来处理较低级别的数据操作。越来越接近原始数据有时会有所帮助。

由于数字特征和日期特征具有.3f / .2f精度，我们将所有内容重新缩放（\* = 1000，\* = 100）为整数。只保留每个站的最短时间。

由于有很多缺失值，运用稀疏矩阵。

**2.泄漏/魔术/数据属性特征：**

发现连续的行具有重复特征和相关联的响应。我们调用由StartStation，StartTime数据块顺序排列的连续ID 。

对于每个数据块，我们用以下特征：记录数量、等级上升、等级下降

我们使用原始数据来根据上一个和下一个记录添加特征：响应、每个站的哈希特征相等、StartTime，EndTime、数字原始特征

**特征处理：**

**时间特征（发现**[0.01时间粒度意味着大概6分钟](https://www.kaggle.com/gaborfodor/bosch-production-line-performance/notebookd19d11e4f2" \t "https://www.kaggle.com/c/bosch-production-line-performance/discussion/_blank)**）**

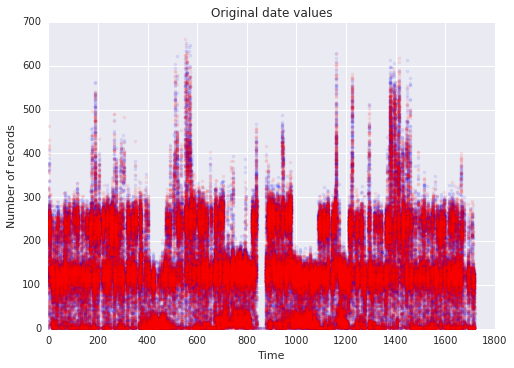
* 很多列(1157)
* 80%+ 缺失值
* 相同的站经常拥有相同的日期值

目标：时间戳在本次比赛中是匿名的。以下是为了了解测试/训练周期有多长。这将允许使用一些直觉进行特征工程。

1. 检查每个站点的最小和最大时间

只是删除了缺失的值。由于站的观测时间几乎总是唯一的，因此Id对可以为每个站点只读取一次，从而节省大量内存。

1. 选择特征



**发现：**

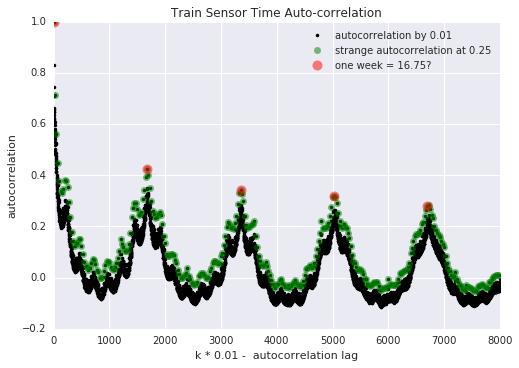
训练和测试集有相同的时间周期

有一个清晰的周期模式

将日期转换为0 - 1718，粒度为0.01

中间有一个缺口

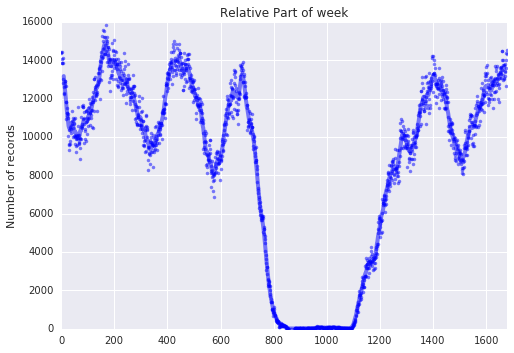
C.自相关：



最大的峰值大约位于1680。

每个星期可以看到7个极大值的天

D.每周总统计：



因此要处理2年的制造数据。这一观察结果没有给出直接的性能提升，但给了足够的直觉来构建大量基于时间的特征。

StartStationTimes  
StartTime, EndTime, Duration  
StationTimeDiff  
Start/End part of week (mod 1680)  
每个站上/下2.5h、24h、168h的记录数

同一时间段的记录数(6分钟)  
MeanTimeDiff since last 1/5/10 failure(s)  
--Mean(EndTime for latest K failures where EndTime < CurrentEndTime) - CurrentEndTime

MeanTimeDiff till next 1/5/10 failure(s)

--Mean(EndTime for the first K failures where EndTime > CurrentEndTime) - CurrentEndTime

