城市交通道路流量预测

王雪霏 3120181096
冯雨晴 3120180988

1. 引言

随着计算机、网络技术的快速发展，我们生活的方方面面都在发生着变化，建设智慧城市、智慧出行等以移动互联网技术为支撑的更加高效、便捷的新型生活模式已经成为我们发展的重要方向。然而，在生活的诸多方面中，城市交通问题始终面临着挑战，亟待解决。考虑到如今的移动互联网时代使得每个出行者都成为了交通信息的贡献者，超大规模的位置数据在云端进行处理和融合，生成城市全时段、无盲区的交通信息，故可以运用大数据的方法缓解拥堵毒瘤对城市管理的影响，更好的帮助交通管理者提前制定管控方案，预防和削减拥堵，实现智慧出行。基于此，本文利用互联网交通信息建立算法模型，精准预测各关键路段在某个时段的通行时间，实现对交通状态波动起伏的预判，并通过计算提交预测值和记录真实值之间的误差确定预测准确率，评估所提交的预测算法，旨在提供有效的方法来助力智慧出行和城市交通智能管控。

2. 问题描述

2.1 数据描述

本文采用阿里天池比赛提供的数据集，主要包括城市关键路段

（link）的属性信息，路段间网络拓扑结构以及每条路段在历史各时间段内的通行时间。具体如下面三个表格所示：

（1）路段(link)属性表：每条道路的每个通行方向由多条路段（link）构成，数据集中提供每条 link 的唯一标识，长度，宽度，以及道路类型。

表 2. 1

属性	类型	说明
link_ID	string	每条路段（link）的唯一标识
length	double	link 长度（米）
width	double	link 宽度（米）
link_class	int	link 道路等级，例如 1 代表主干道

（2）link 上下游关系表：link 之间按照车辆允许通行的方向存在上下游关系，数据集中提供每条 link 的直接上游 link 和直接下游 link。

表 2. 2

属性	类型	说明
link_ID	string	每条路段（link）的唯一标识
In_links	string	link 的直接上游 link，linkID 之间以#分割
out_links	string	link 的直接下游 link，linkID 之间以#分割

（3）link 历史通行时间表：数据集中记录了历史每天不同时间段内（2min 为一个时间段）每条 link 上的平均旅行时间，每个时间段的平均旅行时间是基于在该时间段内进入link的车辆在该link上的旅行时间产出。

表 2. 3

属性	类型	说明
link_ID	string	每条路段（link）的唯一标识

date_time	date	日期，例如‘2015-10-01’
time_interval	string	时间段，例如[2015-09-01 00:00:00’, 2015-09-01 00:00:10）
travel_time	double	车辆在路段上的平均旅行时间（秒）

2.2 预测目标

已知数据：link 属性表，link 上下游关系表，以及 2016 年 7 月，2017 年 3 月至 2017 年 6 月 132 条 link 的覆盖全天的旅行时间数据(2min 粒度)和 2017 年 7 月每天[6:00 - 8:00)，[13:00 - 15:00)，[16:00 - 18:00)三个时段 132 条 link 的旅行时间数据。

预测目标：2017 年 7 月 1 日至 7 月 31 早高峰、日平峰和晚高峰三个时段中各一个小时的数据，预测时段为早高峰[8:00 - 9:00)，日平峰[15:00 - 16:00），晚高峰[18:00 - 19:00)。

2.3 评估指标

$$\mathrm{MAPE} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{T_i}\frac{\left|ttp_{i,j} - ttr_{i,j}\right|}{ttr_{i,j}}}{\sum_{i=1}^{N} T_i}$$

