

# 通过时间、地点信息预测三藩市犯罪类型

高思达  
计 35 班  
2013011413  
gsd13@mails.tsinghua.edu.cn

魏鑫鼎  
计 32 班  
2013011316  
weixinding@gmail.com

张浩天  
计 35 班  
2013011416  
zhang-ht13@mails.tsinghua.edu.cn

**摘要**—这篇文章是清华大学 2016 年数据挖掘课程的大作业报告。我们小组选择参加 Kaggle 竞赛：三藩市犯罪分类 (San Francisco Crime Classification) 作为本次大作业的项目。在该项目中，我们通过学习部分标记了犯罪类型的时间、地点数据，对另一部分未标记的时间、地点数据预测犯罪类型。我们基于数据的可视化分析，利用已有的数据构造了一些更高层次的特征，并使用多种模型进行预测和评价。我们发现，我们构造的特征几乎都出现了对于训练集数据过拟合的问题，导致对于测试集的预测产生了严重的负面效果。最终，预测效果最好的方法为，使用原始特征训练 Xgboost 模型进行预测，该结果在排行榜上的排名为 81 名 (共 2335 支队伍，约前 4%)。

## I. 引言

三藩市 (San Francisco) 是一座在美国加州的淘金热潮中繁荣起来的城市。然而在城市快速扩张、人口迅速增加的同时，红灯区等诸多社会问题逐渐涌现出来，这些问题直接表现为犯罪率的上升。时至今日，三藩市早已不再是当年的矿镇，而已经成为了世界的一大科技中心。但是，常年高企的犯罪率却仍然没有好转。

作为科技重镇，三藩市公开了 2003 年至 2015 年的犯罪记录供数据科学家研究之用。通过挖掘这些数据，可以使得警察局能够更好地了解不同类型犯罪的趋势。通过根据时间和地点信息预测犯罪类型，警察局能够提前了解犯罪可能性更高的地区和时段，并针对性地采取措施以降低犯罪率。这个犯罪记录数据集在公开后，也成为了 Kaggle 网站的一个比赛。

本篇文章中，第 II 节对问题和数据集进行描述；第 III 节对数据进行可视化探索分析；第 IV 节介绍特征的提取方法；第 V 节介绍使用的分类模型，并比较实验结果；最后，第 VI 节进行总结。

## II. 问题描述

### A. 数据集

本次 Kaggle 竞赛使用的数据集来自三藩市警察局的犯罪事件数据库，包括了 2003 年 1 月 1 日至 2015 年 5 月 13 日的 878049 条犯罪记录 (共 4510 天，每天平均发生约 195 起犯罪)。训练集和测试集使用隔周的方式划分，即从第一周开始，奇数周作为训练集，偶数周作为测试集。

在数据中，犯罪细节描述 (Description) 和犯罪解决方式 (Resolution) 两个字段只有训练集才有，因此我们无法利用这些信息。除此之外，我们可以利用的字段如下：

- **Date**: 犯罪案件的时间戳，包括年、月、日、时、分、秒。
- **DayofWeek**: 犯罪案件发生在星期几。
- **PdDistrict**: 犯罪案件发生在哪个警局的管辖地区。
- **Address**: 犯罪案件发生的街道地址。
- **X**: 犯罪案件发生地点的经度。
- **Y**: 犯罪案件发生地点的纬度。

### B. 预测和评价

在训练集中，额外给出了一个字段：

- **Category**: 犯罪类型。

该字段是预测的目标。数据集中，一共有 39 种犯罪类型。在测试集上的预测结果并不是直接给出某一种确定的犯罪类型，而是需要给出该事件、地点 39 种犯罪类型的概率分布。竞赛进行评测的方法是，计算多分类的对数损失 (multi-class logarithmic loss)，即

$$\text{logloss} = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})$$

其中， $N$  是测试集样本总数； $M$  是类型总数 (本问题中  $M = 39$ )； $y_{ij}$  为一个标志量，当样本  $i$  属于类型  $j$  时为 1，否则为 0； $p_{ij}$  是模型对样本  $i$  属于类型  $j$  预测的概率。可见，当预测完全准确，即  $p_{ij} = 1$  时，对损失的贡献为 0；预测概率  $p_{ij}$  越小，对损失的贡献越大。因此，这是一个合理的对于多分类问题的评价标准。

在实验过程中，由于比赛每天限制 5 次提交，因此我们需要在训练集通过交叉验证来评价预测方法 (特征和模型)。在交叉验证中，我们同样使用 *logloss* 来评价结果。实验中，我们直接调用了 Python 的 sklearn 包中的实现。