## 数字特征

原始数字特征（大多数时候使用原始数字特征或基于xgb特征重要性的简单子集）

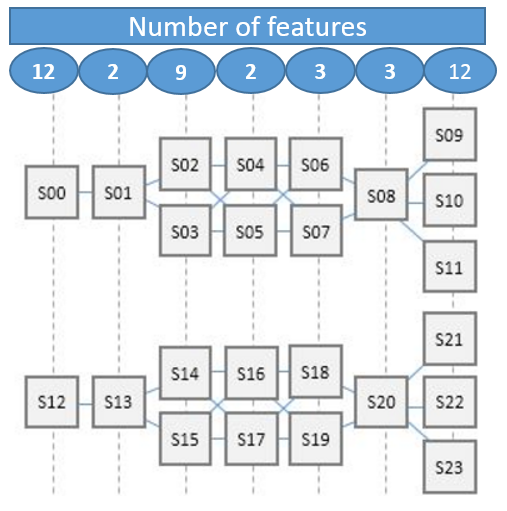
每周的Z度量的特征

每个值的计数编码

特征组合（f1 + - \* f2）

在站S0-S11和S12-S23之间的站转换概率矩阵中看到了有趣的相似性。在Shopfloor可视化中也能看到相同内容。我们还注意到这些站组的特征数量是相同的。

这些站的相关矩阵是相似的，所以我们尝试结合相同阶段的数字特征。(如:L0S0F0 +L0S12F330等)



## station_0_numeric_correlationstation_12_numeric_correlation

**分类特征**

无法从分类特征中挤出太多东西。大部分时间都放弃它们。在[24,25,26,27,28,29,32,44,47]中为每个站点设置了一些具有聚合分类特征的模型。'T1'被1替换，所有其他值被替换为100。

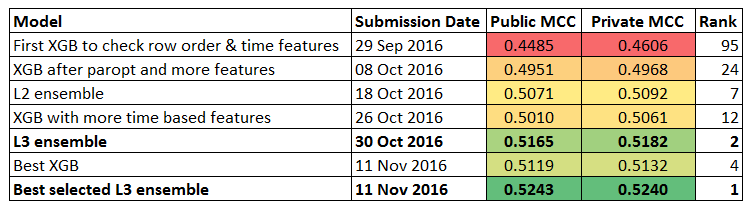
**3.1.2** 方案一模型设计、建立部分方案

**1. 验证**

鉴于不平衡的标签和二元评估标准，在开始时注意到相当高的5倍CV stds（~0.01）。经过几次实验后，决定使用4倍“泄密分层”交叉验证（强制每次折叠都有相同数量的重复）。  
 每个“单一”L1模型在相同的折叠上用3种不同的种子进行训练。最新的集成得分相当稳定（标准<0.004），我们的CV / LB得分通常会提高。