ttp ： 提交的预测通行时间。

ttr: 真实通行时间。

N: 预测 link 数量。

T_i: 第 i 个 link 上的预测时间片数量。

MAPE 值越低说明模型准确度越高。

3．技术方案

3.1 可视化分析

我们尝试从道路属性（长度、宽度），拓扑结构（入度，出度），时间特征(年月日小时)等方向对给定的数据集做粗略的可视化分析。下面展示可视化预处理的结果图。

（1）道路属性（长度、宽度）与平均旅行时间的关系

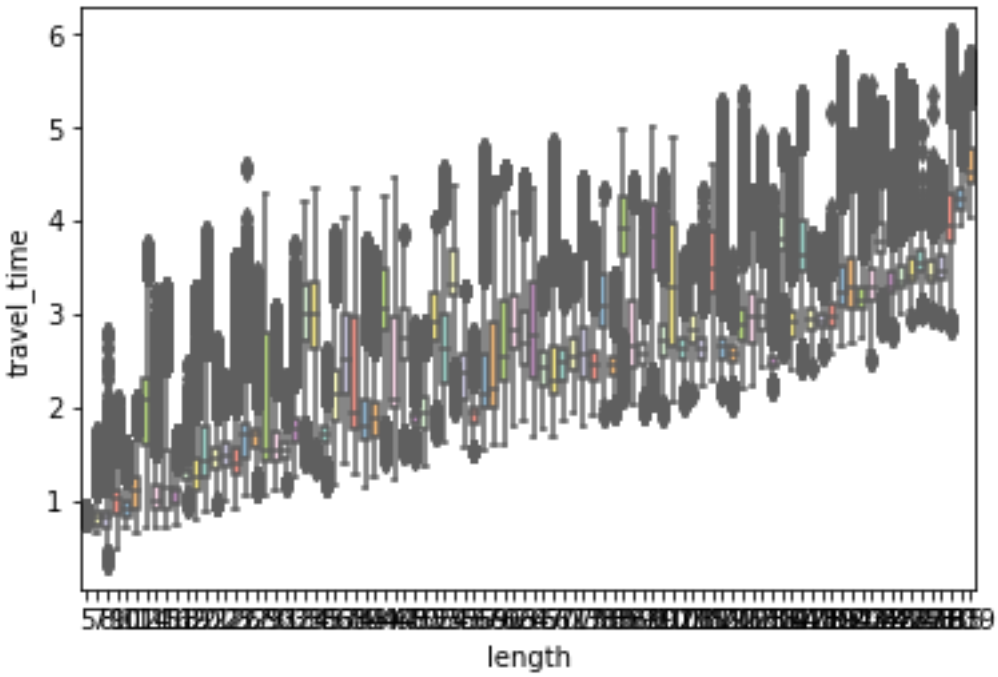


图 3.1 道路长度--平均旅行时间盒图

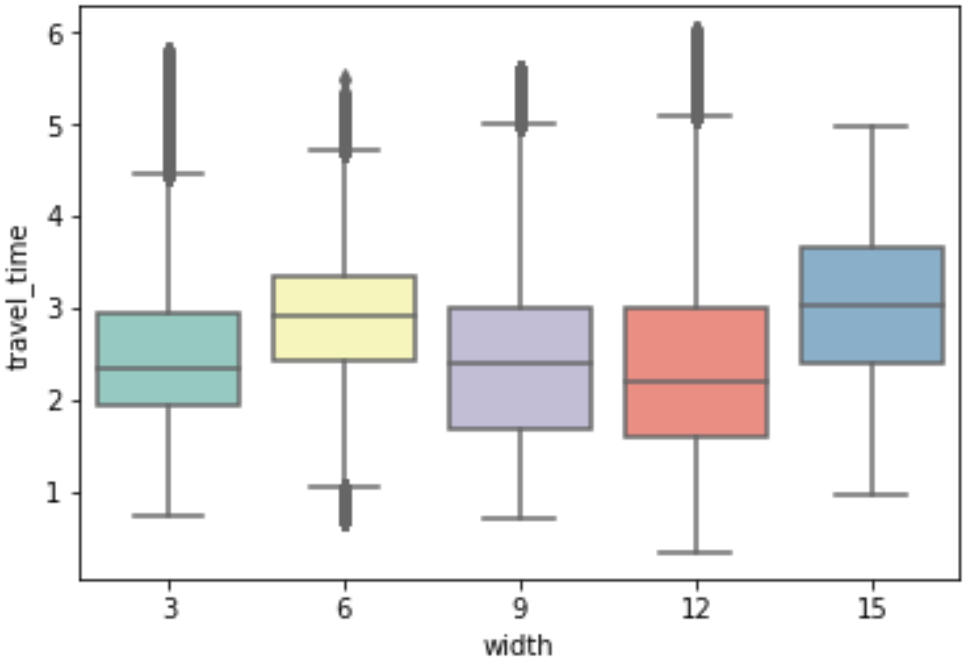


图 3. 2 道路宽度--平均旅行时间盒图

上面图 3. 1 和图 3. 2 使用盒图分别展示了道路的长度，宽度对旅行时间分布的影响。可以看出随着道路长度的增加，旅行时间的整体分布呈上升趋势，这符合我们实际生活中的普遍认知。而对于不同宽度的道路，旅行时间有所不同，但整体并没有表现出道路越宽，旅行时间越短的下降趋势，可能与道路的设计规划有一定关联。此外，两张图中存在较多的离群点，应该与堵车存在一定关联。

