学　　院　　软件学院

课　　程　　数据挖掘

小组成员　　孙蒙新、许宇秀、张晓颖

2018 年 6 月 27 日星期

目录

1. 比赛过程汇总

1.1. 比赛重要结果

经过进 2 个月的努力，我们小组的最好成绩是 0.11062， 排名是 100/2749，比赛结果提交次数和比赛过程如表 1-3 所示。

表1. 比赛结果提交次数表

	有效提交次数	无效提交次数	总次数
孙蒙新提交次数	29	42	71
张晓颖	29	39	68
许宇秀	25	36	61
团队提交总次数	83	117	200

表2. 比赛的重要事件时间表

时间	主要内容
2018.4.3	下载数据，简单查看数据属性和意义，用直方图、饼图等可视化工具检测了线下数据的特性。
	测量线上数据的属性，包括：每个属性的缺失值、最大最小值、中位数、所有数据的总量
	针对每个属性的缺失值预处理
	基于用户的粒度上，使用 callstate 和 speed 第一次获得正的 GINI 数值。
2018.5.24	基于用户的粒度上，做更多的特征工程一直超时。
	听从老师的建议，转到基于 trip 的粒度上，结果一直超内存。
	基于 trip 的粒度上，线上提交结果为-1（预测结果错误过多直接无效）
	基于 trip 的粒度上，尝试更多的特征工程和模型，一直无效。
	基于用户和基于 trip 的特征工程结合，依然无效。
2018.5.26	转到基于用户的粒度上，重新开始特征工程。

1.2. 数据详情

针对比赛方给出的数据，我们分析了不同属性的最大值、最小值、均值、标准差和丢失率等情况，具体情况如 4。

表3. 数据详情表

	均值	标准差	最小值	最大值	丢失率
经度	115.55	6.48	-122.25	145.76	1.23（万分）
维度	31.594	6.1	1.28	55.66	0.32（十万分）
海拔	122.475	316.84	-450	5916	0.921（万分）
速度	9.18	8.11	2.14	258.434	8.44（万分）
电话状态	——	——	——	——	60.7%
非零 Y 值	5.476	13.02	0.025	348.878	

结合数据，我们发现的初步想法是：

针对 unix 时间戳：因为驾驶事故跟出行时间段有关系，可以分析早上/中午/凌晨不同时间段的赔率

经度/纬度/海拔：可以考察驾驶路段地点，但具体状况需要映射到具体的地图分析。

方向：暂时看不出。

速度：重点关注行驶速度快的数据，因为速度快更容易出驾驶事故

电话状态：通话状态若为连通容易使驾驶员分心导致事故，但数据中有将近 40000 条为未知的电话状态。因此进一步我们对各个电话状态的赔付情况进行分析，电话状态为未知/断连/连通的赔付率较高。

1.3. 初步的聚类分析

我们小组运用 Kmeans 算法，主要基于一份较好特征聚类计算类中心点相似度进行分析

1.4. 数据存在的问题

- TRIP_ID 和时间不匹配，需要按照时间重新分配 TRIP_ID
- 经纬度可能大小超出中国的范围，需要处理异常值
- 速度存在-1 的缺失值
- 高度存在超过中国最高城市高度
- 对于 Y 值：由于存在极少的 Y 值过大，对模型训练有碍

1.5. 数据预处理

针对 1.4 章节提出的问题，我们采用以下方式清洗数据：

- 使用速度均值填充速度缺失值
- 修改电话状态为打电话和没打电话两种状态
- 选择在经度 73-135 和纬度 20-53 之间的数据
- 选择在 6700 海拔以下的数据
- 高度归一化
- 速度归一化
- 按照用户 ID 和 TRIP_ID 排序
- 最大的 Y 值改成比第二大的 Y 值大 10 的数值

最后，针对有赔率用户和没有赔率用户的数据量差距非常大的情况，我们采用分割训练样本的方式。将 train 数据分割多次，每次取和有赔率用户数量大致差不多的数量，将有赔率和没有赔率的用户一起送进分类器学习。