## 2.再平衡

我们保留每一个失败并记录重复。对于剩余的0s，使用50-90％的下采样。它使训练更快，并作为模型平均部分的装袋，帮助后来的集成阶段。

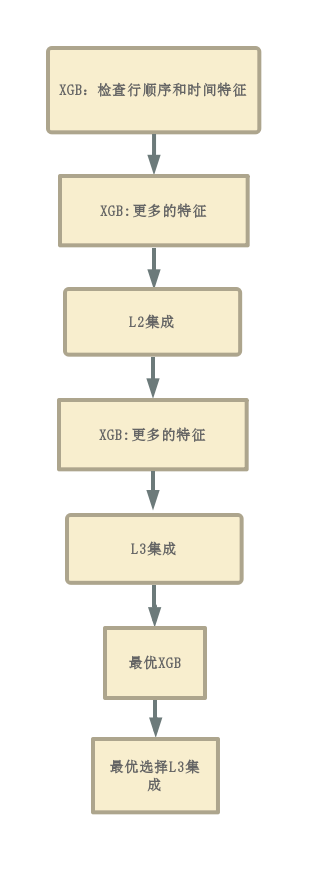


**3.1.3** 方案一结果、排名等

结果：0.52401 排名：1

**3.1.4** 方案一算法流程图

**图3-1**

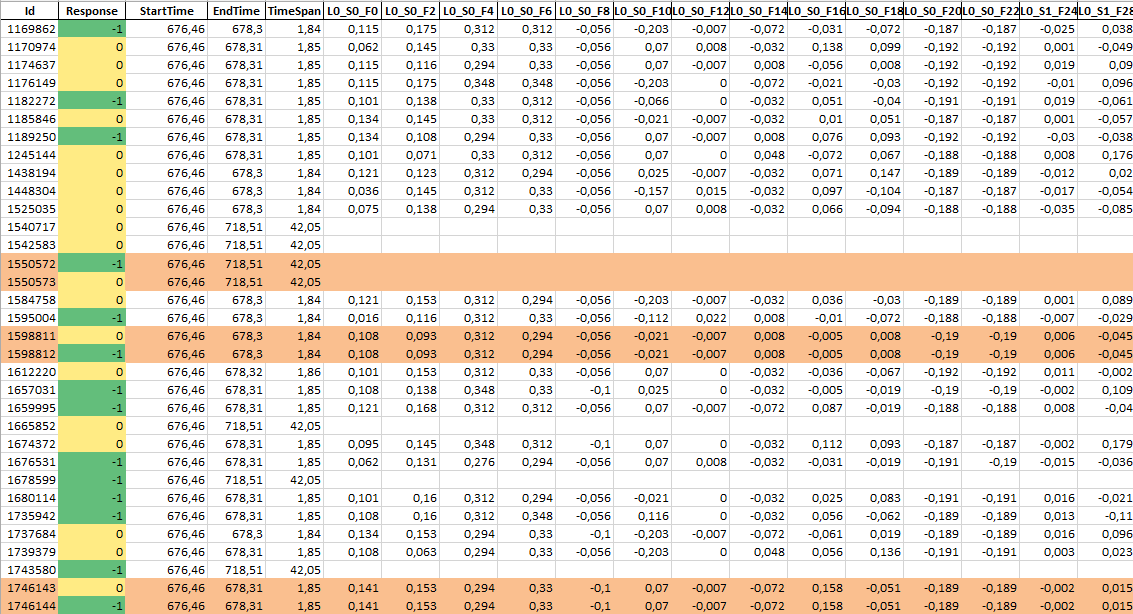
****

**3.2 方案二（第三名方案）**

**3.2.1** 方案二数据预处理及特征工程部分方案

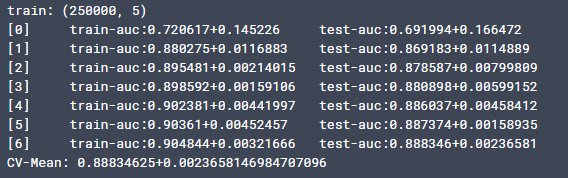
* **泄露：**

发现特征：



神奇的特征：

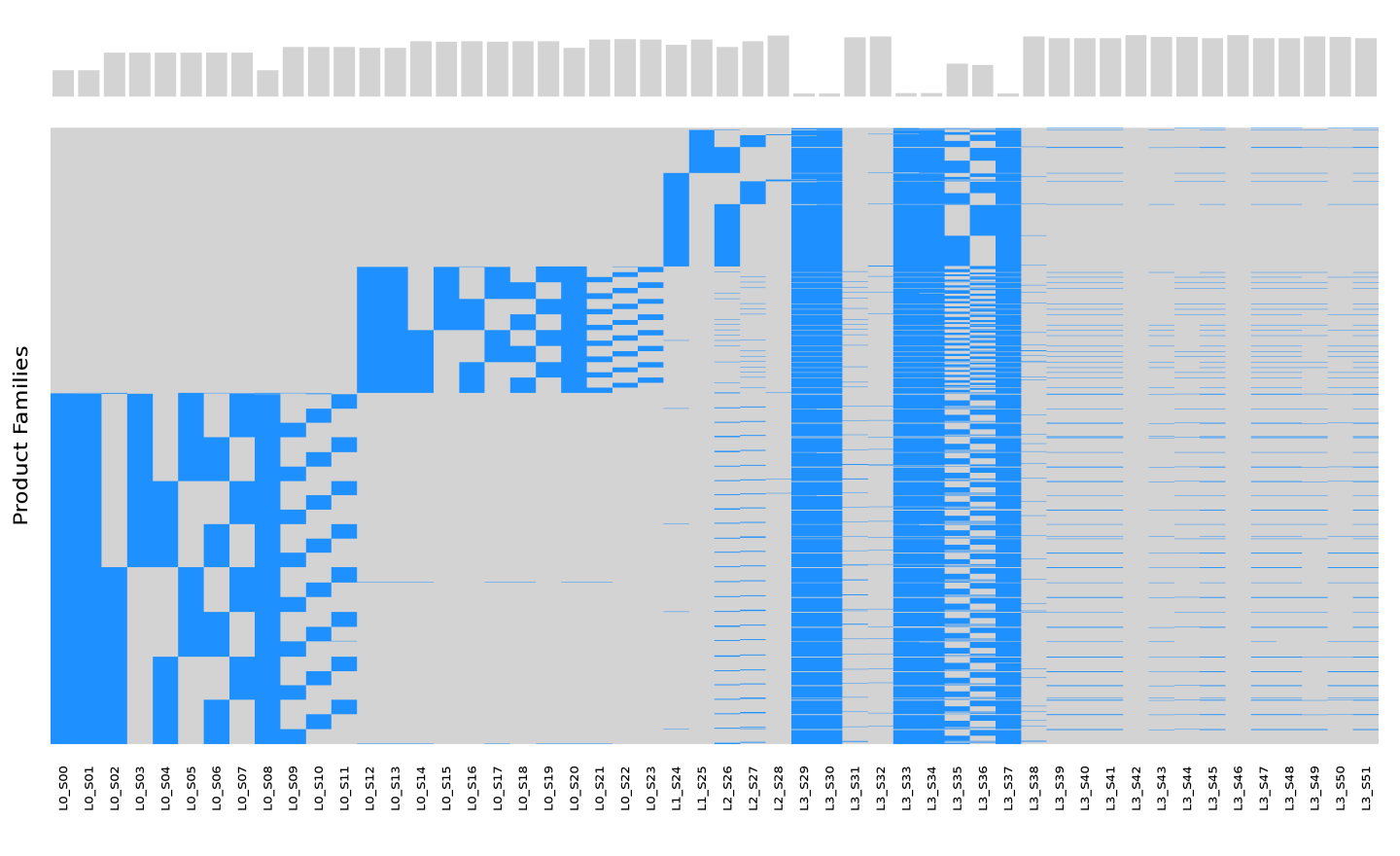




泄漏涉及具有相同数值读数的具有高故障率的顺序组件。

通过对顺序组件使用不同的标准找到了额外的信息。有这么多不同的生产线和工位，两个组件可以在一个工位连续，但随后通过不同的工位进行下一个生产阶段。鉴于此，使用相同的数字读数在每个单独的站识别顺序组件，按日期。我们发现一些站点，例如L3\_S29和L3\_S30，在大多数组件通过这些站点时工作得特别好。这在[可视化制造过程中](https://www.kaggle.com/jpmiller/bosch-production-line-performance/flowpath-viz/code" \t "http://blog.kaggle.com/2016/12/15/bosch-production-line-performance-competition-winners-interview-3rd-place-team-data-property-avengers-darragh-marios-mathias-stanislav/_blank)尤为明显制造过程。在确定了这一点之后，我们可以通过计算一对组件中有多少个工作站具有相同的值，或者通过计算这些数值在整个数据集中出现的次数来构建它。





制造过程可视化