（2）拓扑结构（入度，出度）与平均旅行时间的关系

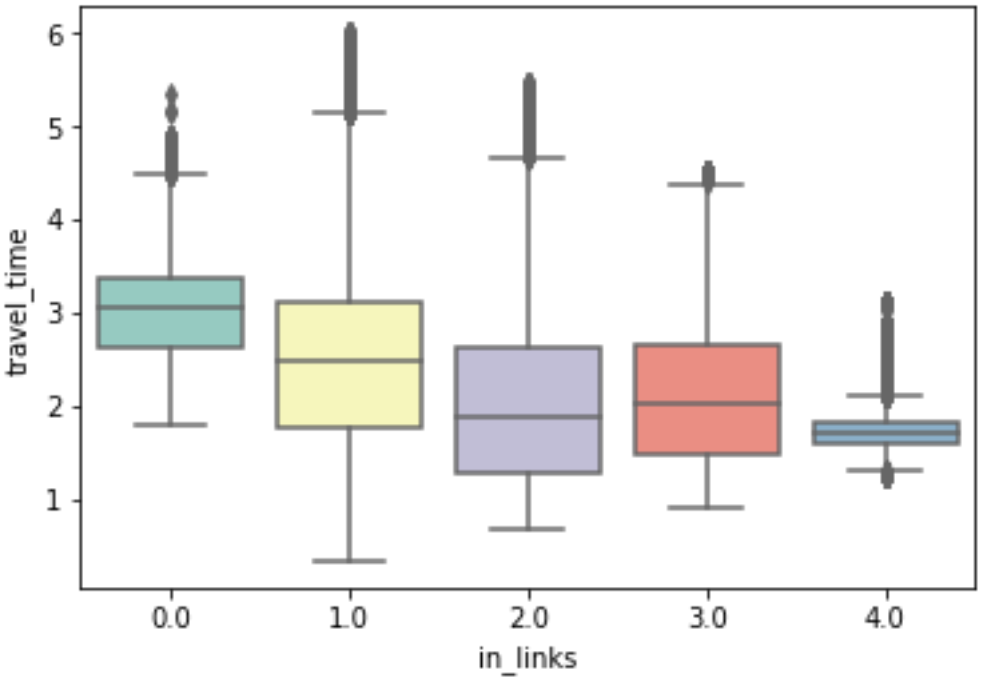


图 3.3 上游路段数量--平均旅行时间盒图

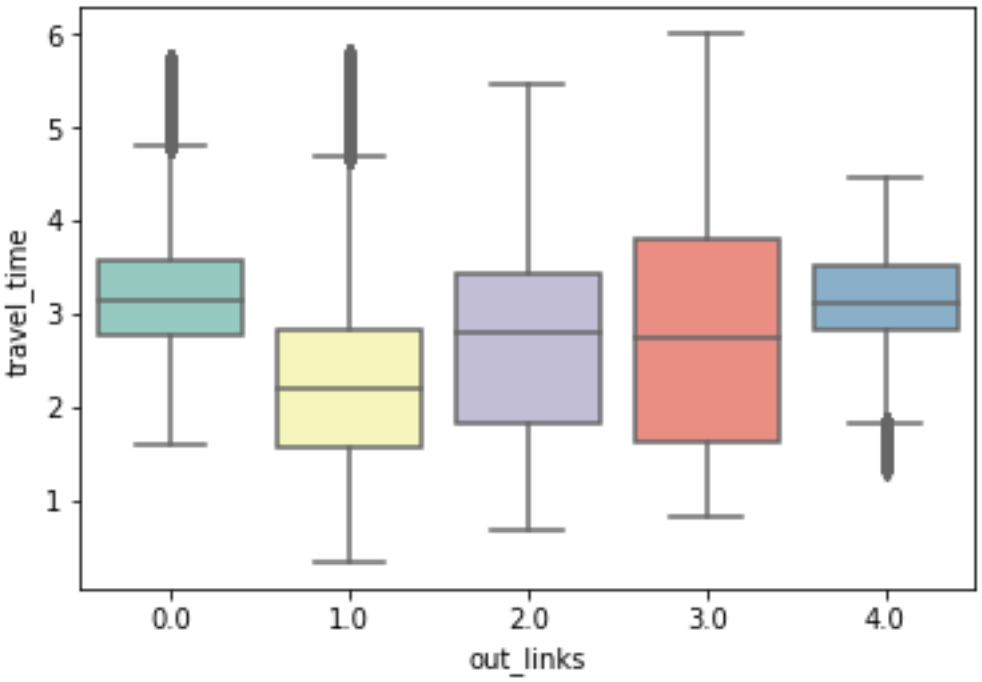
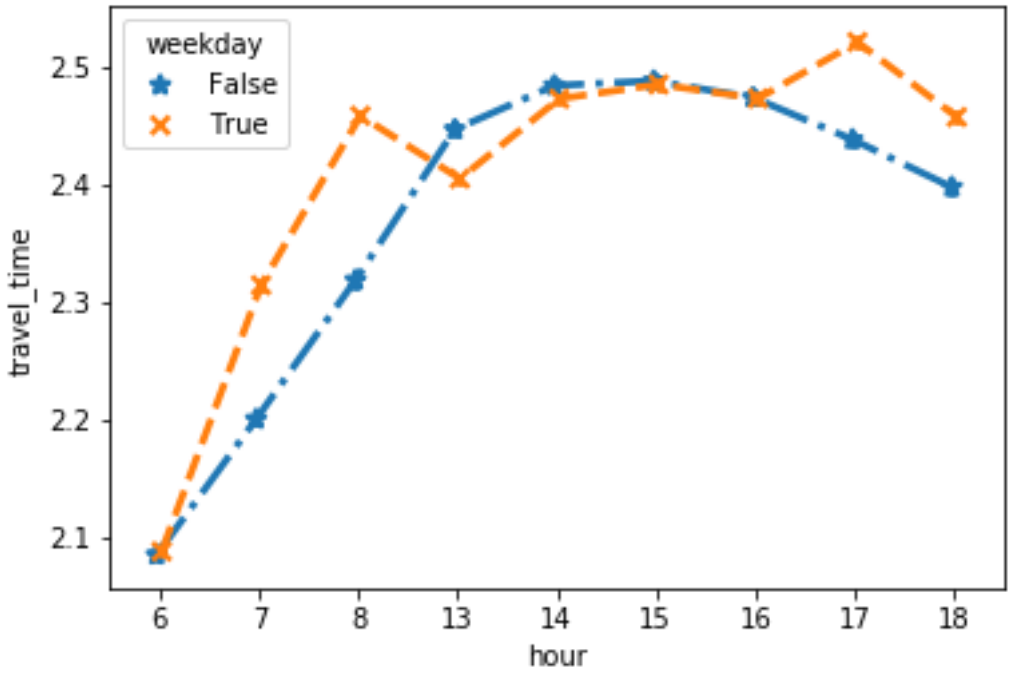