1.6. 特征提取

在特征提取中，我们主要参考了附件 1-4 的文件，从中提取了 20 种类型的特征，具体如表 所示，大部分的特征可同时基于用户和基于 trip。

表4. 特征表

序号	特征名称	特征备注
1	用户连续驾驶最长时间	时间间隔为 10 分钟的行程都是连续驾驶
2	用户的总行程	用户所有数据的行驶总路程
3	用户的平均行程	用户行程总路程/用户 trip 数量
4	用户是否经常疲劳驾驶	凌晨 0 点-3 点，下午 5 点-7 点 是疲劳驾驶
5	用户急转弯次数	转弯角度>90 度是急转弯
6	用户行驶上坡的比例	
7	用户行驶下坡比例	
8	平均速度，最大速度	
9	高度，速度差值	每分钟的高度和速度差值
10	行驶地点距离北京、广东、西藏和乌鲁木齐的距离	
11	上坡路程占行程总长的比例	上坡路程的比例
12	下坡路程占路程总长的比例	下坡路程的比例
13	下坡时间占行程总时间的比例	下坡时间比例
14	紧急刹车	速度超过 30 的停车是紧急刹车
15	快速起步	速度超过 30 的汽车启动时快速启动
16	行程的停顿次数	行程中速度为 0 的次数
17	打电话的总次数	
18	打电话时的平均速度	
19	打电话的最大速度	
20	左拐次数	转弯角度没有限制

21	右拐次数	转弯角度没有限制
22	打电话的左拐次数	打电话时左转的次数
23	打电话的右拐次数	打电话时右转的次数
24	打电话时上坡次数	
25	打电话时下坡次数	
26	打电话时平均速度、最大速度	

1.7. 特征评估

特征评估有以下三种方式：

- 利用 traindata 检验。 将新添加特征所在的 train 数据放进模型预测出预测值，求 traindata 的预测值与 Y 值的方差
- 利用现成的特征选择方法。直接利用 sklearn 工具包中已有的特征选择方式评估特征价值，获取每个特征对模型的贡献度。
- 反向检验。将新的特征值作为标签值，原有的特征值和 Y 值作为特征训练分类器，得出新特征与 Y 值的相关系数值。