我们还看到了短期和长期的失败率随时间变化的趋势。我们使用基于所有工作站开始和结束日期排序的组件故障的滚动平均值来训练模型。我们使用不同的滚动窗口--5,10,20,100,1000,5000组件 -- 计算，以捕获故障率的长期和短期趋势。重要的是从折叠中计算这些特征以防止过度拟合。下面可以看出[OOF轧制平均值](https://www.kaggle.com/c/bosch-production-line-performance/forums/t/25359/3-place-solution?forumMessageId=144485" \l "post144485" \t "http://blog.kaggle.com/2016/12/15/bosch-production-line-performance-competition-winners-interview-3rd-place-team-data-property-avengers-darragh-marios-mathias-stanislav/_blank)与通常[轧制平均值](https://www.kaggle.com/c/bosch-production-line-performance/forums/t/25359/3-place-solution?forumMessageId=144485" \l "post144485" \t "http://blog.kaggle.com/2016/12/15/bosch-production-line-performance-competition-winners-interview-3rd-place-team-data-property-avengers-darragh-marios-mathias-stanislav/_blank)相比，窗口尺寸为5个组件。



类似地，可以捕获目标的滞后和引导，这是基于树的模型不能很好地捕获开箱即用的东西。

除此之外，还有很多常用特征：

* 编码了一些带有折叠贝叶斯均值的分类列
* 非重复分类列的计数，非重复日期和数字列的计数。
* 通过站编码组件的路径; 组件是否通过相同的站序列。
* 行数NA计数数字，以及每个站的最大/最小值。

总的来说，在一级模型上拥有超过2000种特征。

**3.2.2** 方案二模型设计、建立部分方案

### **验证**

我们使用了5倍交叉验证。

### **建模**

第一级约有160个模型。其中大部分都是XGBoosts，但在本次比赛中，LightGBM已经证明非常好。我们还有Extra Trees Classifiers，Neural Nets，Random Forests和Linear Models。

