机器学习纳米学位毕业项目

基于机器学习算法的旧金山罪案预测

梅钰婷 2019 年 1 月 20 日

I. 问题定义

项目概述

旧金山是一座繁荣的美国城市，高收入的技术人员和创纪录的游客数量为城市带来大量金钱。与此同时，这里的财产犯罪也在激增。项目的任务在于通过运用机器学习帮助警员判断罪案发生的类型，判断罪案类型的意义在于, 能够帮助警局灵活且快速地调配警力以及安排优先级, 防止警力浪费, 加快破案效率[1]。

问题所涉及到的领域为数据挖掘中的分类问题，首先从数据中选出已经分好类的训练集，在该训练集上运用数据挖掘分类的技术，建立分类模型，对于没有分类的数据进行分类[2]。这个项目的数据集来自 SFPD 犯罪事件报告系统，数据集的时间范围是从 2003 年的 1 月 1 号到 2015 年的 5 月 13 号。数据集从八个维度来描述犯罪事件的特征，还有一个目标维度为不同的犯罪事件类别。训练集和测试集按周轮换，即第 1、3、5、7 周等等属于测试集，第 2、4、6、8 周属于训练集[3]。该项目通过适当提取特征，利用训练集和验证集数据训练机器学习算法，最终将测试集所提供的犯罪数据的八个维度特征使用机器学习来预测每条数据的犯罪事件类型。

多分类算法有很多，比如决策树，贝叶斯分类、人工神经网络、SVM、XGBoost 等等。该项目使用的算法是使用 LightGBM 进行多分类预测。LightGBM 是个快速的，分布式的，高性能的基于决策树算法的梯度提升框架。可用于排序，分类，回归以及很多其他的机器学习任务中。[4]。

问题陈述

对于 39 种犯罪种类的分类是一个监督学习分类问题，我们的目标是通过挖掘适当的特征来把测试集中未知的犯罪记录进行犯罪种类的分类。

我首先会按照数据的特性讲数据进行可视化，对于存在离群的异常值进行剔除，建立适当的特征向量，挖掘出对结果预测有利的关键信息。数据集中训练集约有 88 万条数据，数据量非常大。因此建立适当数量的特征不会对模型的训练造成过拟合。通过建立的特征工程利用 sequential 添加神经层对模型进行预测，通过改变参数比较结果。

我期望模型最终能以较小的误差预测犯罪类型，并且我所建立的特征能够对目标预测具有较大的贡献。

评价指标

由于这是一个具有大容量样本的分类问题，因此我使用了两种评价指标：

- 对数损失：评价分类结果的指标。数值越小，则误差越小，预测结果越接近真实值。由二分类问题衍生到多分类问题的的对数回归，设 Y 为指示矩阵，即当样本 i 的分类为 k 时 $yi,k=1$；设 P 为估计的概率矩阵，即 $pi,k=\Pr(ti,k=1)$，则对每个样本的对数损失为[5]：

$$L_{\log}(Y_i, P_i) = -\log \Pr(Y_i|P_i) = \sum_{k=1}^{K} y_{i,k} \log p_{i,k}$$

- 时间：算法运行开始到完成预测所需的时间。因为需要处理的是一个数量庞大的数据集，因此算法的效率非常重要。因此时间长短可以作为模型评估的一个标准。

II. 分析

数据的探索

数据集可以在 kaggle 的官网上获得[3]，数据集有以下九个特征如下：

- Dates：犯罪事件的时间（包含年、月、日）
- Category ： 犯罪事件类别(仅在训练集中)，这是需要去预测的目标变量。
- Descript：详细描述犯罪事件(仅在训练集中)
- DayOfWeek：一周中的一天
- PdDistrict ： 警署分区的名称
- Resolution：如何解决犯罪事件(仅在训练集中)
- Address：犯罪事件的大致街道地址
- X：经度
- Y ： 纬度

在这九个特征中，我选择其中六个特征进行了数据分析可视化。

数据集已经被划分成了训练集和测试集，训练集总共有 878050 条数据，测试集共有 884263 条数据。训练集和测试集中每条数据都是非空值数据，因此不需要额外的对数据进行缺失值处理。训练集中的数据记录着不同的犯罪类型，如：纵火、人身攻击、伪钞、贿赂、盗窃、品行不端、酒后驾车、贪污、勒索等等。我们看一下一条随机的数据：

Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y
2015-05-13 23:53:00	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.7745

可以发现里面的数据类型有浮点数、字符串，可以直接进行分析的有 X，Y 表示的经纬度特征。因为犯罪事件具有较强的空间依赖性。因此可以经纬度应在一定区间范围内较为合理。为了研究有无离群值，我选择讲 X、Y 的数据先标准化，然后计算第一四分位数 Q1 和第三四分位数 Q3。以 1.5 倍四分位距为异常阶 step[7]，我们认为小于 Q1 -step 和大于 Q3+step 的数据为离群值。我选择讲 X 和 Y 小于离群值绝对值最大值的数据删除，并将其可视化如下：

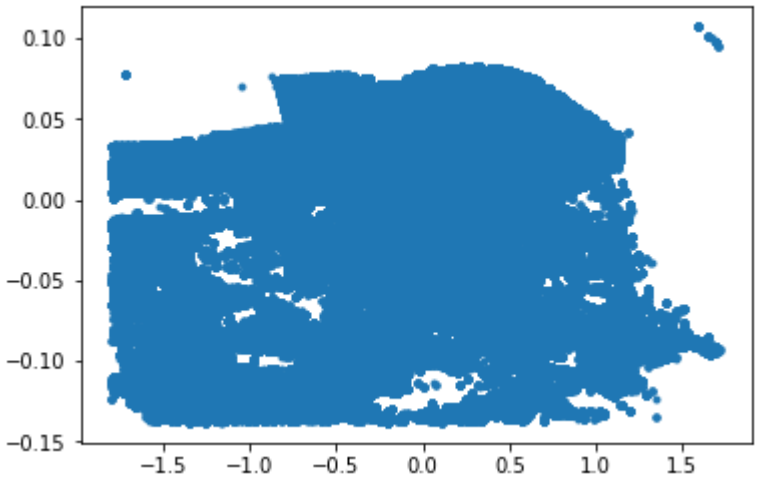


图 1：经纬度处理后图片

探索性可视化

我选择对数据的 Dates，Category，DayOfWeek，PdDistrict 进行了可视化分析。首先我对目标变量 Category 中不同种类所占数据的比例进行了可视化如下：