图 3.4 下游路段数量--平均旅行时间盒图

每条道路的入度和出度与整个交通网络的空间特征有关。根据直观感受，汇入的道路的越多，在通过时可能会更加拥堵，旅行时间也

会相应越长，然而上述两幅图展示的结果与直观感受并不一致，可能还受其他因素影响。不过每条道路的入度和出度不同，旅行时间也存在变化，具有一定的区分度，可以作为预测的特征。

（3）时间特征与平均旅行时间的关系



上图展示的是一天时间范围内旅行时间均值的变化。可以看出对于工作日和周末，一天内的旅行时间均值变换存在明显区分。对于一天的不同时段，旅行时间也存在明显变化。因此，时间特征也可以作为预测旅行时间的重要特征。

3.2 特征选择

我们需要预测的目标是 2017 年 7 月份每天高峰期各路段的旅行时间，主要解决思路是根据道路历史通过时间预测其未来一个小时内的通过时间。因此进行预测使用的特征主要和时间，道路相关，并将

其分为以下几类：

①时间特征：是否是节假日，年月日，小时分钟等。

②道路属性：长度，宽度，道路类型等。

③拓扑结构特征：每天道路的入度和出度。

④路况统计特征：以不同时间粒度、不同时间窗口取历史同一时刻的最大值、平均值、最小值、中位数、方差等。根据时间特征，为了极大化前段时间数据对后面一段时间连续数据的影响，采用了滑窗方法来对前一段时间的 travel_time 进行处理。

3.3 预测模型

在完成数据预处理和特征提取之后，采用 XGBoost 和 LGBM 两个模型建立旅行时间预测模型，并进行模型融合。

XGBoost 是一种集成学习方法，将多个分类回归树进行组合，以达到更强的泛化能力和更好的效果。XGBoost 不仅学习效果很好，而且速度也很快，相比梯度提升算法在另一个常用机器学习库 scikit-learn 中的实现，XGBoost 的性能经常有十倍以上的提升。XGBoost 实现的是一种通用的 Tree Boosting 算法，此算法的一个代表为梯度提升决策树（Gradient Boosting Decision Tree，GBDT）。GBDT 的原理是，首先使用训练集和样本真值（即标准答案）训练一棵树，然后使用这棵树预测训练集，得到每个样本的预测值，由于预测值与真值存在偏差，所以二者相减可以得到“残差”。接下来训练第二棵树，此时不再使用真值，而是使用残差作为标准答案。两棵树

训练完成后，可以再次得到每个样本的残差，然后进一步训练第三棵树，以此类推。树的总棵数可以人为指定，也可以监控某些指标（例如验证集上的误差）来停止训练。在预测新样本时，每棵树都会有一个输出值，将这些输出值相加，即得到样本最终的预测值。

LGBM（LightGBM）是微软 2017 年新提出的比 Xgboost 更强大、速度更快的模型，性能上有很大的提升，与传统算法相比具有更快的训练效率、较低的内存使用、更高的准确率，并且支持并行化学习、可处理大规模数据等优点。