## III. 数据观察

由于我们参加比赛的时候，比赛已经进行了大约 8 个月，Kaggle 上已经有了较多公开的脚本。在这一节的分析中，

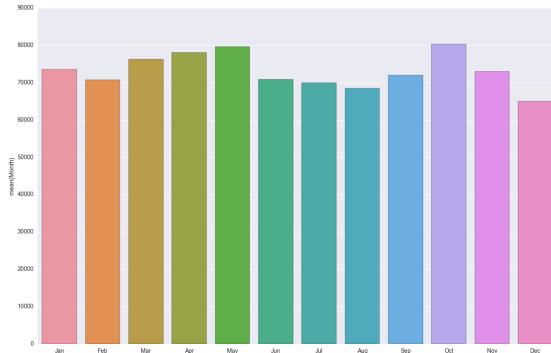


图 1. 月份对犯罪总量的影响

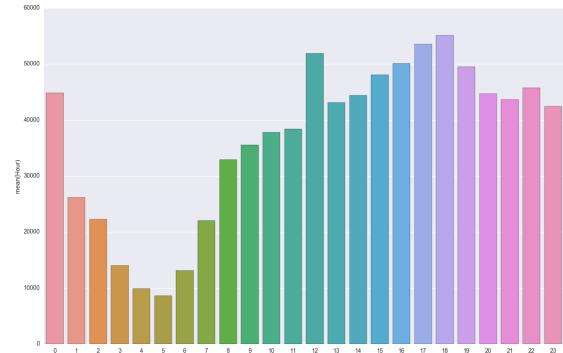


图 3. 小时对犯罪总量的影响

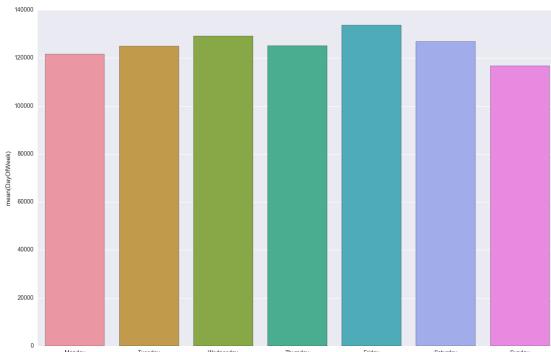


图 2. 星期对犯罪总量的影响

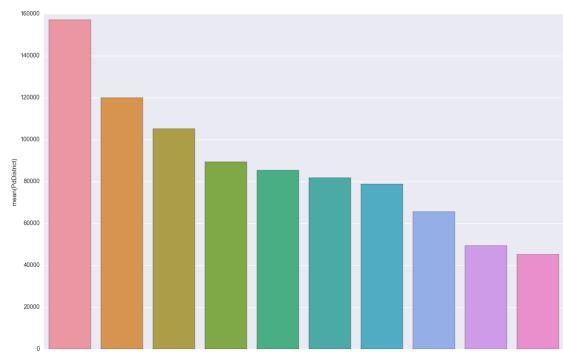


图 4. 各警察局片区的犯罪总量影响

一些可视化结果是参考了公开发布的脚本做出来的。值得说明的是，除了本节之外，第 IV 节中提到的所有数据处理和第 V 节中使用的模型都是我们独立实现的脚本，没有参考其他任何来源的资料。

#### A. 犯罪总量分析

在这一小节，我们对数据集进行粗粒度的分析。即，只变化数据点的某一个字段，而对其他字段的全体可能取值统计总数，观察每个字段的取值对于犯罪总体情况的影响。所谓犯罪总体情况，就是忽略犯罪类型，关注犯罪案件的总数。这部分分析看似并不能帮助我们进行分类，但是可以先帮我们从一个比较宏观的角度看时间和地点是如何影响犯罪的。站在实际应用角度，从时间、地点等侧面统计总体犯罪情况，可以作为当地警察局部部署警力的有益参考。

图 1、图 2 和图 3 分别展示了月份、星期和小时对犯罪总量的影响。可见，月份变化时，犯罪总量会发生比较明显的起伏波动，但相比之下，每周的每一天的犯罪总量的变化则不那么明显。符合预期的是，小时对于犯罪的影响是最显著的：中午、傍晚和午夜有三个比较明显的犯罪高峰，

而凌晨的犯罪则比较少，这一分布情况用生活经验很容易理解。

下面观察不同地点对犯罪总量的影响。首先，由图 4 可以看出，不同警察局区域的犯罪量的差异是巨大的。由此可以推断出城市不同区域的犯罪率有巨大的不同。作为验

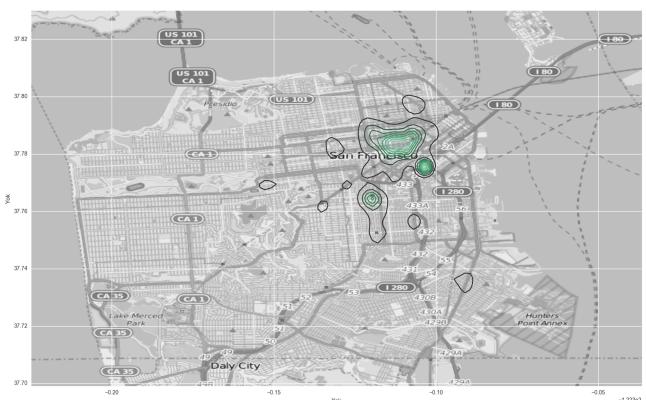


图 5. 犯罪总量的热度图

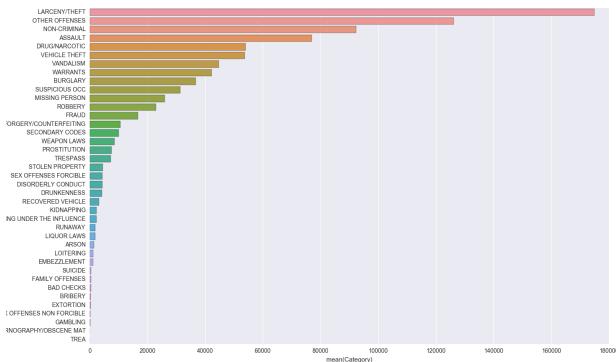


图 6. 不同犯罪类型的总量

证，我们参考 Kaggle 上公开的一份代码<sup>1</sup>，对全部类型的犯罪总量做密度图，即图 5。从中可以看出，城市东北角的犯罪率要显著高于其他部分。

### B. 犯罪类型分布的分析

首先，我们先不区分时间和地点，考察总体上各种犯罪类型的分布。图 7中可见，不同类型的犯罪总量是一个长尾分布，即频数最高的几个犯罪类型的总量占到了总体犯罪总数的大部分（最多的 5 种犯罪占整体犯罪总数的 66%）。由此可以推知，有些犯罪类型的概率总是很高的，即便一些长尾类型在某个地点集中，它的概率可能仍然没有常见类型的概率大。这种情形将会对我们的预测带来一些困难。下面，带着对犯罪类型分布的总体认识，我们进一步从地点、时间角度进行一些分析，并进一步对时间做出限制进行细粒度分析。