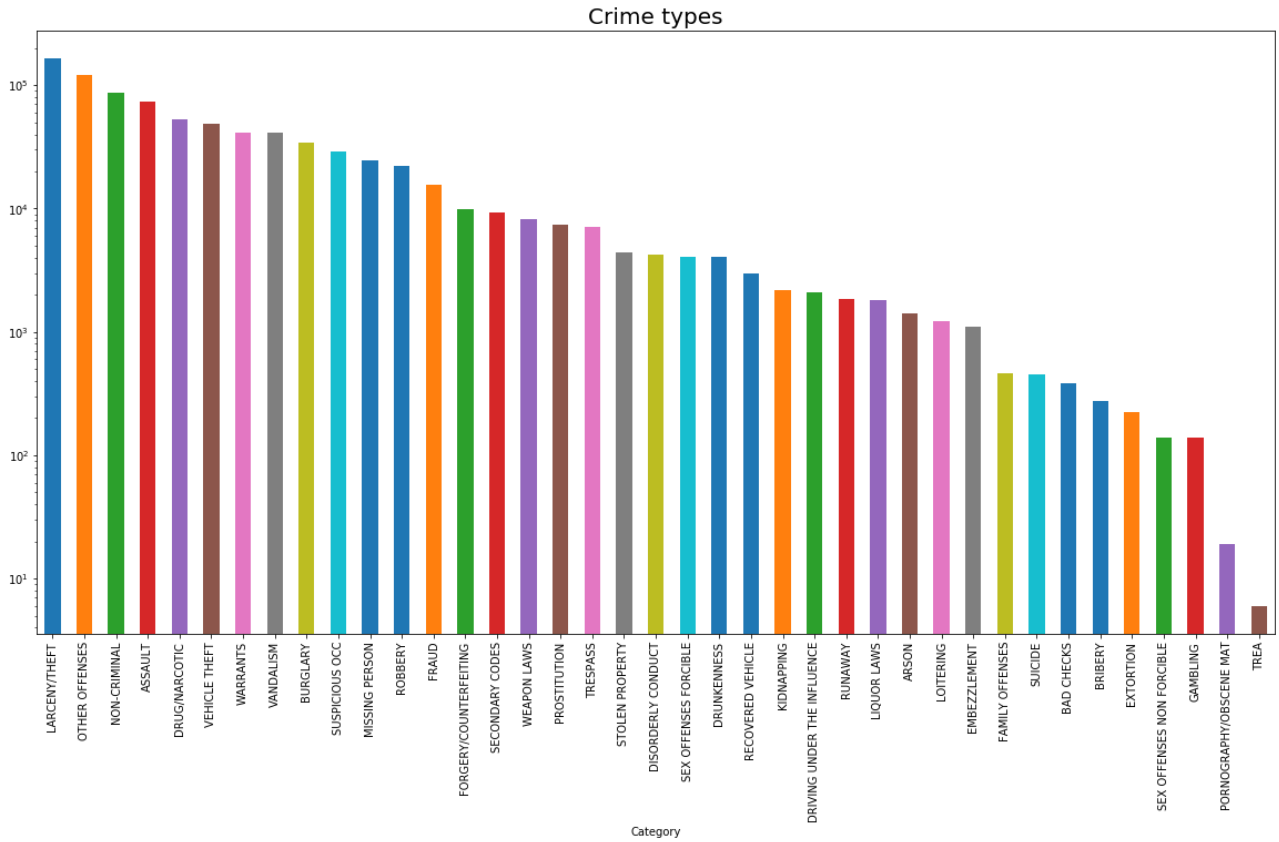


图 2：犯罪类型数目比较

可以发现这是一个样本不平衡的数据集，部分样本数量过小，可能会导致模型的泛化能力受到影响。盗窃案数量最大。

接下来可视化了不同警署分区的犯罪数量：

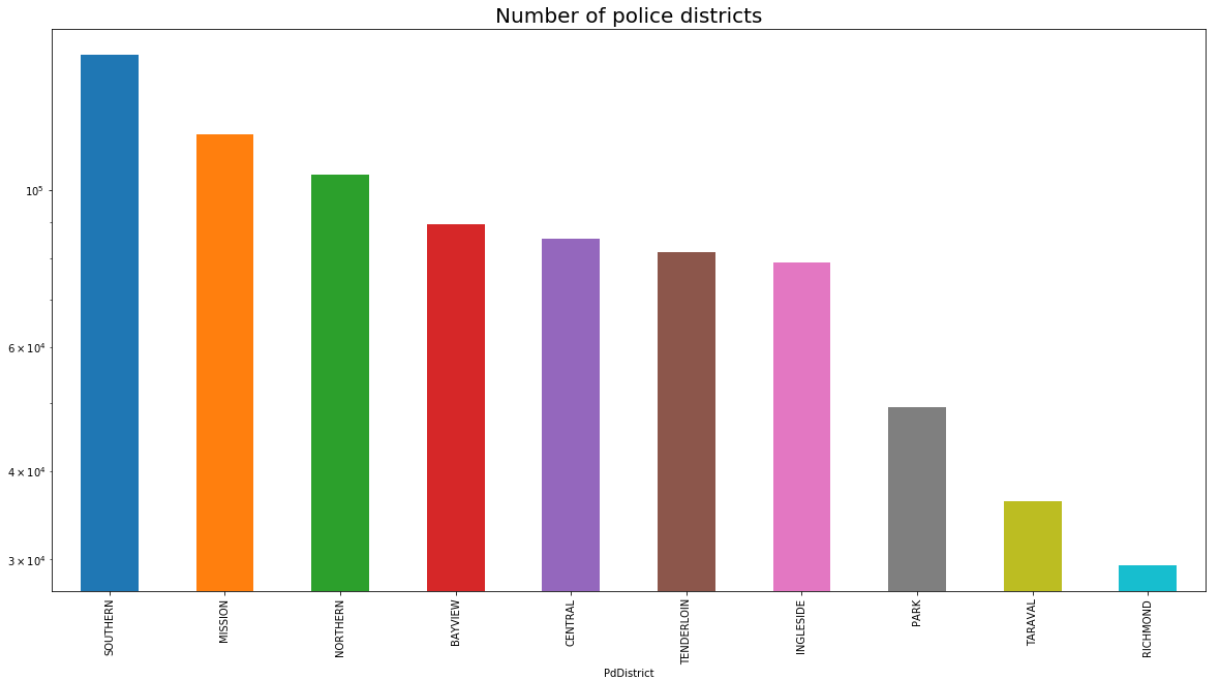
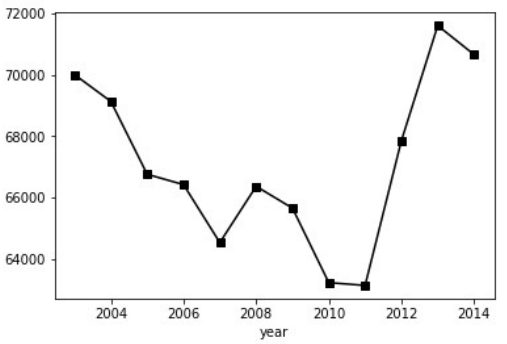
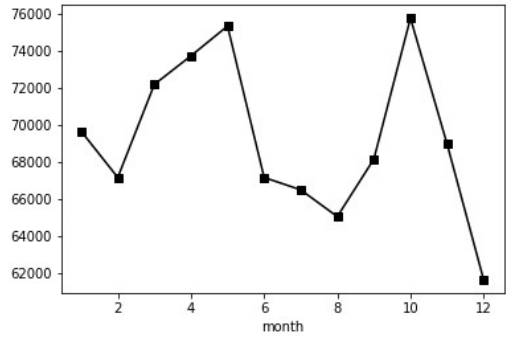