1.8. 实验结果与分析

1.8.1. 聚类结果

在原始数据上的聚类结果如图 1-3 所示。结果发现原始数据样本差异很大，聚类无法得到有用结论。

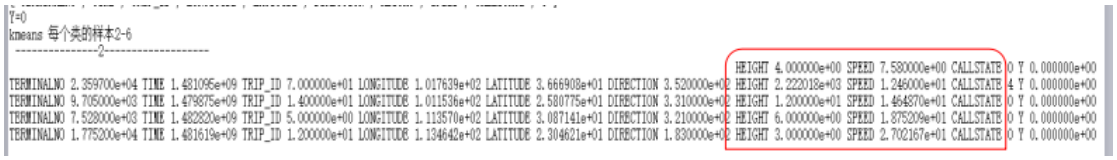

图1. Y=0 用户的数据样本

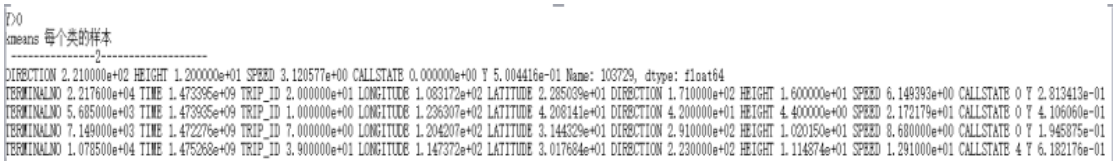

图2. Y!=0 用户的数据样本

在归一化后的数据上聚类，结果如图 3-4 所示。结果发现有些特征具有较强的关联性。

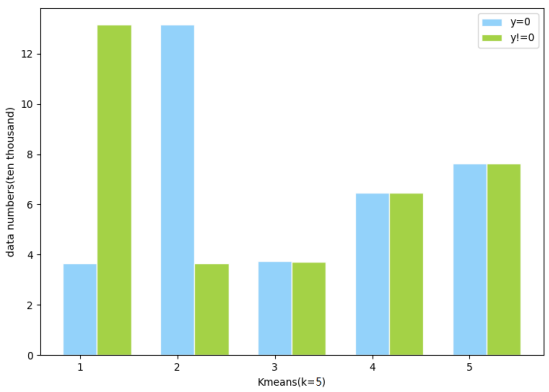

图3. 聚类后每类样本的数据量分布图

Table3 similarity

Y!=0 Y=0	0	1	2	3	4	5	6	7
0	0.8803	0.9828	0.9383	0.8634	0.9005	0.7857	0.8665	0.9850
1	0.9999	0.7831	0.9910	0.9986	0.9992	0.4136	0.9993	0.9507
2	0.9962	0.8289	0.9984	0.9922	0.9991	0.4819	0.9934	0.9720
3	0.9962	0.7238	0.9743	0.9987	0.9913	0.3318	0.9983	0.9180
4	0.3852	0.8747	0.5092	0.3566	0.4250	0.9997	0.3595	0.6530
5	0.9978	0.7351	0.9784	0.9990	0.9938	0.3455	0.9992	0.9252
6	0.9870	0.8675	0.9997	0.9800	0.9932	0.5424	0.9819	0.9867
7	0.4440	0.9035	0.5636	0.4163	0.4825	0.9991	0.4191	0.7001

图 4. 特征之间的相似度

在提取的特征数据上做聚类，结果如表 5 所示。结果发现 y=0 时打电话状态、疲劳驾驶状态、海拔高度等的特征的中心点数值平均值（出去标黄色的，将这些点认为是奇异点）与有赔付率的 y 中这些特征的数值有很大差异，所以可以初步把这些特征判断为与是否有赔率有直接关系。使用这种分析方法的不足在于由于数据集数目很大，需要增加聚类的还有特征

的数目。

表5. 特征聚类结果表

Y==0					
CALLSTATE_SUM	CALLSTATE_MEAN	SPEED_MEAN	SPEED_MAX	TIME_MEAN	'HEIGHT_MEAN'
0.0002166884	0.0024359703	0.0383747733	0.0694286848	0.0015513099	0.0000878197
0.0005508682	0.0063823664	0.0514215167	0.0907393722	0.9936570767	0.0425335690
0.0013447634	0.0083021066	0.0572125024	0.1123554535	0.5032136802	0.0515186591
0.0303624278	0.6985050793	0.0441697682	0.0800101549	0.0040891327	0.0002627430
0.0006311997	0.0039650281	0.0783278797	0.1348795489	0.0037101119	0.0004302934
0.0137854979	0.2265190831	0.0414449363	0.0870049547	0.0024989243	0.0001665742
0.0196359346	0.4437391467	0.0432719027	0.0824768436	0.9916656257	0.0346521643
平均值（除掉标黄色的数值，将这些数值视为奇异值）					
0.000466252	0.005271368	0.052868126	0.093842145	0.00296237	0.000286537

Y != 0						
CALLSTATE_SUM	CALLSTATE_MEAN	SPEED_MEAN	SPEED_MAX	TIME_MEAN	'HEIGHT_MEAN'	Y
低赔率的 Y 值						
0.0013054113	0.0199659219	0.0503520324	0.0924397588	0.2692545282	0.0126645113	0.7626050141
0.0013572042	0.0195492332	0.0502443102	0.0919375913	0.2736310920	0.0128849714	3.8031579326
0.0011503919	0.0159926108	0.0514671541	0.0941678925	0.2760219126	0.0132909454	10.640000996
低赔率的 Y 对应的特征平均值						
0.001271002	0.018502589	0.050687832	0.092848414	0.272969178	0.012946809	5.068587981
高赔率的 Y 值						
0.0014089468	0.0246591587	0.0513270840	0.0944437565	0.2778676125	0.0113074632	23.5585693426
0.0007970141	0.0103631848	0.0553813204	0.0975280728	0.2569435665	0.0114085908	42.4741549516
0.0010361074	0.0142502315	0.0481948359	0.0900473585	0.2564209386	0.0106447152	65.6133772251
0.0012348192	0.0243571253	0.0426208926	0.0828266390	0.2530833501	0.0087571487	102.981508643
高赔率的 Y 对应的特征平均值						
0.001119222	0.018407425	0.049381033	0.091211457	0.261078867	0.010529479	58.65690254