### **元建模**

在二级使用了一个袋装XGBoost。元模型选择是基于在基础级别使用的相同的5倍交叉验证完成的。模型选择后，共有45个模型。任何其他东西都不起作用。



使用的元建模示意图。

### **最后的集成**

在这场比赛中，应该严格预测0或1 - 所以在选择概率阈值时，接近阈值的行显示出很多随机性 - 有时被预测为0，有时被预测为1.为了减轻这种影响，使用多数投票作为最终选择，有许多不同的离散预测。结果给了一个很好的提升 - 在公共和私人LB上大约0.003。

可以在下面看到25个提交的行方向总和（在y轴上截止），其中有一些信息表明在不同的提交中，总共有多少行被预测为1或0或有时两个类：

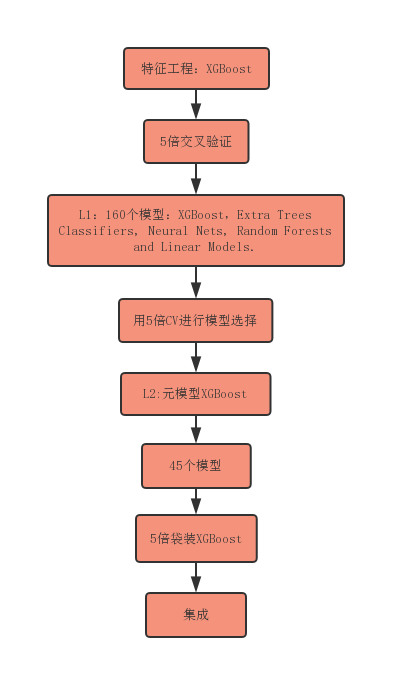


**3.2.3** 方案二结果、排名等

结果：0.51621 排名：3

**3.2.4** 方案二算法流程图

**图3-2**

****

**3.3 方案三（第八名方案）**

**GitHub：<https://github.com/joostgp/kaggle_bosch>**

**3.3.1** 方案三数据预处理及特征工程部分方案

**预处理：**

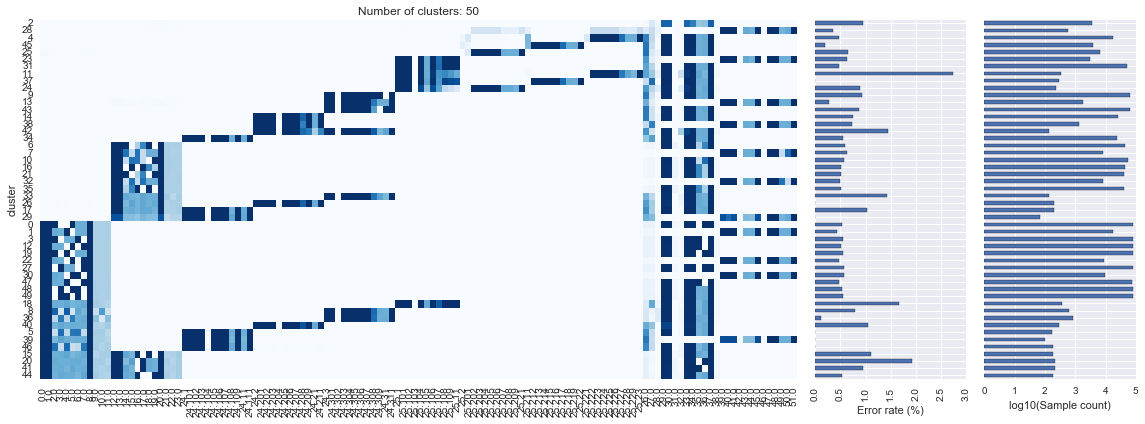
1. 将数据从密集格式转换为稀疏格式。稀疏度平均约为80％，因此可以在更小的内存占用情况下更快地生成功能。



1. 创建一个查找表，帮助合并数字特征，分类特征和时间戳数据集之间的数据。



1. 根据遍历生产线的路径对所有样品进行聚类。[这里](https://github.com/joostgp/kaggle_bosch/blob/master/pre_path_per_sample.ipynb)创建路径，并[在这里](https://github.com/joostgp/kaggle_bosch/blob/master/pre_kmeans_cluster.ipynb)分类合并。以下示例结果。



**特征工程：**

**顺序：**

* 按照tmax,tmin进行排序后，找前后样本的相关性



* 按照数字值进行排序后，找前后样本的相关性



* 按照tmax排序后，前后样本相同的时间戳



* 前后相应的重复

### **路径:**

* [通过路径聚类样本并计算每个样本的数值/时间差（最大 - 最小）特征和聚类均值之间的绝对和相对差异](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_path.ipynb)



* [基于路径的错误率（留出一个方法），但它没有提高分数](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_randomized_loo_time_path.ipynb)