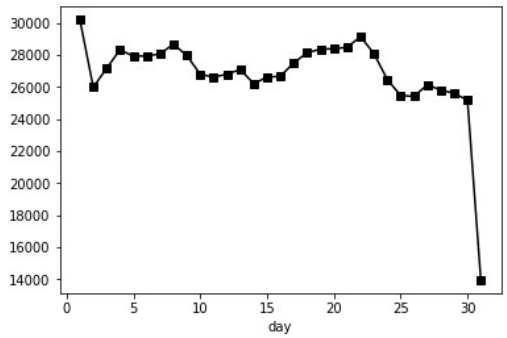
图 3：不同警署区域的犯罪数量统计

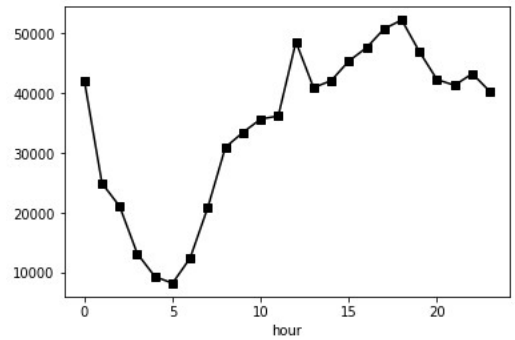
可以发现 SOUTHERN 所占的数量较多，RICHMOND 数量最少。可以推测 SOUTHERN 可能治安或经济水平相对于 RICHMOND 来说较低。

对于 Dates，我选择先将数据的时间格式处理成年、月、日、小时、分钟后进行分析，









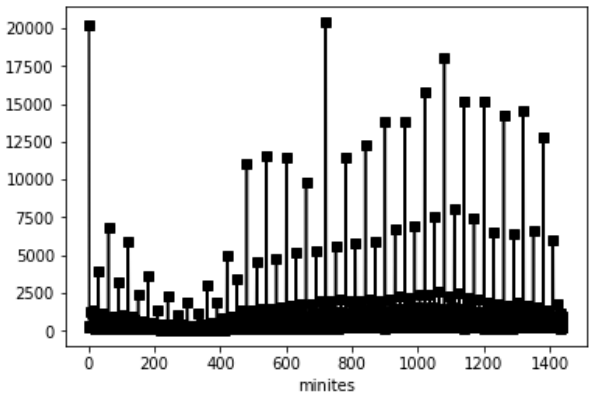
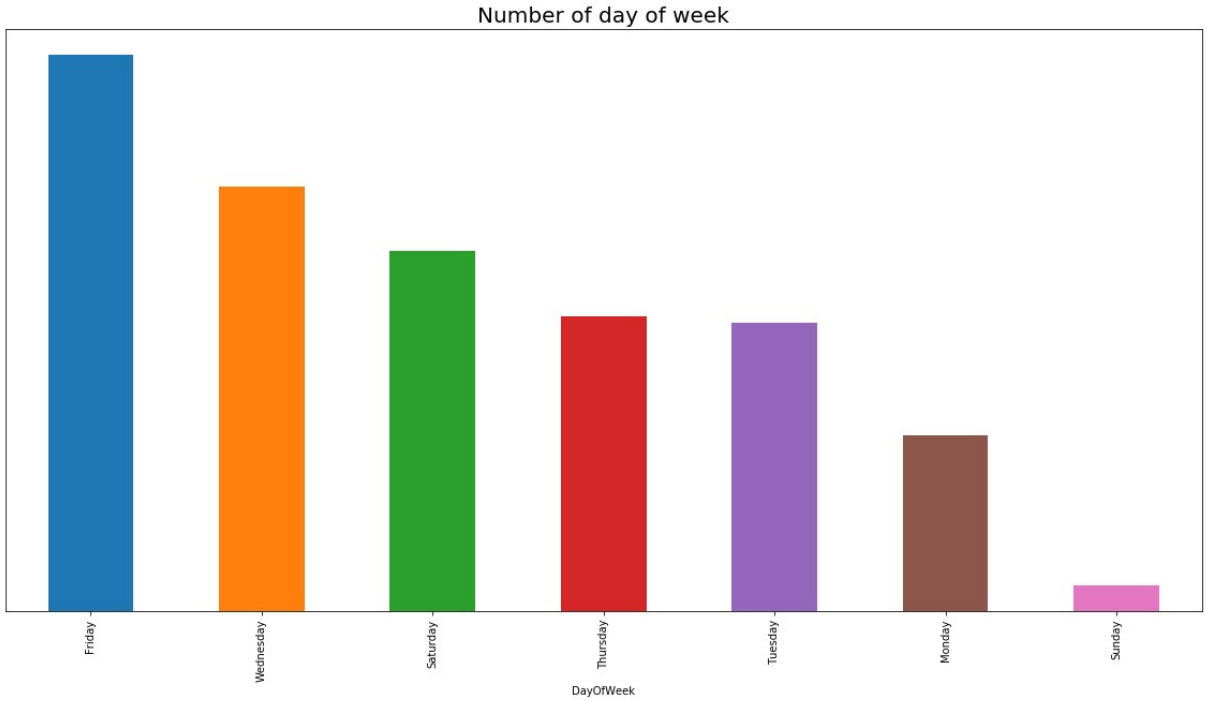