1.8.2. 部分实验结果

每次实验都是在一份效果较好的实验结果上增加几个特征，测试新特征的效果。表 6 为部分实验结果。

1.9. 脑洞

- 用聚类，根据距离做 Rank。首先根据对训练数据中进行聚类，根据数据的数量级对所聚的类的数目进行确定，比如说这次比赛的数据有几百万条数据，可以聚成个几百类，然后根据在慢慢根据自己的判断和理解进行类的融合，最后融合到几十个（比如说 30 个）可以明显划分出是否有赔率的聚类，计算出聚类的类中心点。当有一个新的数据到来时，可以根据 K 近邻进行判断，当 K=1 时，计算最近的距离所对应的类，如果是有赔率的，离中心点越近，赔率越高（可以用公式），如果是没有赔率的，力中心点越远，赔率越高。
- 首先根据对训练数据中进行聚类，根据数据的数量级对所聚的类的数目进行确定，比如说这次比赛的数据有几百万条数据，可以聚成个几百类，然后根据在慢慢根据自己的判断和理解进行类的融合，最后融合到几十个（比如说 30 个）可以明显划分出是否有赔率的聚类。然后对最后的这几个聚类中类中心进行比较，比较出有赔率和没有赔率区别。
- 将数据集分别用不同的模型进行训练，训练出来的结果当成的新特征，统统放进去原数据集，然后对新形成的数据集用某一种模型进行预测分析。

2. 学到的内容

2.1. 代码书写

2.1.1. 内存优化

在比赛中，由于刚接触 python 语言和 dataframe 工具包，很多的编程基础知识和技巧完全不懂，加上比赛时间要限制在 6 个小时内，所以代码的优化非常重要。代码优化主要分为时间优化和内存优化。

时间优化 15 条总结：

- 在所有优化技巧中，最重要的是代码的逻辑优化。即是相同的效果，使用不同的获取方式，可以大幅度节约时间代价。
- 所有数据的读取、使用、保存尽量使用 dataframe。
- 使用 dataframe 的 groupby 代替一层 for 循环，group 函数可以获取分组的数据，每个分组中包含一层数据，相当于是一层 for 循环。
- 使用 dataframe 的 groupby 和 apply 代替两层 for 循环，对数据使用 groupby 函数后，获取的是<pandas.core.groupby.SeriesGroupBy object at 0x04005770>类型的数据，这个数据包含每个分组的数据，在使用 apply 函数对每一组的每个元素进行操作。Apply 支持使用 python 所有现成的函数，也支持 apply 函数，通过自定义函数可以快速遍历每个元素并对每个元素执行相应的操作。
- 使用 copy，少使用 deepcopy，尽管这两个函数都是复制对象函数，但 deepcopy 是递归复制，效率较低。
- 使用 dict 或 set 查找元素，python 的 dict 和 set 都是类似 hash 表的实现，查找元素的效率为 O(1)，查找效率非常快，所以尽量使用 dict 和 set 查找。
- 循环之外的能做的事情要放在循环外面。
- 不借助中间变量交换变量的值。使用中间变量时，python 要重新生成变量，复制变量，所以效率很低。
- 使用 if is true/false 来真假，不要使用==。
- 使用**代替 power。
- List 中装的是 index 时，想要获取上一个索引和后一个索引时，先将 list 转为 numpy，直接对 numpy 对象加一或者减一即可。
- 使用 map(list)的方式对处理 list 的每个元素
- 在并行计算中，dataframe 的 trunk 可以节约时间，但是如果不是并行结算的话，trunk 更浪费时间。
- 使用 timeit 函数查看每个函数调用的次数和运行时间，查看性能瓶颈，优化代码。

内存优化的 3 条总结：

- 减小数据的精度。一般 dataframe 对整型数据默认的精度是 int64，浮点型数据是 float64，也就是说 dataframe 默认对数据使用最大精度。这是非常浪费内存的，所以要把 dataframe 的属性设置成合适的精度。注意精度设置要根据数据数值范围设置，如果精度太小，会使数据直接变成 inf。
- 随时使用 del 删除不必要的对象
- 多使用生成器而不是迭代，xrange 一次只返回一条记录，range 一次返回所有的记录

内存优化一般都涉及到内存管理等底层知识，所以想完全弄清楚这个就可能要专门研究了，所以这里不做过多介绍。

2.1.2. Dataframe

- 通过 list、numpy 和字典创建 dataframe
- 查看数据：dataframe.head（头），dataframe.tail（尾）,dataframe.describe（数据的统计汇总），dataframe.column（列）, dataframe.index（索引），dataframe.values（值）
- 数据的选择:datafame[‘column_name’]，返回一行 series 值，dataframe[start:end] 返回数据切片,dataframe.loc[row,column]返回数据切片
- 数据的选择:dataframe[(datafame.A > 0) & (datafame.B > 0])]，
- data[dara[‘column_name’].isin([A,B,C,D])]
- 数据的设置：通过列和 loc 函数设置数值