* 出入站点
* [前后站点对于S29-S37（单热编码）](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_source_dest_stations.ipynb)



### **日期时间:**

* 时间戳：每个站，每条生产线，每个合并路径



* [大到小：每个站，每条生产线，每个合并路径](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_date.ipynb)



* [每条生产线的峰度+峰度（非常强，令人惊讶）](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_timestamp.ipynb)



* 按tmax，id排序后的超前/滞后响应率统计
* 按tmax，numeric，id排序后的超前/滞后响应率统计
* [基于时间戳的错误率（保留一个方法），但它没有提高分数](https://github.com/joostgp/kaggle_bosch/blob/master/feat_randomized_loo_time_path.ipynb)
* 基于时间戳标签的密度（过度拟合）

### **Numeric:**

* 使用受监督的决策树尝试使用日期预测数值，每站=>分割点允许定义阈值，按日期组（超过4000个特征）聚集值=>每个站使用xgboost将其缩小120个特征，当添加4000个特征时没有用
* [消除时间趋势后的数字（没有提高分数）](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_numeric_detrended.ipynb)

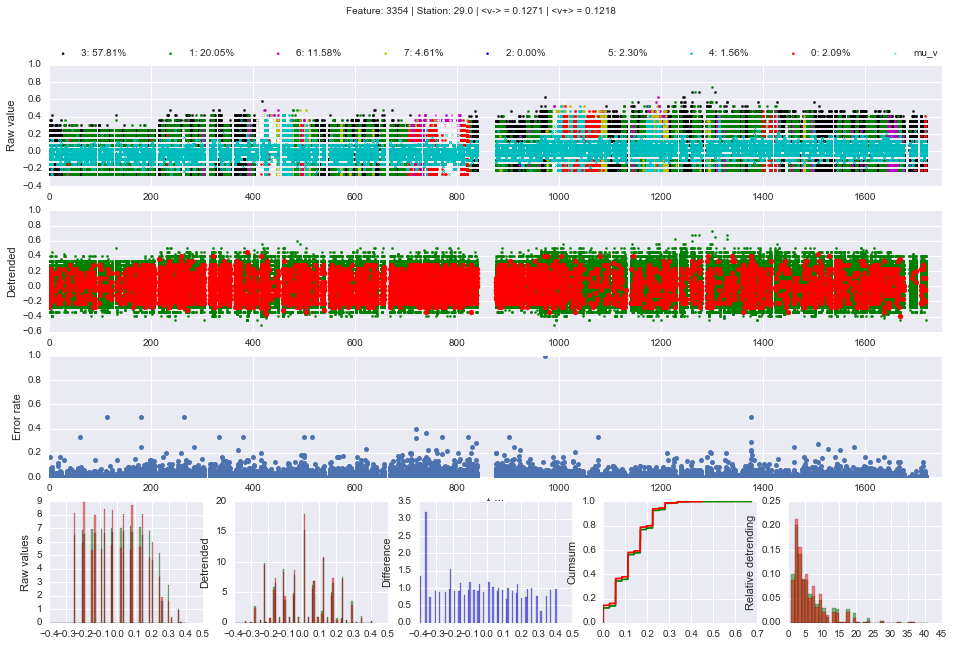


* [偏离站点/线路的平均值](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_numeric.ipynb)



* 原始数字数据（特定选择）

下方用于评估数字特征的仪表板图像。

****

### **分类：**

* [一对特征热编码](https://github.com/joostgp/kaggle_bosch/blob/master/feature_set_categorical.ipynb)增加了一点点



**3.3.2** 方案三模型设计、建立部分方案

### **一层模型:**

* 数据集1 (0.477 gbm): 顺序,原始数字，分类，日期特征 [GBM Model](https://github.com/joostgp/kaggle_bosch/blob/master/model_lgbm.R)



* 数据集 2 (0.482 gbm, 0.477 xgb, 0.473 rf): 顺序, 路径, 原始数字，日期特征
* 数据集 3 (0.479 gbm, 0.473 xgb): 顺序, 路径, 数字，日期，重新定义的分类特征
* 数据集 4 (0.469 xgb, 0.442 rf): [XGBoost model](https://github.com/joostgp/kaggle_bosch/blob/master/model_xgb.R)