1) 不同犯罪类型对地点的分布：图 5中已经看到了犯罪总量对于地点的分布，发现犯罪整体上趋向于同一个中心。如果每一种犯罪类型单独分析时，都趋向于这个中心，那么可以认为地点对于每种犯罪类型的影响是一致的，即地点信息对于预测犯罪类型没有任何帮助。但是，图 7中的结果可以看到，虽然绝大多数犯罪类型都会有城市东北角的密度中心，很多犯罪类型也会有其他的密度中心。比如，纵火 (ARSON) 和赌博 (GAMBLING) 都在城市的东南角有一个密度中心。更有趣的是一个例子是，离家出走 (RUNAWAY) 的密度中心和其他任何一种犯罪类型的都十分不同，这应该是源于“离家出走”其实并不是真正的犯罪。总之，根据图 7，不同犯罪类型在地点上的分布是有所不同的，所以地点信息对预测很有帮助。另外值得注意的是，频率最高的几种犯罪类型：盗窃 (THEFT)、其他侵犯 (OTHER OFFENSES)、无罪 (NON-CRIMINAL)、攻击 (ASSAULT)、吸毒 (DRUG) 的密度中心都集中在

城市东北角，即全部犯罪类型的集中位置。也就是说，对于高频犯罪，地点信息的辨别能力不强。

<sup>1</sup><https://www.kaggle.com/codechamp/sf-crime/crime-density-by-location/code>, 作者 codechamp

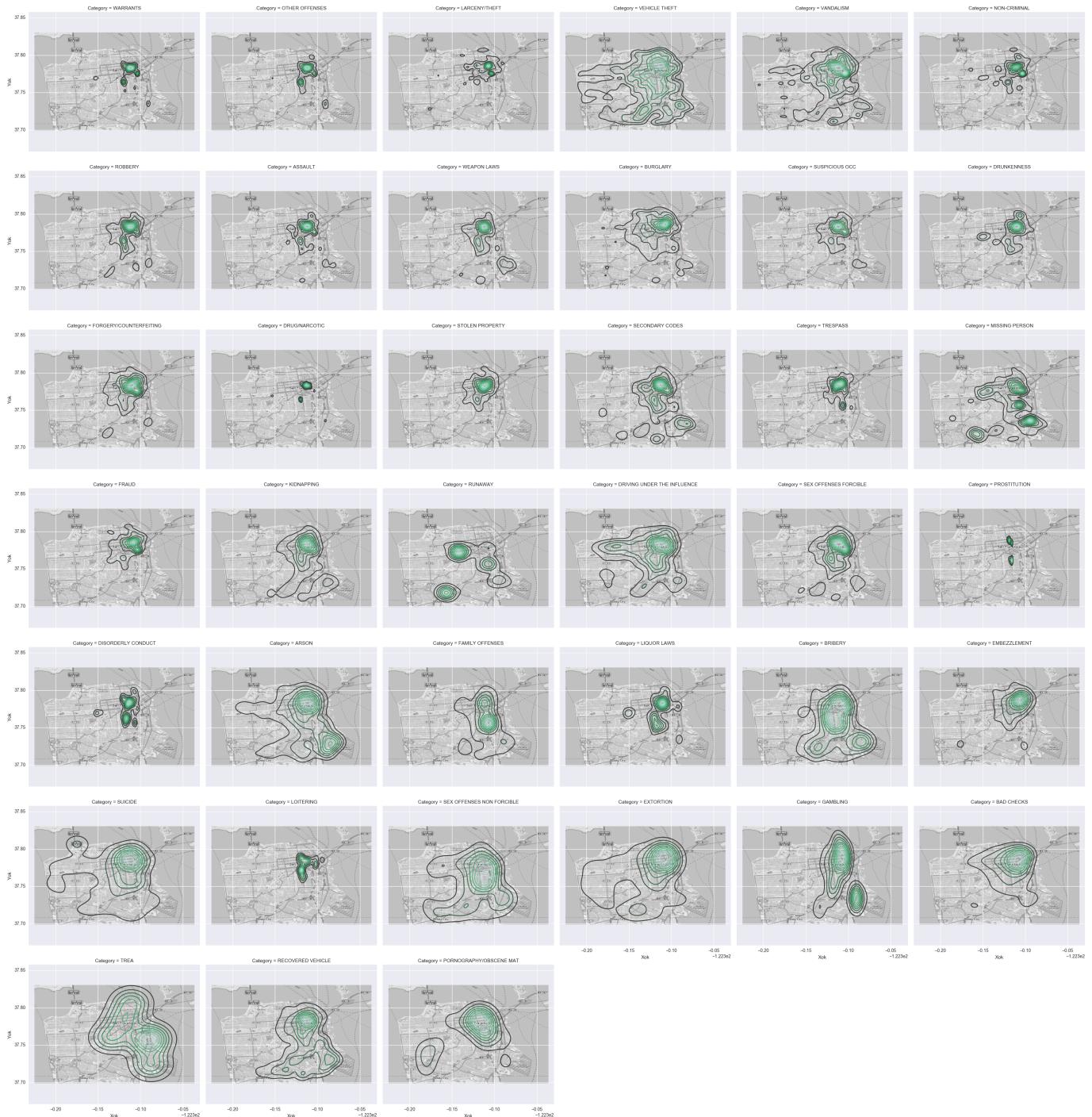


图 7. 每种犯罪类型的密度图

2) 不同犯罪类型对时间的分布: 对于时间分布的分析, 我们采用的方法是比较不同时间特征下, 犯罪分布的相似程度。对于离散分布的相似性比较, 我们使用的方法是计算两个分布的相关系数:

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

其中,  $x_1, x_2, \dots, x_{39}$  表示一个时间特征下的犯罪类型分布(即每种犯罪类型发生的概率);  $y_1, y_2, \dots, y_{39}$  表示需要比较的另外一个分布。如果两个分布接近, 即几乎满足  $y_i = x_i$ , 则线性相关程度较高, 计算出的相关系数也会越接近 1。否则, 如果分布差异很大, 则计算出的结果会更接近 0。可见, 使用相关系数比较离散分布的相似性是合理的。