- 处理缺失值：fillna 和 dropna
- 直方图可视化:dataframe.hist
- 数据的合并：merge（类似数据的连接）、concat（基于轴的拼接）

2.2. 模型使用

2.2.1. 决策树

最开始，我们使用了 python 的 sklearn 工具包的回归决策树模型，并且使用默认参数。通过使用决策树模型，我们发现决策回归树具有以下几点优点：

- 简单易用，而且输出的结果易于解释，树能够被图形化，加深了直观的理解。
- 几乎不需要对数据进行预处理，可以直接使用未经过预处理的数据。
- 模型的时间开销很小，而且模型一旦建立，对于未知的样本预测特别快。

同时决策树也有缺点：

- 决策树模型容易产生一个过于复杂的模型,这样的模型对数据的泛化性能会很差。这就是所谓的过拟合.一些策略像剪枝、设置叶节点所需的最小样本数或设置数的最大深度是避免出现 该问题最为有效地方法。
- 决策树可能是不稳定的，因为数据中的微小变化可能会导致完全不同的树生成。这个问题可以通过决策树的集成来得到缓解
- 如果某些类在问题中占主导地位会使得创建的决策树有偏差。比赛数据中有赔率和没有赔率的样本数量差距悬殊，所以决策树对不平衡的数据十分不友好。所以比赛结果不太好。大约是 0.06。

2.2.2. Xgboost

和同学讨论之后，我们改换成 Xgboost 模型。Xgboost 和 GBDT 是竞赛和工业界使用都最频繁的几种模型之一，能有效的应用到分类、回归、排序问题。虽然使用起来不难，能够理解参数的意义并且熟练，调参的过程对刚刚接触这个模型的我们还是有些难度的。

参考了网上的几篇博客：

Xgboost 原理 - https://blog.csdn.net/sinat_22594309/article/details/60957594

调参过程 - http://www.cnblogs.com/harvey888/p/7203256.html

调参过程 - https://blog.csdn.net/han_xiaoyang/article/details/52665396

调参经验 - http://www.cnblogs.com/mfryf/p/6293814.html

针对比赛，我们尝试调整了一下几个参数：

reg：学习目标

eval_metric：评估指标

max_depth：树的最大深度

min_child_weight：孩子节点中最小的样本权重和

max_leaf_nodes：叶子节点数量

subsample：随机采样的比例

scale_pos_weight：调节正负样本不均衡问题的参数

但是不知道是模型已经达到最优效果还是参数调整方式不对，调完参后的模型效果没有任何变化。

2.2.3. LightGBM

LightGBM 算法是一个轻量级的 GBM 梯度提升机算法。最大的特点是耗时短、内存少、准确率高。这个算法也是这次比赛中第一次接触到的算法。

LGB 原理 - https://blog.csdn.net/niaolianjiulin/article/details/76584785

LGB 参数速查 - https://blog.csdn.net/weiyongle1996/article/details/78446244

LGB 的参数和 Xgboost 的参数很相似，所以我们按照调整 Xgboost 的方式来调整 LGB 的参数。结果发现效果提升了 0.03。

2.3. 模型融合

在比赛的最后阶段，我们尝试将基于两份完全不同的结果进行融合，但是实验结果比较差，分数在 0.02 左右，原先的每份代码的分数在 0.11 和 0.14。两份代码采用了不同的数据预处理和特征，模型融合是将不同用户的预测值相加取均值结果作为最后输出。

3. 暴露出的问题

3.1. 数据观察和清洗不充分

由于是第一次正式参加比赛，刚拿到数据时完全不懂得如何处理数据，更没有数据预处理的想法。即使老师在课上讲过数据一定要预处理，但是我们做的时候依然没有要预处理的想法，只是简单的看了一下线下数据的最大值、最小值和数据分布，用可视化工具简单的画了一下数据，但是这些可视化结果对我们没有帮助，我们没有结合比赛方给出的数据范围确定错误数据和异常数据。同时，另一个更大的问题是我们没有检查线上数据的分布、线上数据和线下数据的区别。这一点对我们以后处理的数据具有很大的警示意义。