4．结果与分析

实验结果分为两个部分，一方面输出了各特征的重要程度，以便使用正确的特征进行预测；另一方面输出了前面所阐明的预测目标，即 2017 年 7 月 1 日至 7 月 31 日早高峰[8:00－9:00)，日平峰[15:00－16:00），晚高峰[18:00－19:00)（2 分钟为粒度）的关键道路上的平均旅行时间。

4.1 特征的重要程度

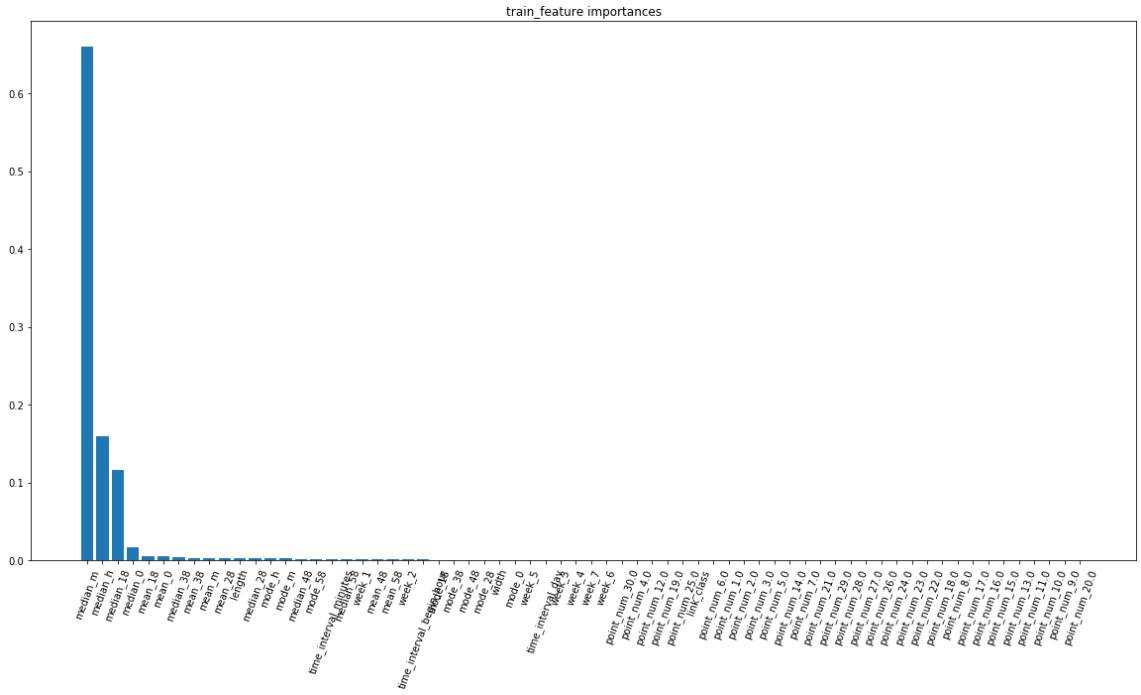


图 4.1 特征重要程度的结果图

4.2 预测结果

由于数据过多，故将结果写入了文件中，数据共有四列，分别为：link_ID，date_time，time_interval，**travel_time**（即预测值）。部分截图如下图所示：

```
337790628002851051 4#2017-07-01#[2017-07-01 08:00:00,2017-07-01 08:02:00)#2.1604579999999998
337790628002851051 4#2017-07-01#[2017-07-01 08:02:00,2017-07-01 08:04:00)#2.172084
337790628002851051 4#2017-07-01#[2017-07-01 08:04:00,2017-07-01 08:06:00)#2.1848544
337790628002851051 4#2017-07-01#[2017-07-01 08:06:00,2017-07-01 08:08:00)#2.1831626
337790628002851051 4#2017-07-01#[2017-07-01 08:08:00,2017-07-01 08:10:00)#2.2518286
337790628002851051 4#2017-07-01#[2017-07-01 08:10:00,2017-07-01 08:12:00)#2.1958162000000003
337790628002851051 4#2017-07-01#[2017-07-01 08:12:00,2017-07-01 08:14:00)#2.2161624
337790628002851051 4#2017-07-01#[2017-07-01 08:14:00,2017-07-01 08:16:00)#2.2041008
337790628002851051 4#2017-07-01#[2017-07-01 08:16:00,2017-07-01 08:18:00)#2.2119720000000003
337790628002851051 4#2017-07-01#[2017-07-01 08:18:00,2017-07-01 08:20:00)#2.2566082
337790628002851051 4#2017-07-01#[2017-07-01 08:20:00,2017-07-01 08:22:00)#2.2395216000000002
337790628002851051 4#2017-07-01#[2017-07-01 08:22:00,2017-07-01 08:24:00)#2.2124004
337790628002851051 4#2017-07-01#[2017-07-01 08:24:00,2017-07-01 08:26:00)#2.24552
337790628002851051 4#2017-07-01#[2017-07-01 08:26:00,2017-07-01 08:28:00)#2.1976388
337790628002851051 4#2017-07-01#[2017-07-01 08:28:00,2017-07-01 08:30:00)#2.2151166000000004
337790628002851051 4#2017-07-01#[2017-07-01 08:30:00,2017-07-01 08:32:00)#2.2192656
337790628002851051 4#2017-07-01#[2017-07-01 08:32:00,2017-07-01 08:34:00)#2.2068284
337790628002851051 4#2017-07-01#[2017-07-01 08:34:00,2017-07-01 08:36:00)#2.2370132
337790628002851051 4#2017-07-01#[2017-07-01 08:36:00,2017-07-01 08:38:00)#2.2108142
337790628002851051 4#2017-07-01#[2017-07-01 08:38:00,2017-07-01 08:40:00)#2.2058822
337790628002851051 4#2017-07-01#[2017-07-01 08:40:00,2017-07-01 08:42:00)#2.1917780000000002
337790628002851051 4#2017-07-01#[2017-07-01 08:42:00,2017-07-01 08:44:00)#2.22745
337790628002851051 4#2017-07-01#[2017-07-01 08:44:00,2017-07-01 08:46:00)#2.2202523999999997
337790628002851051 4#2017-07-01#[2017-07-01 08:46:00,2017-07-01 08:48:00)#2.2436692000000003
337790628002851051 4#2017-07-01#[2017-07-01 08:48:00,2017-07-01 08:50:00)#2.19427
337790628002851051 4#2017-07-01#[2017-07-01 08:50:00,2017-07-01 08:52:00)#2.2154092
337790628002851051 4#2017-07-01#[2017-07-01 08:52:00,2017-07-01 08:54:00)#2.213604
337790628002851051 4#2017-07-01#[2017-07-01 08:54:00,2017-07-01 08:56:00)#2.204882