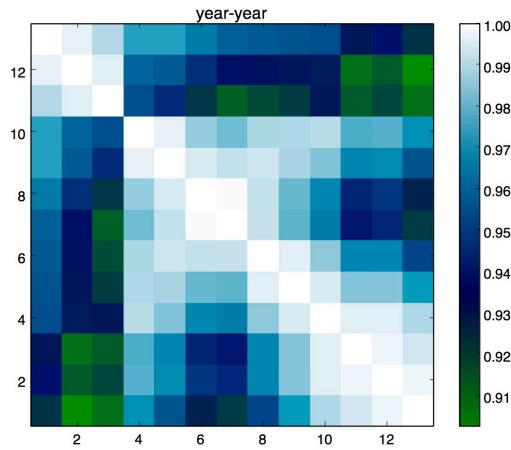


图 8. 不同年犯罪分布的比较

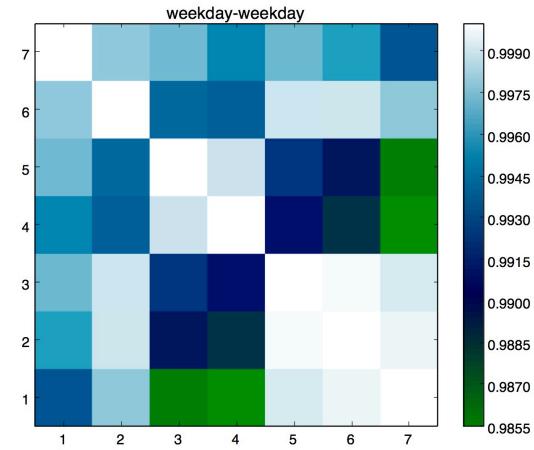


图 10. 每周各天的犯罪分布的比较

从图 8可以看出，不同年份犯罪类型分布还是有一定差异的。特别地，我们看出一个明显的分界，那就是第 3 年（2005 年）和第 4 年（2006 年）之间。前三年的分布比较接近，后就九年的分布比较接近，而这两个时间段之间的差异还是比较大的。图 9中的结果则表明，不同月份之间的分布相似度都较大（相关系数最小的也有 0.992），但是，也可以通过图中对角线的浅色区域看出月份越接近，犯罪分布越接近。图 10中则表明，每周各天的差异也比较小（相关系数至少有 0.9855），但是也明显看出周末，即周五、周六、周日的犯罪分布十分接近，而一周中间的周三、周四和其他各天的犯罪分布的差异都比较大。

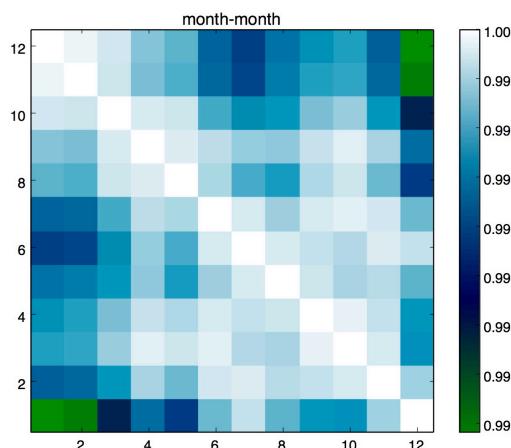


图 9. 不同月犯罪分布的比较

3) 细粒度分析方法：本部分对犯罪分布进行较细粒度的分析，即不仅仅考虑犯罪的总量，也要深入下去对每种不同的犯罪进行考虑。我们既关心不同犯罪类型在时间、空间上的分布是否不同，也关心不同的时间、地点下犯罪类型的分布（即一个  $M = 38$  项的多项分布）是否不同。因此，我们不能仅仅考虑在全部时间段上的计数，而需要对考虑的时间做出一些限制。

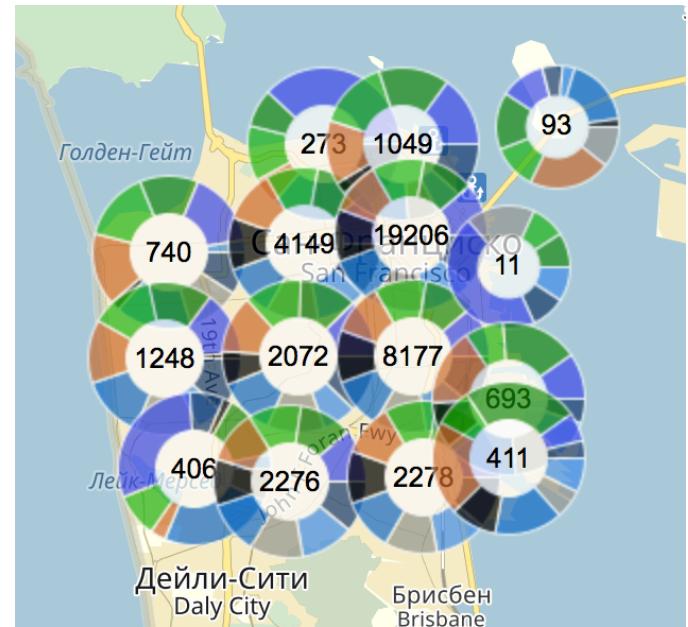


图 11. 在线分析工具给出的聚类结果

这种细粒度的分析方式如果使用一个数据仓库是比较方便进行的，但数据仓库的实现十分复杂。在 Kaggle 论坛上，有参赛者公开了一个十分方便的交互式可视化分析工

具<sup>2</sup>, 可以起到数据仓库的作用。在这个工具中, 我们可以对时间进行多种限制: 指定日期区间、年份区间, 指定某几个星期几和某几个小时。在此基础上, 我们可以选定几个犯罪类别。该工具的输出是指定的犯罪类型, 在指定的时间限制下, 在地图上的聚类结果。同时, 在每个聚类中心还会展示这个聚类中每类犯罪的分布情况。这个工具对于我们的探索分析是十分理想的。