在数据预处理之后是数据清洗，但是我们组内成员对于不同属性的清洗方法有不同的意见，每个人都有自己的看法，这时就没有一个科学合理的方式来确定到底哪一个方式是好的。之后是老师给出了建议。从老师给出建议的方式，我学到了几种判断策略。①缺失数据比例小就删除。如果缺失数据只占总数据很小的一部分的话，可以直接删除。这点使用必须要建立在全面确定所有数据的数量，不能自己以为数量少就随意删除，小部分比例应该是百万分之一或者是更小。②根据属性的实际意义使用合适的值填充。对于高度和速度等在一段时间内连续的属性来说，就直接用某段时间内的平均值来填充就好。对于电话状态，一般默认为开车中打电话的人是少数，大部分人开车是不打电话的，将电话状态填充为 0(未接通状态)就好。

同时我们参考了几篇博客：

https://blog.csdn.net/lujiandong1/article/details/52654703

最终，我们决定使用哪种预处理方式是根据比赛方的计算条件限制和属性意义来确定。结果证明预处理操作使结果提升了 0.01 左右。

3.2. 数据归一化的疑惑

我认为归一化操作是消除数据数值对结果的操作，所以我是在预处理之后将速度和高度归一化。这种处理方式没有使比赛提交的结果更好。但是其他成员在预处理时没有做归一化操作，选完特征工程之后，在基于用户的粒度上进行归一化，这种方式使线上比赛结果提升了 0.02。这种使用归一化的方式使我很疑惑，到底应该怎么使用归一化操作？

4. 值得肯定的习惯

4.1. 写比赛记录文档

在比赛过程中，每天都有各种关于比赛的想法，为了记录想法和确定想法是否有用，用电子文档的方式记录想法。同时还记录比赛过程中当前存在的问题和可能的原因，和不同解决思路。因为做比赛是一个不断尝试的过程，而且我们没有经验，无法确定哪种想法最好，只能不断尝试所有想法，依次淘汰不合理的想法，所以就需要使用文档来记录过程。记录文档可以使我们理清现在的问题和解决思路还可以帮助我们相互交流，不同的成员之间可以相互参考文档相互学习，避免重复叙述浪费时间。而且文档还可以为总结提供文字资料。

4.2. 写技术学习文档

因为我们对 python 非常不熟练，很多基本的命令都需要现查，而且在写代码的过程中，可能当时用了会了，但是过会就忘记了，所以我们用文档的方式记录在写代码中遇到的所有问题。从最开始的 list 和 array 的基本使用到后面的内存管理每一点都记录在文档中，刚开始几乎是每次写代码要查很久、查完之后要写到学习文档中，第二天复习第一天的学习知识，到后来几乎不需要查命令，可以毫无障碍地代码，这么快的转变都是因为使用了学习文档。这次的经历也使我们明白了学习一个新的编程语言最好最快的方式就是收集自己的不会的内容行程自己的学习文档。

4.3. 周报记录个人周进展

比赛过程中，我们每个使用了不同的方法处理数据，但是不同方法之间存在相同的地方，所有我们文档的方式记录每个人已经做的事情，这样大家可以相互参考。同时也为了区分小组成员的工作量，监督每个人做一些事情。

表6. 部分实验结果表

序号	特征	数据粒度	模型	实验与分析	改进方向
1	10 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向	用户	决策树	分数：0.03921 这份数据是早期结果中较好的一份，以这份为基础，加入新的特征寻找有效特征	a.数据处理可以考虑 PCA 降维处理看看结果如何 对训练集中 Y 最大的奇异值做处理 b.替换训练数据中的速度大于 200 为 15，因为平均速度在 15 左右
2	12 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向、急转弯、方向连续变换（基于电话和速度）	用户	决策树	分数：0.04469，急转弯和方向连续变换有助于结果的提升	尝试一下用改用户的速度和方向的平均值和中位数来填补缺失值，测试一下效果，如果还是差不多的话建议不用对异常值进行处理
3	18 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向、急转弯、方向连续变换（基于电话和速度）、总路程（基于经纬度）、平均路程、不安全停车速度数量占所有行程比例、不安全停车速度数量占所有路段比例、不安全起步数量占所有行程比例、不安全起步数量占所有路段比例	用户	XG	分数 0.04229/0.03241，总路程（基于经纬度）、平均路程跟原有结果差不多，暂时保留该特征；加入不安全停车速度数量占所有行程比例、不安全停车速度数量占所有路段比例、不安全起步数量占所有行程比例、不安全起步数量占所有路段比例使结果下降，可能这类属性太宽泛了，不能对预测起作用。	a.海拔高度的缺失值用用户经常出现的区域填充 b.增加每个用户的训练数据 c.有赔率用户和没有赔率用户单独训练 d.考虑基于 trip 记录进行训练和预测。
4	16 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向、急转弯、方向连续变换（基于电话和速度）、总路程（基于经纬度）、平均路程、驾驶最长时间、长途判断	用户	XG	分数：0.0276，连续驾驶最长时间、长途判断，这两部分都只针对每个用户最大连续行驶时间来判断，提取的信息不够完整，一个用户可能只是偶尔长途，但有的司机长期都是长途，还需算长途驾驶频度	a.将特征值长途驾驶频度也加进新特征里 b.将经纬度两列特征计算出一个距离的新特征来，该特征主要是两个临近时间段的距离，对距离归一化再放入模型训练
5	19 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、	trip	岭回	分数：0.079，加入长途驾驶频度、距离、坡度、通话时左拐、通话时右拐特征应	a.可以将经纬度划分成一些区域，对于区域赋予风险系数值，来做量化分析