图 4-8：从左至右从上到下分别代表了年、月、日、小时、分钟的犯罪数量统计

可以看出犯罪时间发生的数量符合人的作息规律，发生数量并非是年年都稳定，在 10-11 年之间数量巨减。并且犯罪事件发生符合季节变化，极冷和极热天气会减少犯罪时间的发生。一天之内从中午开始到夜间的犯罪发生数量最多。

接着我们将 DayOfWeek 每周哪天的犯罪数量可视化如下：



看出周五的犯罪量最高，周日最低。

算法和技术

根据之前所述，旧金山犯罪预测是一个多分类样本不均衡问题（MultiClass），输入数据是含有多个维度的犯罪数据，输出是对于目标值的各个犯罪种类的概率值，考虑到数据集数量过大，如果采用欠采样和过采样来解决会导致内存不够。解决多分类问题的算法很多，比如有朴素贝叶斯、KNN、支持向量机、决策树等等，也可以通过深度学习搭建神经网络。

在机器学习中, 多类(*multiclass*)或者多项式(*multinomial*)分类是将实例分配给一个而非多于两个类别的种类(将实例分类给两类中的一个称为二元分类 *binary classification*)。很多分类算法自身支持多于两类的使用, 剩下的就是二元分类算法了, 这就可以通过很多策略去转换成多项式分类器。[6]上述所提到的算法为二元问题的扩展。下面将要描述我所使用的算法，除了 keras 神经网络，其他我使用的都是默认参数，为了来比较不同算法之间的预测精度和所耗时间：

- **Adaboost** 算法是一种提升方法，将多个弱分类器，组合成强分类器。 它的自适应在于：前一个弱分类器分错的样本的权值（样本对应的权值）会得到加强，权值更新后的样本再次被用来训练下一个新的弱分类器。在每轮训练中，用总体（样本总体）训练新的弱分类器，产生新的样本权值、该弱分类器的话语权，一直迭代直到达到预定的错误率或达到指定的最大迭代次数 [8]。
- **朴素贝叶斯**（Naive Bayes）是贝叶斯分类器的一个扩展。假设某个体有 n 项特征（Feature），分别为 F1、F2、…、Fn。现有 m 个类别（Category），分别为 C1、C2、

…、Cm。贝叶斯分类器就是计算出概率最大的那个分类，假设所有特征都彼此独立，也就是求这个算式的最大值：P(F1F2...Fn|C)P(C) = P(F1|C)P(F2|C) ... P(Fn|C)P(C) [9]。

- **决策树**（DecisionTree）是通过一系列规则对数据进行分类的过程。它提供一种在什么条件下会得到什么值的类似规则的方法。一棵决策树的生成过程主要分为以下 3 个部分[10]：

  （1）特征选择：特征选择是指从训练数据中众多的特征中选择一个特征作为当前节点的分裂标准，如何选择特征有着很多不同量化评估标准标准，从而衍生出不同的决策树算法。

  （2）决策树生成： 根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。 树结构来说，递归结构是最容易理解的方式。

  （3）剪枝：决策树容易过拟合，一般来需要剪枝，缩小树结构规模、缓解过拟合。剪枝技术有预剪枝和后剪枝两种。

- **XGBoost**（eXtreme Gradient Boosting）是一个大规模、分布式的通用 Gradient Boosting（GBDT）库，它在 Gradient Boosting 框架下实现了 GBDT 和一些广义的线性机器学习算法。 GBDT（又称 Gradient Boosted Decision Tree/Grdient Boosted Regression Tree），是一个基于迭代累加的决策树算法，它通过构造一组弱的学习器（树），并把多颗决策树的结果累加起来作为最终的预测输出。由于 GBDT 的核心在与累加所有树的结果作为最终结果，而分类结果对于预测分类并不是这么的容易叠加，而是每一步每一棵树拟合的残差和选择分裂点评价方式都是经过公式推导得到的，所以 GBDT 中的树都是回归树。而 xgboost 把树模型复杂度作为正则项加到优化目标中,实现了分裂点寻找近似算法。利用了特征的稀疏性, 数据事先排序并且以 block 形式存储，有利于并行计算。 具体来说 XGBoost 在优化时对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数，并支持自定义代价函数，只要函数可一阶和二阶求导。XGBoost 在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的 Score 的 L2 模的平方和。从 bias-variance tradeoff 角度来讲，正则项降低了模型的 vairance，使学习出来的模型更加简单，防止过拟合。XGBoost 在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间,XGBoost 在训练之前，预先对数据进行了排序，然后保存为 block 结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个 block 结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。

- **LightGBM**：LightGBM 与上述的 XGboost 对比具有以下特点：1、xgboost 采用的是 level-wise 的分裂策略，而 lightGBM 采用了 leaf-wise 的策略，区别是 xgboost 对每一层所有节点做无差别分裂，可能有些节点的增益非常小，对结果影响不大，但是 xgboost 也进行了分裂，带来了务必要的开销。 leaft-wise 的做法是在当前所有叶子节点中选择分裂收益最大的节点进行分裂，如此递归进行，很明显 leaf-wise 这种做法容易过拟合，因为容易陷入比较高的深度中，因此需要对最大深度做限制，从而避免过拟合。 2、lightgbm 使用了基于 histogram 的决策树算法，这一点不同与 xgboost 中的 exact 算法，histogram 算法在内存和计算代价上都有不小优势。在算法上，lightGBM 的预排序算法在选择好分裂特征计算分裂收益时需要遍历所有样本的特征值，时间为(#data),而直方图算法只需要遍历桶就行了，时间为(#bin)；而在内存上，lightGBM 直方图算法的内存消耗为(#data* #features * 1Bytes），而 xgboost 的 exact 算法