图 11 的结果便来自这个工具。图中给出了一定时间范围内, 指定的几种犯罪的聚类结果和分布。可以再次验证不同地点的犯罪类型分布是不同的。同样, 通过缩小时间范围, 并比较不同时间范围的分布, 也可以验证之前关于时间的发现, 但其实不如我们之前的分析清晰, 这里不再罗列。

经过数据可视化分析, 我们可以明显地观察出地点和时间信息对于犯罪类型分布的影响。虽然我们发现的一些分布规律不容易直接应用到预测任务之中, 但是, 这种分析至少让我们可以肯定如果提取适当的特征、选择合适的模型, 确实可以通过时间、地点信息推测出犯罪类型。

## IV. 特征工程

### A. 对原数据集进行数值化

由于原始的数据集中的字段除了经纬度之外都是使用字符串形式给出的, 为了方便后面的处理, 我们先对各个字符串字段进行数值化。每个字段的数值化方法不只一种, 除了直接对应为一个整数之外, 也有更合理的方式进行数值化。

**Date** 对于时间戳字符串, 我们直接把它进行分割, 拆成 6 个字段, 即年、月、日、时、分、秒。每个字段都是一个整数。每一个字段的大小关系都是有含义的 (时间的先后顺序、接近程度), 因此使用整数是比较合理的。除了直接对应外, 也可以进行离散化, 比如对一天中的时间进行分段。但是在实验中, 我们并没有进行这种尝试。

**DayofWeek** 对于星期几这个字段, 我们直接把星期一到星期天编码为整数 1 到 7。这个字段使用整数编码是有意义的, 整数的大小关系可以表示星期的先后。但是, 由于星期是来回轮转的, 因此使用一个 7 维二进制向量对星期进行编码也是合理的: 比如, [ 1 0 0 0 0 0 0 ] 表示星期一, [ 0 1 0 0 0 0 0 ] 表示星期二。

**PdDistrict** 警察局字段一共有 10 个不同的取值。我们可以直接地映射到 1 到 10 十个整数, 但是这种编码是不合理的, 因为这会在警察局之间引入大小关系 (线性关系), 而这种关系实际是不存在的 (即, 人为加入了干扰信息)。所以, 更合理的方式是, 使用 10 维的二进制向量进行编码。

<sup>2</sup><https://www.kaggle.com/tyz910/sf-crime/yet-another-map/notebook>, 作者 Evgeny Ivanov。

**Address** 地址信息的处理比较棘手。地址字段有三种形式: 一个街道名; 两个街道名; 或者一个街道名和一个街区号。在训练集和测试集中, 一共出现了 2128 种街道名称, 我们首先按照字典序对街道进行整数编码 (1 到 2128)。然后使用如下方式, 用两个整数表示地址:

- 两条街道名的地址: 使用两街道的编号, 小编号在前、大编号在后;
- 街道和街区号的地址: 首先是街区号的相反数 (小于等于 0), 然后是街道编号。
- 单一街道名: 首先是街道编号, 然后是 -1。

这是从字符地址到整数的一种直接翻译, 这些整数之间的大小关系、线性关系都是没有实际意义的。显然这不是一种很好的编码方法。但是, 由于 Kaggle 比赛要求不能引入外界信息, 我们无法使用街道的朝向和相互之间的位置关系对街道进行编码, 只能使用字典序这种办法进行编码。在实验中, 我们只利用了地址中的街道信息。

至此, 我们得到了数据集直接数值化后的特征。经过这一处理的特征, 虽然数值化的方法仍有不合理之处, 但我们已经可以用这个特征来训练模型了。

### B. 对于街道进行犯罪概率统计

由于我们对于街道的编码方式不合理, 我们希望构造新的特征来更好地刻画街道的特性。一个直观的想法是, 统计各个街道上的犯罪概率分布。首先, 统计各个街道上 39 种犯罪的总数。对于每一个地址, 使用如下步骤构造特征:

- 1) 如果地址中出现了两条街道, 则对两个街道的犯罪计数求和。否则就直接取所在街道的犯罪频数。
- 2) 进行 Laplace 平滑处理, 即对每一类的频数加 1。
- 3) 计算每种犯罪类型在该街道上的频率  $p_j = \frac{f_j}{\sum_{k=1}^M f_k}$ , 其中,  $f_j$  是类别  $j$  的频数,  $M$  为总类型数 39。
- 4) 把每个频率  $p$  替换成  $\text{logodds}(p) = \log \frac{p}{1-p}$ 。这一处理可以更好地刻画频率之间的大小差距。

至此, 我们使用 39 个浮点数刻画了犯罪地点所在街道的犯罪类型特征。注意到, 上面对每类犯罪频率的计算是考虑了整个训练集的, 即在时间上的粒度是很粗糙的。但是, 如果每个街道上实际存在一个犯罪分布, 那么在更长的时间段中计数应该能更准确地计算这个分布, 所以这样对全局计数是合理的。

### C. 利用时空数据点的 K-最近邻信息

利用经纬度坐标信息, 我们可以计算两个数据点之间的距离, 由此可以采用 K-最近邻方法来进行预测。除了直接进行预测外, 也可以根据 K-最近邻的结果构造特征。此外, 考虑到除了接近的地点可能有类似的犯罪类型分布之外, 接近的时间也可能有类似的犯罪类型分布。根据第 III 节中的结果, 在时间的分布上, 月份 (直接与季节、节日相关) 和小时 (白天和半夜的犯罪情况差别很大) 应该