337790628002851051 4#2017-07-01#[2017-07-01 08:56:00,2017-07-01 08:58:00)#2.1795678
337790628002851051 4#2017-07-01#[2017-07-01 08:58:00,2017-07-01 09:00:00)#2.1588984000000004
337790628002851051 4#2017-07-01#[2017-07-01 15:00:00,2017-07-01 15:02:00)#2.2713410000000005
337790628002851051 4#2017-07-01#[2017-07-01 15:02:00,2017-07-01 15:04:00)#2.3614176000000002
337790628002851051 4#2017-07-01#[2017-07-01 15:04:00,2017-07-01 15:06:00)#2.3928719999999997
337790628002851051 4#2017-07-01#[2017-07-01 15:06:00,2017-07-01 15:08:00)#2.3828378
337790628002851051 4#2017-07-01#[2017-07-01 15:08:00,2017-07-01 15:10:00)#2.4474139999999998
337790628002851051 4#2017-07-01#[2017-07-01 15:10:00,2017-07-01 15:12:00)#2.3984428
337790628002851051 4#2017-07-01#[2017-07-01 15:12:00,2017-07-01 15:14:00)#2.4123844
337790628002851051 4#2017-07-01#[2017-07-01 15:14:00,2017-07-01 15:16:00)#2.3928778
337790628002851051 4#2017-07-01#[2017-07-01 15:16:00,2017-07-01 15:18:00)#2.3993542000000003
337790628002851051 4#2017-07-01#[2017-07-01 15:18:00,2017-07-01 15:20:00)#2.4622644
337790628002851051 4#2017-07-01#[2017-07-01 15:20:00,2017-07-01 15:22:00)#2.4616464000000002
337790628002851051 4#2017-07-01#[2017-07-01 15:22:00,2017-07-01 15:24:00)#2.4233219999999998
337790628002851051 4#2017-07-01#[2017-07-01 15:24:00,2017-07-01 15:26:00)#2.4589438
337790628002851051 4#2017-07-01#[2017-07-01 15:26:00,2017-07-01 15:28:00)#2.3975816
337790628002851051 4#2017-07-01#[2017-07-01 15:56:00,2017-07-01 15:58:00)#2.395595
337790628002851051 4#2017-07-01#[2017-07-01 15:58:00,2017-07-01 16:00:00)#2.3701456
337790628002851051 4#2017-07-01#[2017-07-01 18:00:00,2017-07-01 18:02:00)#2.472902
337790628002851051 4#2017-07-01#[2017-07-01 18:02:00,2017-07-01 18:04:00)#2.4474862
337790628002851051 4#2017-07-01#[2017-07-01 18:04:00,2017-07-01 18:06:00)#2.4086950000000003
337790628002851051 4#2017-07-01#[2017-07-01 18:06:00,2017-07-01 18:08:00)#2.3854578
337790628002851051 4#2017-07-01#[2017-07-01 18:08:00,2017-07-01 18:10:00)#2.3863872
337790628002851051 4#2017-07-01#[2017-07-01 18:10:00,2017-07-01 18:12:00)#2.3719236
337790628002851051 4#2017-07-01#[2017-07-01 18:12:00,2017-07-01 18:14:00)#2.4046347999999997
337790628002851051 4#2017-07-01#[2017-07-01 18:14:00,2017-07-01 18:16:00)#2.407272
337790628002851051 4#2017-07-01#[2017-07-01 18:16:00,2017-07-01 18:18:00)#2.3999639999999998
337790628002851051 4#2017-07-01#[2017-07-01 18:18:00,2017-07-01 18:20:00)#2.3687067999999996
337790628002851051 4#2017-07-01#[2017-07-01 18:20:00,2017-07-01 18:22:00)#2.3644407999999997
337790628002851051 4#2017-07-01#[2017-07-01 18:22:00,2017-07-01 18:24:00)#2.3383608
337790628002851051 4#2017-07-01#[2017-07-01 18:24:00,2017-07-01 18:26:00)#2.3152768000000004
337790628002851051 4#2017-07-01#[2017-07-01 18:26:00,2017-07-01 18:28:00)#2.324205