内存消耗为：(2 * #data * #features* 4Bytes)，因此 lightGBM 具有明显优势。除此以外，lightGBM 还具有直方图做差加速、支持直接输入 categorical 的 feature 并在 feature 和 data 上都做了并行[11]。因此选择 LightGBM 可以快速有效地进行预测。

基准模型

Yehya Abouelnaga[15]使用随机森林模型训练以后的 log loss 在 n_elements 为 200 时结果为 2.410574498，添加了特征以后 loss 有所下降。使用 xgboost 模型在 max_depth 为 6 的时候训练 log loss 和验证 log loss 分别为 2.57428 和 2.565219931。我最后参考了 kaggle 上的排行榜，我认为模型 log loss 基准为小于 2.6 较好。

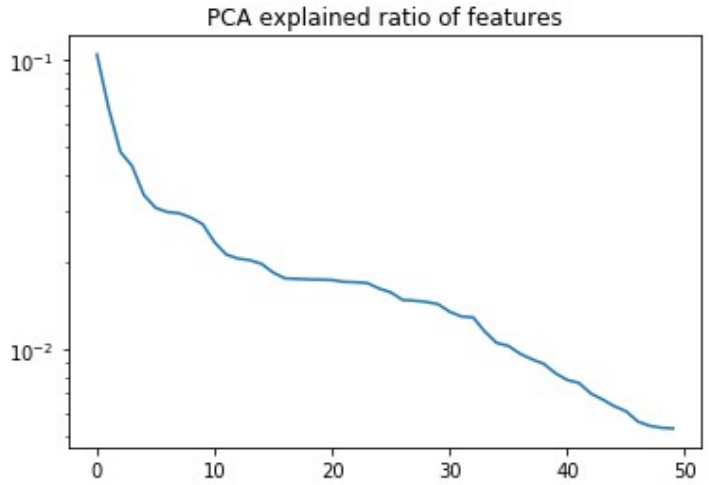
III. 方法

数据预处理

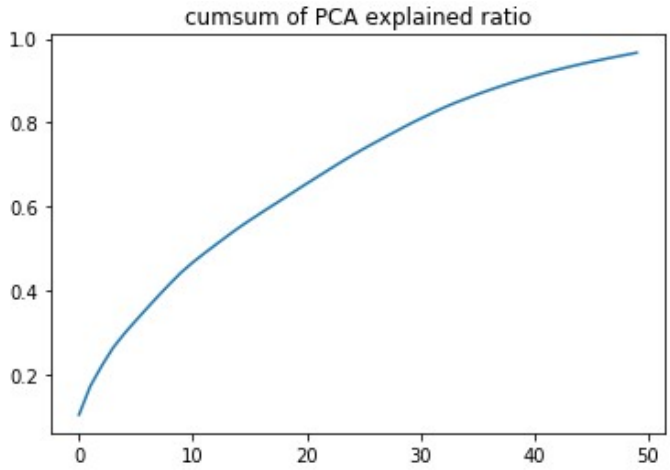
在上一步已经处理完异常值之后，首先要做的就是对特征进行提取。在可视化的时候，我通过 Dates 特征增加了年、月、日、小时和分钟作为新的标签。同时，前面有提到季节对于犯罪数量有影响。因此我根据旧金山的气候进行了季节的划分，将四个季节分别作为训练标签。我没有对数据进行自然语言方面处理，因此我选择删去 Descript，Resolution 和 Address，并将已经处理过的 Dates 删去（后期完善时选择将 Address 再做处理）。

创建完所有特征后，我将目标 Category 从训练集中删除，并将其命名为 target，将出去目标的数据集命名为 features。对于 features 中的数值变量进行了最大最小值标准化处理（**MinMaxScaler**），它通过将每个特征放缩到给定范围内来统一量纲。接下来对 features 所有值利用 get dummies 进行独热编码，完成以后将其进行标准化，转换为均值为 0，方差为 1 的正态分布。接着对 target 进行 Label Encoder，相当于将每个犯罪标签给一个序列号。这些步骤完成以后，我选择加载测试集，进行了上述同样的处理。

接下来在已经产生的 69 个标签中（后期完善将特征从 28 个增设至 69 个），我使用 PCA 进行降维处理，这样可以加快后面机器学习算法处理效率，在压缩数据的同时让信息损失最小化[12]。在几次尝试以后，我选择调整参数 n_components=50，将数据维度从 69 压缩至 50。PCA 关于各个特征的可解释比率如下：



关于累积可解释比率曲线如下，可以看到维度到 22 时对于总和的可累计解释比率几乎接近 1：



最后将 training 与 label 转化成数组，使用 train_test_split 将训练集比例定为 0.9 进行随机洗牌。

执行过程

我使用了 adaboost，Decision Tree，NB，XGBoost 和 LightGBM 来训练数据。起初，我对于前四个算法设置的默认参数来比较不同算法之间所耗的时间和 log loss 大小。

在一开始使用算法进行预测时，我遇到了困难，我发现我不管怎么预测 log loss 都有 200 多。这是很惊人的，因为正常情况下 log loss 也只会在小于 5 的数以内。后来我发现原来是因为上一步数据预处理时，我的训练集和测试集的特征没有对应上，并且测试集比训练集少了 Description 和 Category 以外多了一个标签 id。这对于训练数据并没有任何帮助，应该选择删去。在处理完这个巨大的 bug 之后我的 logg loss 终于恢复正常范围内了。

紧接着我开始记录不同算法之间的耗时和 logg loss，比较差异。在后期的完善中我又通过增加特征工程的方式来提高模型的准确度，以下是 adaboost，Decision Tree，NB，XGBoost 在未增加新的特征工程和增加之后的 log loss 和训练时间表现差异：

表 1：未增加特征工程之前

指标	adaboost	Decision Tree	NB	XGBoost
Log loss	3.579935	25.829218	2.632796	2.457101
Time cost（s）	96.104480	9.207989	0.727246	3028.320346

表 2：增加特征工程之后

指标	adaboost	Decision Tree	NB	XGBoost
Log loss	3.579383	25.071722	2.553726	2.288247
Time cost（s）	200.661457	18.433188	3.094795	5539.385860

通过比较可以发现，未经调参的算法中，XGBoost 的 log loss 最小，但是同事所花时间最长。综合来看朴素贝叶斯和 XGBoost 的表现不错，Decision Tree 的表现最差。