	每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向、急转弯、方向连续变换（基于电话和速度）、总路程（基于经纬度）、平均路程、长途驾驶频度、距离、坡度、通话时左拐、通话时右拐		归	该保留下来	b.更换模型，试试用回归的思路和模型 c.将 TRIP_ID 删除，基于用户的时间进行排序，同时赋予新的 TRIP
6	23 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向、急转弯、方向连续变换（基于电话和速度）、总路程（基于经纬度）、平均路程、长途驾驶频度、距离、坡度、通话时左拐、通话时右拐、高度差、速度差、方向差、上下坡角度	trip	岭回归	分数： 0.07024，没有对单个特征相关性进行判断，无法得知新加的哪些特征对结果产生较什么样的影响。但是总体结果的值变化不大，这些特征可以考虑保留	a.对长途驾驶频度、距离、坡度、通话时左拐、通话时右拐高度差，速度差，方向差，上下坡角度这些特征逐一分析，看哪个特征起到关键作用 b. 在加入特征的过程中，比较 train_predict_label 和 train_Y 之间的方差
7	27 个特征：打电话状态均值、打电话次数求和、疲劳驾驶状态均值、疲劳驾驶次数求和、每个行程打电话频次、启动速度、熄灭速度、平均速度、平均海拔、方向、急转弯、方向连续变换（基于电话和速度）、总路程（基于经纬度）、平均路程、长途驾驶频度、距离、坡度、通话时左拐、通话时右拐、高度差、速度差、方向差、上下坡角度加 4 个统计特征值:速度最大值、海拔方差、基于 TIME 的出行时刻、速度方差	trip	LGB 回归	分数：0.10124，加入新的统计值特征后上升 0.03 多,结合以往实验发现统计特征能提高效果，主要因为统计特征里面可以反应数据最原始的信息，并且已经包含了新创建特征所包含的内容；而且新创建特征是否能表达我们所赋予它的意义，这一点我们很难考证，大多主观上的理解	
8	24 小时中行程记录比例、工作日行程记录比例、周末工作日记录比例、行程总数、平均速度、高度、速度方差、打电话比例	用户	LGB 回归	分数：0.11062。发现从统计学的角度找新特征的结果更好。	在更细致地找一些统计特征。

参考文献

十分钟搞定 - pandas https://www.cnblogs.com/chaosimple/p/4153083.html
Xgboost 原理和实战 - https://blog.csdn.net/sinat_22594309/article/details/60957594
天池比赛经验总结 - https://segmentfault.com/a/1190000012084849
Xgboost 原理 - https://blog.csdn.net/sinat_22594309/article/details/60957594
Xgboost 调参过程 - http://www.cnblogs.com/harvey888/p/7203256.html
Xgboost 调参过程 - https://blog.csdn.net/han_xiaoyang/article/details/52665396
Xgboost 调参经验 - http://www.cnblogs.com/mfryf/p/6293814.html
LGB 原理 - https://blog.csdn.net/niaolianjiulin/article/details/76584785
LGB 参数速查 - https://blog.csdn.net/weiyongle1996/article/details/78446244
缺失值处理 - https://blog.csdn.net/lujiandong1/article/details/52654703