对于犯罪有很大的影响。于是，我们构造了四维数据点 ( $X$ ,  $Y$ , Month, Hour) 进行 K-最近邻分析。

注意到，通过欧式距离计算最近邻的时候，需要对各个维度进行标准化。特别是对于本数据集构造的这种四维数据点。其中， $X$  的 25% 分位数和 75% 分位数的差只有  $122.43 - 122.40 = 0.03$ ，同时  $Y$  的 25% 分位数和 75% 分位数的差也为  $37.78 - 37.75 = 0.03$ 。而刻画时间的两个维度都是整数，差至少为 1，远远大于经纬度之间的差。如果不做标准化处理，则时间的差别会掩盖掉经纬度的差别，这样进行最近邻分析就没有意义了。因此，对于 4 位数据点的每一个维度，使用下面的方法进行标准化：

$$X' = \frac{X - \bar{X}}{s}$$

其中， $\bar{X}$  为该维坐标的平均值， $s$  是该维全体坐标的标准差。由这些坐标点得到的 K-最近邻信息，我们可以使用两种办法编成特征。其一，直接把 KNN 模型的预测概率结果作为特征；其二是根据最近邻的距离进行加权得到每个犯罪类型的权值。加权的方式，可以简单地对距离取倒数，即类型  $i$  的权重为：

$$w_i = \sum_{j=1}^N \frac{1}{d_{ij}}$$

#### D. 利用聚类结果构造特征

对于一种犯罪，我们根据生活经验，认为其具有一定的空间局部性，就是说某些区域发生某种犯罪的几率会比较高。因此我们决定做一些处理来体现空间局部性。

对于每种犯罪，我们统计所有该种犯罪的记录的地点信息，可以得到二维平面点的信息，这些数据使用 K-means 算法进行聚类。但是对于一种犯罪，有几个犯罪中心实际上是不确定的，K-means 算法也不能自动分辨出有几种类别。为了解决这个问题，我们枚举了类别数进行计算，为了度量分几类最好，我们将数据分为两折，一折用来聚类，另一折用来测试，用在测试集上和在训练集上的相对误差来选取一个最好的分类数。

如此，我们便得到了每种犯罪的若干中心，接下来我们把各个犯罪记录到每种犯罪的最近的中心距离计算出来，作为特征加入到特征向量之中。

## V. 模型实验

### A. 实验框架的实现

本次实验框架主要使用的是 Python 的 sklearn 包进行实现的。此外，还使用了 Python 的 xgboost 包。后者的模型也进行了 sklearn 的封装，因此我们可以编写统一的实验框架。我们使用的实验框架中，使用的模型、特征文件、特征列表、模型参数等等都可以以命令行参数的形式进行指定。这样，在实验时，我们不需要修改训练模型的

代码，只需要修改命令就行了。同时，通过记录命令的详细内容和每次实验的中间结果，我们可以很方便地对实验过程进行总结，比较特征的能力，并把模型调到最优的表现。

### B. K 最近邻

这部分，我们首先对于前面在 IV-C 中设计的四维空间数据点进行简单探索。我们使用 KNN 模型，以四维时空数据点作为训练数据进行交叉验证测试，观察做数据标准化的效果。我们对数据统一进行 3000 邻居的 KNN 预测，标准化之前的交叉验证结果为 2.67，标准化之后的交叉验证结果提升到了 2.62。说明数据标准化确实是有利的。但是，现在的效果还有提升的空间：我们可以进一步调整时间、空间特征差异的权重（现在二者的权重是一样的），强调某些更重要的特征，这样可能能用 KNN 得到更好的结果。

### C. 随机森林

由于随机森林的训练也测试速度比较快，因此我们使用随机森林模型来比较不同特征的预测能力。使用的特征有：

- **特征 0：**直接对原始数据文件数值化之后的特征，其构造方法为第 IV-A 节中的描述。
- **特征 3：**在特征 0 的基础上，加入街道特征。街道特征的构造方法为第 IV-B 节中的描述。
- **特征 4：**在特征 0 的基础上，加入时空数据点的 K 最近邻特征。K 最近邻特征的构造方法为第 IV-C 节中的描述。
- **特征 5：**在特征 0 的基础上，加入空间坐标的聚类结果特征。聚类结果特征的构造方法为第 IV-D 节中的描述。

实验中，我们对随机森林使用两组参数：

- **小规模随机森林：** 使用更少的树，每棵树的规模也更小。具体为：30 棵树，树的最大深度为 8，叶子最少有 80 个样本。
- **大规模随机森林：** 使用更多的树，每棵树的规模也更大。具体为：300 棵树，树的最大深度为 15，叶子最少有 30 个样本。

表 I  
随机森林比较各种特征

| 特征编号 | 模型参数 | 交叉验证结果 | 提交结果 |
|------|------|--------|------|
| 0    | 小规模  | 2.42   | 2.45 |
|      | 大规模  | 2.29   | 2.30 |
| 3    | 小规模  | 2.31   | -    |
|      | 大规模  | 1.82   | 2.37 |
| 4    | 小规模  | 2.27   | -    |
|      | 大规模  | 2.03   | 2.49 |

表 I 中展示了我们使用随机森林测试不同特征的结果。首先可以看到的是，规模更大的随机森林能够得到更好的

表 II  
分折编制街道特征后的结果

| 特征编号   | 模型参数 | 交叉验证结果 | 提交结果 |
|--------|------|--------|------|
| 3 (分折) | 小规模  | 2.44   | -    |
|        | 大规模  | 2.38   | 2.37 |

预测效果。但是，在增加了我们编制的特征之后，发生了比较令人费解的现象：虽然交叉验证的结果出现了巨大的提升，但是提交之后，在测试集上的结果却变差了。首先，既然能够在交叉验证时看到预测效果的提升，这可以说明我们编制特征的方法有一定效果。但是，发生了在训练集上效果好，而在测试集上效果远远更差的情况。显然，我们的特征发生了过拟合。推测的原因是，同一周之内的犯罪信息对于预测的影响过大。

为了验证这个想法，我们将训练集分为两折重新编制特征 3，分折的方法仍然为隔周划分（奇数周为第 1 折，偶数周为第 2 折）。第 1 折的街道特征，使用第 2 折的街道信息进行统计和计算；第 2 折的街道特征，使用第 1 折的街道信息进行统计和计算。通过这种方式可以保证每一个数据点的特征，都没有参考同一周的数据进行计算。此时得到的结果见表 II。首先可以看到，测试集的结果和训练集比较接近了，这是一个很好的标志。但是，对比表 I 中的结果可以发现，此时的预测结果仍然劣于原始的特征 0。此外值得注意的是，使用一半数据编制的街道特征在测试集上的表现和使用全部数据的表现几乎一致。由此我们可以推断，使用全部时间段内的街道犯罪类型分布，对于随机森林可能只有负面的效果。