完善

为了达到 kaggle 前 10%的 log loss 为 2.29 的分数,我认为首先可以通过增加特征工程的方式使 log loss 有明显的下降.在初步构建特征工程进行模拟预测时我并没有利用’ Address’特征,结合犯罪对于空间的强依赖性,地址的街道以及数字在分布上会具有一定的特征.例如数据集的’ Address’为:

OAK ST / LAGUNA ST; 600 Block of 47TH AV; 1600 Block of VALENCIA ST…

可以发现数据集的地址特征主要可以归纳为以下两种形式:

- 名称+街道(以/分割的 ST)+/+名称+街道(以/分割的 ST)
- 数字+街区(以 Block of 为特征的街区)+地址名称(部分带有序号)

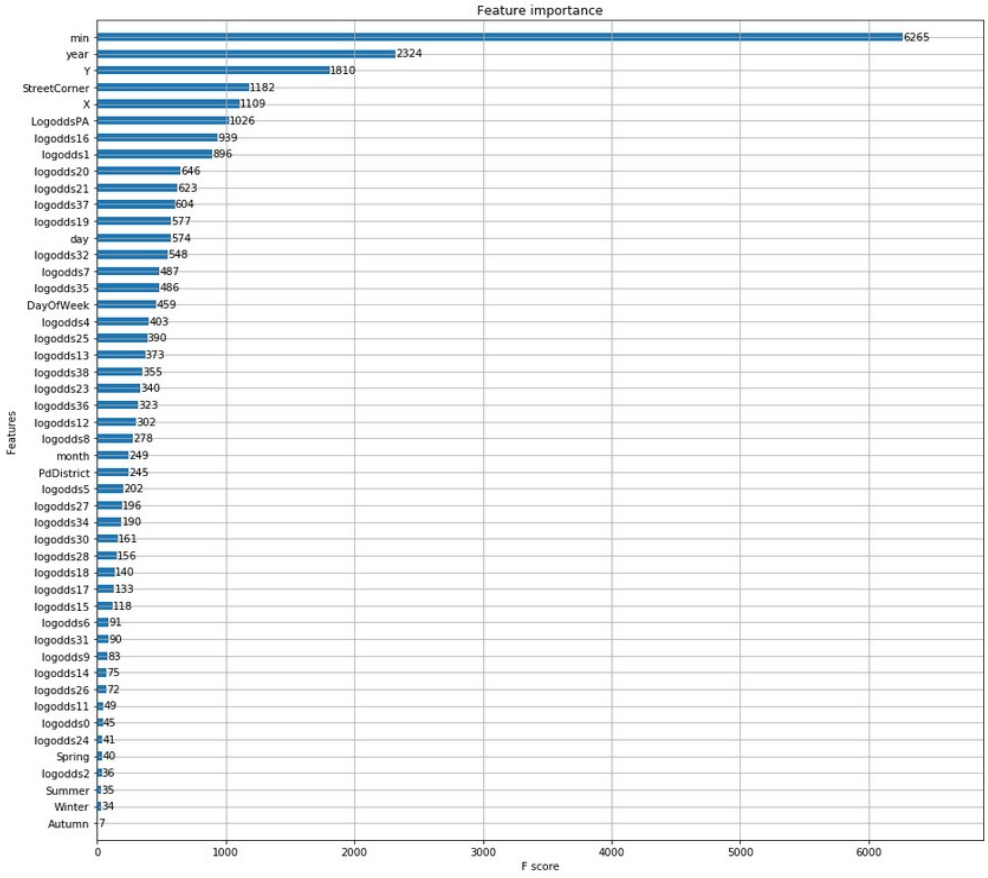
根据这两种特征,参考了 MichaelPluemacher[16]的处理代码,建立了 IntersectionAddress 和 BlockAddress 两个函数处理上述 Address 特征。参考了 papadobic[14]对于不同地址的犯罪数量关系进行的 log 对数处理，并设立的判断 log 值异常的范围，将具有异常数量值的数据单独建立了 logodds 特征。

完成了特征工程的增加以后，开始进行模型调参。首先我选择了 xgboost 模型进行调参，模型的参数选用为：

- learning_rate：学习速率，为了防止过拟合这里是逐渐递减的学习步长，我设立的值为 0.16。
- n_estimators：模型迭代次数，这里我选择的是 20。因为迭代次数越多耗时越长，我希望将 XGBoost 与后面将使用到的 LightGBM 的性能进行比较。
- max_depth：和 GBM 中的参数相同，这个值为树的最大深度。 这个值也是用来避免过拟合的。max_depth 越大，模型会学到更具体更局部的样本。这里我取的值为 10。
- min_child_weight ：决定最小叶子节点样本权重和。一般取值范围是 1~6 之间，我选择的是 3。
- gamma：在节点分裂时，只有分裂后损失函数的值下降了，才会分裂这个节点。Gamma 指定了节点分裂所需的最小损失函数下降值。 这个参数的值越大，算法越保守。在这里我选择的是 0.3。
- subsample：二次抽样，用于训练模型的子样本占整个样本集合的比例。我设置的为 0.8，则意味着 XGBoost 将随机的从整个样本集合中随机的抽取出 80%的子样本建立树模型，数值越大越能防止过拟合。
- colsample_btree：每棵树的列数（特征数），我选择的是 0.8。
- objective：预测的类型，这里需要选择多分类且返回各个类别概率值的类型，为'multi:softprob'。
- scale_pos_weight：如果取值大于 0，在类别样本偏斜时，有助于快速收敛。在这个项目中有类别不平衡的问题，因此设置值为 1。

- random_state：随机状态值，我设置的为 27。

模型的评分标准为 log loss 值大小，拟合后模型预测精度由一开始的 3.21611 变为 2.32303，在二十次拟合以后模型所花时间为 4231.442037 秒。可以看到模型所耗得时间还是很长的。最后将 XGBoost 模型中各个标签的特征重要性如下：