* 数据集 5 (0.43 xgb): 魔法特征, 路径, 无监督的最近邻近

### **2级堆栈模型**

给较弱的模型一个更强的权重是更好的：

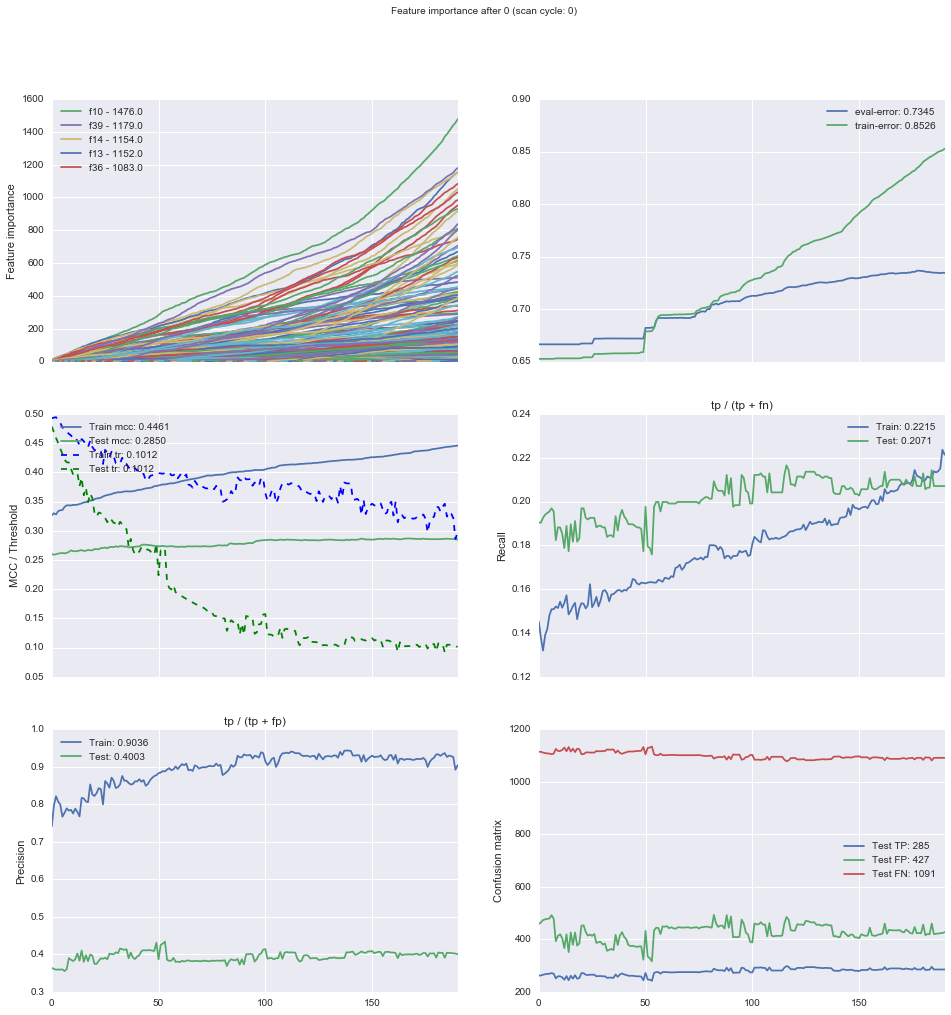
* 30％加权xgboost gbtree（~0.488 CV）



* 70％加权随机森林（~0.485 CV）



典型的训练仪表板：

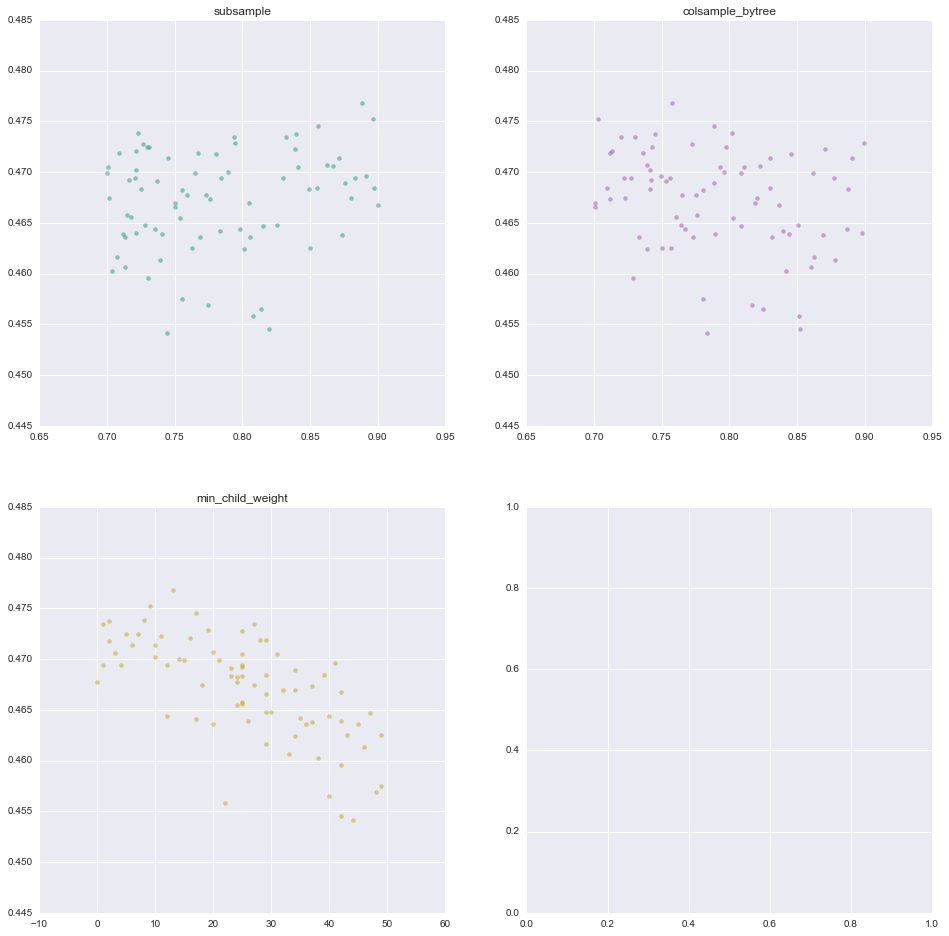


### **使用HyperOpt进行超参数优化**

* HyperOpt用于进行超参数的贝叶斯优化。[XGBoost模型](https://github.com/joostgp/kaggle_bosch/blob/master/model_hyperopt.ipynb)和[RandomForest的](https://github.com/joostgp/kaggle_bosch/blob/master/model_hyperopt-stacker.ipynb)脚本



下面是示例输出

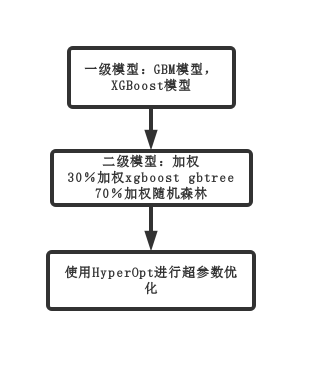


**3.2.3** 方案三结果、排名等

结果： 0.50888 排名：8

**3.2.4** 方案三算法流程图

**图3-3**



4. 算法比较

描述几种算法的基本情况及效果对比，以表格形式：

|  | **评估指标** | **特征工程** | **基础算法** | **基本库** |
| --- | --- | --- | --- | --- |
| **算法1** | **MCC** | **XGBoost** | XGBoost  ensemble | Sklearn |
| **算法2** | **MCC** | **XGBoost** | XGBoost，Extra Trees Classifiers, Neural Nets, Random Forests，Linear Models | Sklearn |