那么，为什么表 I 中特征 3 的交叉验证结果会格外优秀呢？我们推测，这是因为在统计街道犯罪类型分布时，我们并没有包含每个数据点本身。于是，在预测时，如果发现某一类的概率比别的样本都小一点点，那么这个样本很可能属于这个类。所以，在编制特征 3 时，我们无形中提示了该样本的类型。正因如此，我们才在训练集上看到了一个格外好的假象（在测试集上才能看到这一特征的真正表现）。

此外，我们也利用训练出的随机森林模型，对各个特征的重要程度进行了分析，结果如图 12。其中的大多数结果比较容易理解，比如地点坐标的重要性很高。根据第 III 节中对时间特征的分析，我们也可以理解年份和小时都是比较重要的。但是，比较令人费解的是，分钟成为了重要性最高的特征。我们尝试给出的解释是，犯罪记录中的时间并不是犯罪真正发生的具体时刻，而是该次犯罪录入档案的时刻。在记录的过程中，可能在不同犯罪类型之间，人为引入了令人意想不到的差别。此外，两个街道特征的重要性差别，应该是在第 IV-A 部分中，对街道地址的编码方式引入的。

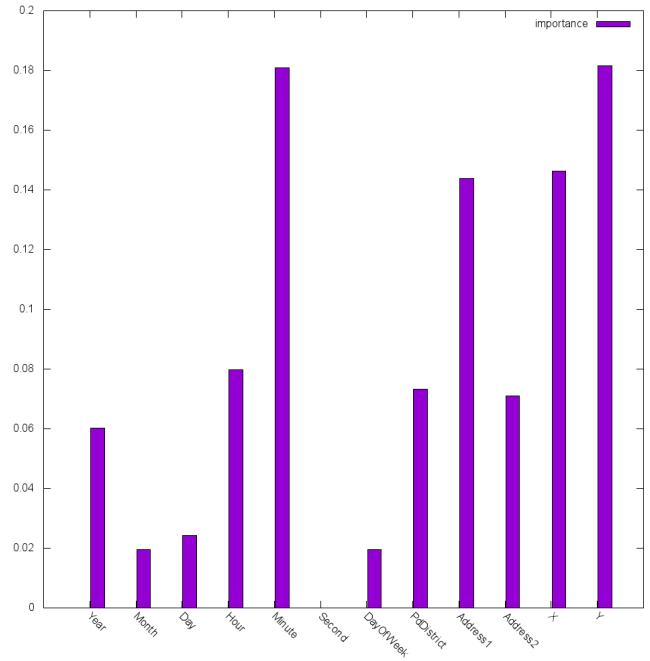


图 12. 特征重要性

#### D. Xgboost

我们最终提交的结果是 Xgboost 跑出的结果，Xgboost 在本次竞赛中相比其它我们使用的模型效果要出色很多。表 III 中是我们各次提交（各种参数配置下）的结果。

由于随机选取参数的 Xgboost 模型就取得了比之前随机森林模型的结果要好很多的结果，我们对 Xgboost 很有信心，因此花了很多时间来调参数。1 至 5 次提交是我们刚刚开始尝试 Xgboost 模型，随意选取了几组参数进行测试，我们主要调整的几组参数为迭代轮数和决策树最大高度。可以发现 Xgboost 的模型抗过拟合效果良好，通常只要增加迭代轮数，无论再小都会有所提升。另外，过高或者过低的决策树高度都是不合适的，低一点的决策树高度会导致决策表达能力不足，高一点的决策树深度会导致过拟合，并且训练时间会长不少。

进行了特征 0 的尝试之后，我们使用了特征 3 和特征 4 进行测试，得到了编号 6 和编号 7 的数据。令我们失望的是，随机森林模型表现出的问题在这里重现了。这两个特征在测试集上取得了更好的结果，但是提交之后的结果变差了很多，这让我们更加怀疑特征 3 和特征 4 由于蕴含了训练集的结果，导致了这个情况的出现，因为这两个特征都直接加入了对测试集标签的统计信息。但由于不能确定是否是参数导致的问题，我们又跑出了编号 9 的结果，没有明显改善。

为了搞清楚是否是特征 3 和特征 4 特有的问题，我们换了间接使用结果信息的特征 5 来进行训练和预测，提交得到了编号 8 的结果，与之前获得的最好的结果差距不大。

表 III  
XgBoost 调参过程

| 提交编号 | 使用特征编号 | 迭代轮数 | 决策树最深宽度 | 学习率 | 每棵树的列采样率 | 提交保留小数位数 | 交叉验证结果   | 提交结果    |
|------|--------|------|---------|-----|----------|----------|----------|---------|
| 1    | 0      | 60   | 8       | 0.3 | 1        | 3        | 2.22376  | 2.31334 |
| 2    | 0      | 100  | 8       | 0.1 | 1        | 3        | 2.259029 | 2.29657 |
| 3    | 0      | 100  | 10      | 0.1 | 1        | 3        | 2.235178 | 2.29852 |
| 4    | 0      | 100  | 6       | 0.1 | 1        | 3        | 2.289943 | 2.31782 |
| 5    | 0      | 200  | 6       | 0.1 | 1        | 3        | 2.263382 | 2.31364 |
| 6    | 4      | 200  | 6       | 0.1 | 1        | 3        | 2.208462 | 2.64618 |
| 7    | 3      | 200  | 6       | 0.1 | 1        | 3        | 2.24326  | 2.43281 |
| 8    | 5      | 200  | 6       | 0.1 | 1        | 3        | 2.229024 | 2.30504 |
| 9    | 3      | 200  | 8       | 0.1 | 1        | 3        | 2.223634 | 2.46337 |
| 10   | 5      | 200  | 8       | 0.1 | 1        | 3        | 2.21475  | 2.33183 |
| 11   | 3      | 100  | 3       | 0.1 | 1        | 3        | 2.313002 | 2.40017 |
| 12   | 0      | 100  | 3       | 0.1 | 0.8      | 3        | -        | 2.2875  |
| 13   | 0      | 100  | 8       | 0.1 | 0.8      | 3        | 2.2503   | 2.28598 |
| 14   | 0      | 100  | 8       | 0.1 | 0.8      | 5        | 2.2503   | 2.25392 |
| 15   | 3      | 200  | 8       | 0.1 | 0.6      | 5        | -        | 2.26982 |
| 16   | 0      | 200  | 8       | 0.1 | 0.8      | 5        | -        | 2.23887 |