XGBoost 模型我认为训练时间过长，并且 log loss 下降的过程非常缓慢，可以使用其他模型预测进行比较。

于是我便开始尝试其他的模型，在这里我选择使用 LightGBM 作为改进模型，首先是对它进行调参，各个参数含义如下：

- task:'train',配置目标是用于训练，所以选 train
- boosting_type: 'gbdt'，设置提升类型也可以理解成训练方式。
- objective: 'multiclass', 因为我们的目标是预测多分类，所以为 multiclass。
- metric: {'multi_logloss'}，依据 kaggle 的评分标准，评估函数为多分类的 log loss。
- num_leaves: 40，叶子节点数，一般默认值为 31，我设置的是 40，num_leaves 不超过 max_depth^2 即可。
- learning_rate: 0.05，学习速率，如果验证数据的一次度量在最近的 early_stopping_round 没有提高模型就会终止训练，在这里我使用的是 0.05.
- feature_fraction: 0.9, 建树的特征时所选择比例。

- bagging_fraction: 0.8, 每次迭代时建树的样本采样比例。
- bagging_freq: 5，k 意味着每 k 次迭代执行 bagging。
- Verbose: 1, 0 为显示致命的, =0 显示错误 (警告), >0 显示信息，这里选择为 1。
- num_class: 39, 多分类时所需分类的数量，我们需要预测的是 39 种。
- max_depth: 30, 限制树的深度。
- min_data_in_leaf: 3，将它设置为较大值时可以防止过拟合，但是太大会导致欠拟合。
- early_stopping_round：设置模型在多少回合以后 loss 没有提高就停止拟合，这里选择的是 10。
- num_boost_round：训练回合我设置的是 300，这个值设置没有上限，因为如果模型在某 10 个回合内 loss 并没有下降，那么模型会终止拟合。

最后模型拟合的结果 log loss 为从 3.52264 下降到 2.23303，总共耗时 955.710346 秒，相比于 XGBoost 效率非常高，而且 loss 下降也很明显。

IV. 结果

模型的评价与验证

我最终选择的模型还是 LightGBM 模型。模型所训练的用时相比 XGBoost 要少很多，而且 loss 精度也高于 XGBoost。

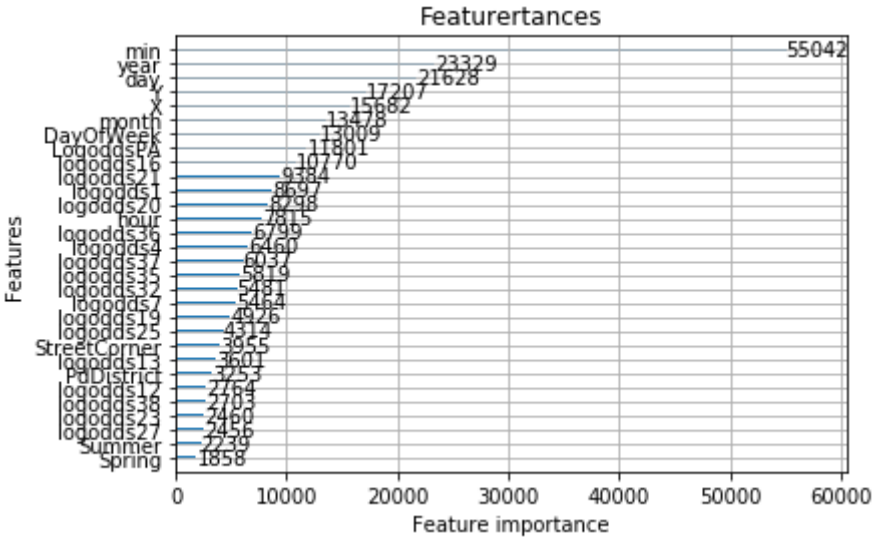
合理性分析

模型最终上交至 kaggle 的得分小于 2.29。参照之前的基准模型，这个结果我认为较为理想。对于地址 Address 的处理可以帮助我很快速的提高模型预测的准确度，而 LightGBM 的使用也很大增加了预测效率。