| **算法3** | **MCC** | **xgboost** | xgboost，Random Forests，  LightGBM | Sklearn |

**表4-1 算法比较**

对于算法优劣的想法描述（加分项）

**XGBoost：**

本竞赛中的大部分特征工程都是使用XGBoost算法，建模中也都用到了XGBoost。

竞赛数据特点：数据量大，且有许多缺失值。

XGBoost优点：

1.xgboost 还考虑了当数据量比较大，内存不够时怎么有效的使用磁盘，主要是结合多线程、数据压缩、分片的方法，尽可能的提高算法的效率。

2.xgboost考虑了训练数据为稀疏值的情况，可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率。

**LightGBM：**

LightGBM 是一个梯度 boosting 框架，使用基于学习算法的决策树。它可以说是分布式的，高效的，有以下优势：

更快的训练效率;低内存使用;更高的准确率;支持并行化学习;可处理大规模数据

对于本次大规模数据，LightGBM有明显优势。

5. 总结与展望

**5.1 总结**

**数据特点：**数量庞大。

**数据探索：**

1.缺失值多：稀疏矩阵；

2.有泄漏：根据数据特征探索工作站与站之间的关系；制造工程可视化，直接观察

**特征工程：**XGBoost

**验证：**

第一名：开始使用5倍交叉验证，后改为4倍“泄密分层”交叉验证（强制每次折叠都有相同数量的重复）

第三名：5倍交叉验证

第八名：5倍交叉验证

**建模：**XGBoost，LightGBM，集成模型

**5.2 建模思路**

预处理：稀疏矩阵

特征工程:XGBoost

L1:选择一些模型，组成集成模型

XGBoost：进行模型选择

L2:将筛选出的模型，组成集成模型

XGBoost

L3：集成