因此我们希望进一步挖掘特征 5 的潜力，看在参数的调整下是否能获得更进一步的提升。于是我们提交了编号 10 的结果，出乎意料的结果变差了，而编号 8 和编号 10 的区别只有最高决策树高度这一参数。因此我们怀疑是对于此类引入训练集结果信息的特征，模型表达能力越强越容易过拟合。因此我们决定接下来使用不引入结果信息的特征 0 来进行训练和预测。

在这之后，我们陷入了一段瓶颈期，测试各种参数的组合得到的本地结果并没有迹象能比较有效的超过编号 2 的结果。于是我们决定调整新的模型以求得到提升，在查阅文档之后，发现 Xgboost 提供了列采样率这一参数，就是说对于每轮建立的新的决策树，并不把特征的所有列作为决策树需要分类的数据，而是随机的选取其中的一部分作为决策树需要分类的数据。这个参数能使得各轮训练得到的决策树区别变大，由于我们的特征维数并不高，容易导致决策树的同质化，因此我们认为这个参数会带来较大的提升。调整了这个参数之后，我们得到了编号 12 的结果，终于得到了优于编号 2 的结果的结果。

编号 12 的参数的决策树最高高度较低，主要是因为我们发现了 SKlearn 的 Gradient Boost 模型的默认参数中决策树最高高度为 3，因此做这个尝试，但是根据本地训练过程中的数据，我们认为对于这个问题该参数还是大一点比较好，于是我们将该参数调高之后得到了编号 13 的结果，有所提升，但是不甚明显，这与我们本地数据出入很大。这也是我们之前的结果出现的一个普遍问题，本地测试的结果与提交结果差别较大，解决这个问题得到编号 14 的结果也是灵光乍现。之前由于提交文件过大，我们的结果文件只保留三位小数来减小提交文件的大小。但是对于本问题，预测的准确率不超过 30%，因此预测误分类的

几率还是很大的，而对于这个比赛的评分方式，假如误分类，并对于正确分类给出的概率较低，即使是小数点后四位的差别也会导致最终评分增加很多。想到这点之后我们将结果文件保留了五位小数，提交后得到编号 14 的结果，基本消除了本地测试和提交结果的区别，不再过拟合测试集数据。

之后我们由于距离提交截止时间非常接近，而且进行一次模型的训练要花很久的时间，所以最后两次结果没有进行分折测试，直接提交了结果。其中编号 16 的结果使用的参数只对于编号 14 的参数改动了迭代轮数，利用了 Xgboost 模型不宜过拟合的特性，取得了我们最终的成绩：第 81 名（前 4%）。

#### E. 逻辑斯蒂回归

虽然我们最终取得了很好的成绩，但是经过上面的实验，我们发现构造的特征都对预测有负面影响，这是一个十分令人沮丧的结果。考虑到我们主要使用的随机森林模型和 Xgboost 模型都是基于决策树的，而决策树主要只是比较特征的大小，而不会对各个特征之间进行线性运算，我们怀疑是这个原因导致了构造的特征全部失效。比如说数据有 A、B、C 三列，当  $A + B + C > 100$  时是正例，其他情况是负例，这种特征对于决策树模型可能就比较难以分析出来，而需要使用一些在特征之间有运算的模型。基于这种猜测，我们使用逻辑斯蒂回归模型对分折构造的特征 3 进行了检验。使用特征 0 训练时，交叉验证的结果为 2.65；而使用分折构造的特征 3 时，交叉验证的结果为 2.63。此时，我们确实能够观察到编制的特征的效果，这也一定程度上验证了我们的猜测：决策树类模型不能很好地利用需要通过数值计算寻找规律的特征。这类特征如果使用神经

网络类模型应该会有更好的效果，但是时间所限，我们没有做出这样的探索。

## VI. 结论

本次 Kaggle 比赛，我们通过精心调试 Xgboost 模型，使用很基础的特征就取得了前 4% 的成绩。同时，我们通过学习 Kaggle 论坛上的脚本，对数据进行了可视化处理和观察分析。我们在构造特征时从多个角度进行了尝试。实验的过程中，我们发现构造的特征对于预测只有负面效果。我们针对这一现象进行了进一步实验，并做出了两种解释：构造特征时引入了对于预测目标的提示导致在训练集上过拟合；选用的模型不能很好地利用特征中的信息。

通过本次参加 Kaggle 比赛的经历，我们学习到了很多数据挖掘的经验。在面对实际的数据挖掘问题的时候，首先要多尝试不同的模型，来得出哪些模型比较适合这个问题。另外，不能闭门造车地纯做特征工程，因为无效的特征不仅会降低最终模型的准确率，而且会增加模型的训练时间，因此增加特征值之后需要用模型对其进行检测，摒弃其中无效的部分；也不能纯凭自己的想法添加特征，应当结合对数据的分析；还要考虑当前使用的模型的表达能力是否足够利用这些特征。另外，在模型调参的过程中，要去思考模型的特性，贴合实际问题去调整参数，如果训练速度较快，可以使用自动的调参工具（Grid Search）来进行参数的选择。最后就是要相信测试集上交叉检验的结果，当出现提交结果与本地分折测试结果相差较大的时候，通常是自己在某些方面犯了错误，应当将其找出并修复。