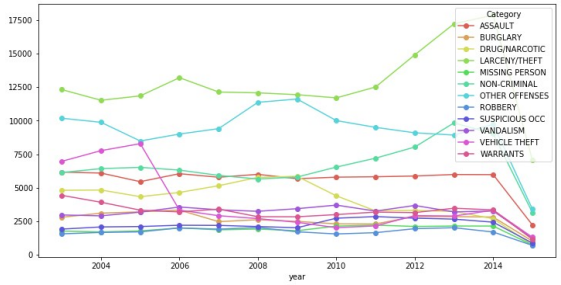
V. 项目结论

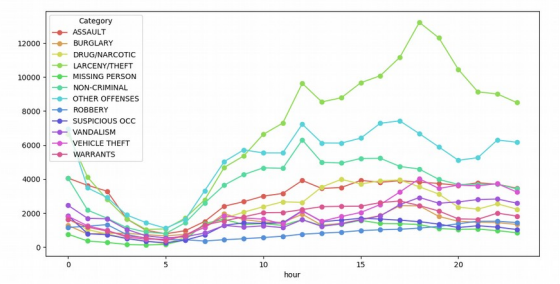
结果可视化

对于之前使用的 LightGBM 模型进行特征重要程度可视化：



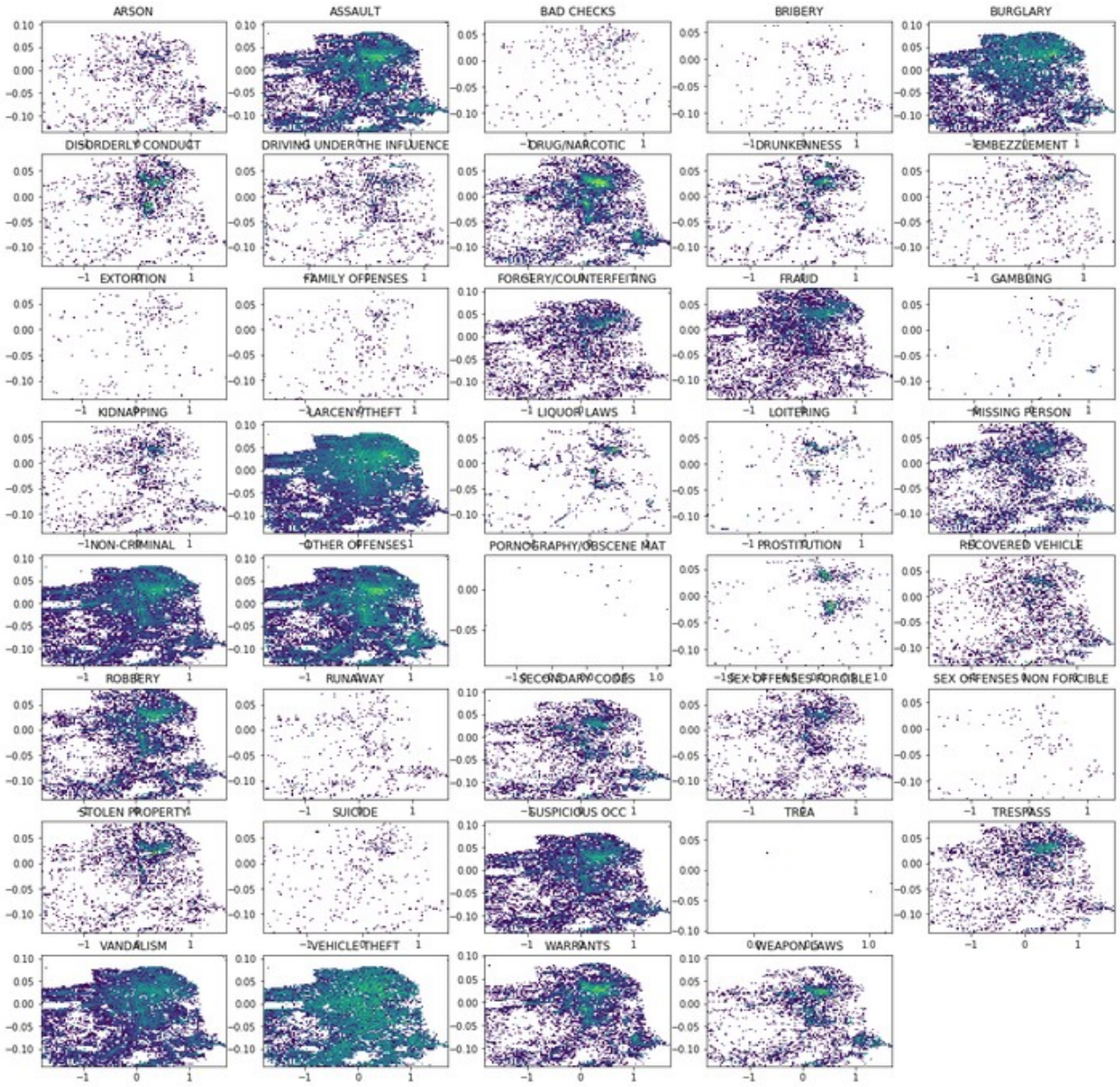
与 XGBoost 的主要重要特征的列举比较相似，但是在特征的重要性排名上还是略有差别的。可以发现分钟、年份、纬度、经度、天数、星期数和部分 logodds 这几个特征对于预测的帮助很大。充分证明了犯罪具有很强的时空依赖性。犯罪发生的经纬度在地理上符合旧金山区域的特征。在时间上，结合之前的可视化：





可以发现很多犯罪在季节和年份、时间上具有时间的依赖性。

参考了 papadobic[14]的部分代码，我对各个犯罪事件的空间热点进行了可视化如下：



可以观察出很多犯罪，例如：ASSAULT，BUROGLARY 等等的犯罪事件具有很强的空间依赖性。印证了之前对于各个特征的排名。

对项目的思考

在完成整个项目之后，我的流程如下：首先对数据的类型、格式进行初步了解，接下来通过可视化的方式进行新的特征的创建，完成特征工程之后对数据进行预处理，然后使用不同的算法比较其中的性能和预测精度差异，选择一个作为最终的模型，并在此基础上进行调参优化进行模型完善。

项目有意思的地方对我来说可能就是数据的特征，与普通的数字特征不同的是，这个数据集中夹杂了时间序列、文本内容。所以在数据的处理和特征工程的搭建上更具有挑战性和趣味。

项目比较困难的地方对我来说就是在选择了最终模型之后对于模型的完善。我想试着提高模型预测的准确度，然而这对于我来说并非一件易事，在网站上学习了很多调参优化的方法发现对于 log loss 的提高效果还是很显著的，还看到有的参赛者选择将 X 和 Y 进行三角变化[17]，并且发现有利于模型预测这个

操作让我联想到了 CNN 进行图像处理时将图片进行旋转对称等等的做法，感觉两者具有相似之处，这也给我后期对特征工程的处理产生更多的思路。

需要作出的改进

以考虑尝试其他算法，如 RNN，LSTM 等等来解决具有时空依赖性的犯罪事件预测问题或者最直接的就是增加特征工程。另外，模型的分类属于类别不平衡问题，关于这个问题我没有很好的去解决它。如果使用欠采样和过采样的策略会增加数据集的大小，超出内存，同时使原始数据信息的准确性有所下降。

可以尝试在上一部分所说的对于经纬度进行一些数学处理，增加模型的预测精度。

[1]https://github.com/udacity/cn-machine-learning/tree/master/sf_crime_classification
[2]https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%8C%96%E6%8E%98/216477?fr=aladdin
[3]https://www.kaggle.com/c/sf-crime/data
[4]https://blog.csdn.net/huacha__/article/details/81057150
[5]https://www.cnblogs.com/zhaokui/p/ml-metric.html
[6]https://blog.csdn.net/qq_27009517/article/details/80264919
[7]https://classroom.udacity.com/nanodegrees/nd009-cn-advanced/parts/635eb098-3c27-4f56-b9ff-45e26ddd2d31/modules/11dfb4a3-14fe-47df-825a-ff66cc09bd42/lessons/2839b5c9-e4e1-4c7f-9d80-0c210aa0adf2/concepts/7403e2b2-5494-4f90-bf1b-4716b687cd73
[8]https://blog.csdn.net/fuqiuai/article/details/79482487
[9]https://blog.csdn.net/fuqiuai/article/details/79458943
[10]https://www.cnblogs.com/sxron/p/5471078.html
[11]https://blog.csdn.net/zwqjoy/article/details/82150528
[12]https://blog.csdn.net/yimixgg/article/details/82894578
[13]https://keras.io/zh/models/sequential/
[14]https://www.kaggle.com/c/sf-crime/discussion/15836
[15]Abouelnaga, Yehya. (2016). San Francisco Crime Classification.
[16]https://github.com/MichaelPluemacher/San-Francisco-crimes
[17]https://www.kaggle.com/c/sf-crime/discussion/18853