337790628002851051 4#2017-07-01#[2017-07-01 18:28:00,2017-07-01 18:30:00)#2.371851
337790628002851051 4#2017-07-01#[2017-07-01 18:30:00,2017-07-01 18:32:00)#2.3698661999999997
337790628002851051 4#2017-07-01#[2017-07-01 18:32:00,2017-07-01 18:34:00)#2.3516708
337790628002851051 4#2017-07-01#[2017-07-01 18:34:00,2017-07-01 18:36:00)#2.3685840000000002
337790628002851051 4#2017-07-01#[2017-07-01 18:36:00,2017-07-01 18:38:00)#2.2692197999999997
337790628002851051 4#2017-07-01#[2017-07-01 18:38:00,2017-07-01 18:40:00)#2.2887412
337790628002851051 4#2017-07-01#[2017-07-01 18:40:00,2017-07-01 18:42:00)#2.2569624
337790628002851051 4#2017-07-01#[2017-07-01 18:42:00,2017-07-01 18:44:00)#2.2357986
337790628002851051 4#2017-07-01#[2017-07-01 18:44:00,2017-07-01 18:46:00)#2.2602004
337790628002851051 4#2017-07-01#[2017-07-01 18:46:00,2017-07-01 18:48:00)#2.2377583999999997
337790628002851051 4#2017-07-01#[2017-07-01 18:48:00,2017-07-01 18:50:00)#2.2219634
337790628002851051 4#2017-07-01#[2017-07-01 18:50:00,2017-07-01 18:52:00)#2.2835343999999997
337790628002851051 4#2017-07-01#[2017-07-01 18:52:00,2017-07-01 18:54:00)#2.2782334
337790628002851051 4#2017-07-01#[2017-07-01 18:54:00,2017-07-01 18:56:00)#2.2132484
337790628002851051 4#2017-07-01#[2017-07-01 18:56:00,2017-07-01 18:58:00)#2.1991702
337790628002851051 4#2017-07-01#[2017-07-01 18:58:00,2017-07-01 19:00:00)#2.2045152000000003
337790628002851051 4#2017-07-02#[2017-07-02 08:00:00,2017-07-02 08:02:00)#2.1291124
337790628002851051 4#2017-07-02#[2017-07-02 08:02:00,2017-07-02 08:04:00)#2.1624852
337790628002851051 4#2017-07-02#[2017-07-02 08:04:00,2017-07-02 08:06:00)#2.1491225999999997
337790628002851051 4#2017-07-02#[2017-07-02 08:06:00,2017-07-02 08:08:00)#2.1548494
337790628002851051 4#2017-07-02#[2017-07-02 08:08:00,2017-07-02 08:10:00)#2.1198898
337790628002851051 4#2017-07-02#[2017-07-02 08:10:00,2017-07-02 08:12:00)#2.1137536
337790628002851051 4#2017-07-02#[2017-07-02 08:12:00,2017-07-02 08:14:00)#2.1237024
337790628002851051 4#2017-07-02#[2017-07-02 08:14:00,2017-07-02 08:16:00)#2.1389769999999997
337790628002851051 4#2017-07-02#[2017-07-02 08:16:00,2017-07-02 08:18:00)#2.1359186
337790628002851051 4#2017-07-02#[2017-07-02 08:18:00,2017-07-02 08:20:00)#2.121433
337790628002851051 4#2017-07-02#[2017-07-02 08:20:00,2017-07-02 08:22:00)#2.1220274
337790628002851051 4#2017-07-02#[2017-07-02 08:22:00,2017-07-02 08:24:00)#2.148893
337790628002851051 4#2017-07-02#[2017-07-02 08:24:00,2017-07-02 08:26:00)#2.1709342
337790628002851051 4#2017-07-02#[2017-07-02 08:26:00,2017-07-02 08:28:00)#2.1818592000000003
```

```
4377906288063800514#2017-07-11#[2017-07-11 15:48:00,2017-07-11 15:50:00)#11.304763
4377906288063800514#2017-07-11#[2017-07-11 15:50:00,2017-07-11 15:52:00)#11.3434042
4377906288063800514#2017-07-11#[2017-07-11 15:52:00,2017-07-11 15:54:00)#11.3479218
4377906288063800514#2017-07-11#[2017-07-11 15:54:00,2017-07-11 15:56:00)#11.3297942
4377906288063800514#2017-07-11#[2017-07-11 15:56:00,2017-07-11 15:58:00)#11.3373864
4377906288063800514#2017-07-11#[2017-07-11 15:58:00,2017-07-11 16:00:00)#11.3330456
4377906288063800514#2017-07-11#[2017-07-11 18:00:00,2017-07-11 18:02:00)#11.372707199999999
4377906288063800514#2017-07-11#[2017-07-11 18:02:00,2017-07-11 18:04:00)#11.3371796
4377906288063800514#2017-07-11#[2017-07-11 18:04:00,2017-07-11 18:06:00)#11.277481
4377906288063800514#2017-07-11#[2017-07-11 18:06:00,2017-07-11 18:08:00)#11.2730154
4377906288063800514#2017-07-11#[2017-07-11 18:08:00,2017-07-11 18:10:00)#11.2442326
4377906288063800514#2017-07-11#[2017-07-11 18:10:00,2017-07-11 18:12:00)#11.2431072
4377906288063800514#2017-07-11#[2017-07-11 18:12:00,2017-07-11 18:14:00)#11.1954326
4377906288063800514#2017-07-11#[2017-07-11 18:14:00,2017-07-11 18:16:00)#11.251227
4377906288063800514#2017-07-11#[2017-07-11 18:16:00,2017-07-11 18:18:00)#11.2223426
4377906288063800514#2017-07-11#[2017-07-11 18:18:00,2017-07-11 18:20:00)#11.1597842
4377906288063800514#2017-07-11#[2017-07-11 18:20:00,2017-07-11 18:22:00)#11.1326088
4377906288063800514#2017-07-11#[2017-07-11 18:22:00,2017-07-11 18:24:00)#11.155811799999999
4377906288063800514#2017-07-11#[2017-07-11 18:24:00,2017-07-11 18:26:00)#10.9181272
4377906288063800514#2017-07-11#[2017-07-11 18:26:00,2017-07-11 18:28:00)#11.0222728
4377906288063800514#2017-07-11#[2017-07-11 18:28:00,2017-07-11 18:30:00)#11.0254748
4377906288063800514#2017-07-11#[2017-07-11 18:30:00,2017-07-11 18:32:00)#10.9179676
4377906288063800514#2017-07-11#[2017-07-11 18:32:00,2017-07-11 18:34:00)#11.022428600000001
4377906288063800514#2017-07-11#[2017-07-11 18:34:00,2017-07-11 18:36:00)#10.950414600000002
4377906288063800514#2017-07-11#[2017-07-11 18:36:00,2017-07-11 18:38:00)#10.9085216
4377906288063800514#2017-07-11#[2017-07-11 18:38:00,2017-07-11 18:40:00)#10.892233
4377906288063800514#2017-07-11#[2017-07-11 18:40:00,2017-07-11 18:42:00)#10.859099400000002
4377906288063800514#2017-07-11#[2017-07-11 18:42:00,2017-07-11 18:44:00)#10.7830048
4377906288063800514#2017-07-11#[2017-07-11 18:44:00,2017-07-11 18:46:00)#10.8157096
4377906288063800514#2017-07-11#[2017-07-11 18:46:00,2017-07-11 18:48:00)#10.7355816
4377906288063800514#2017-07-11#[2017-07-11 18:48:00,2017-07-11 18:50:00)#10.7124694
4377906288063800514#2017-07-11#[2017-07-11 18:50:00,2017-07-11 18:52:00)#10.7126632
4377906288063800514#2017-07-11#[2017-07-11 18:52:00,2017-07-11 18:54:00)#10.7167172
4377906288063800514#2017-07-11#[2017-07-11 18:54:00,2017-07-11 18:56:00)#10.689324200000001
4377906288063800514#2017-07-11#[2017-07-11 18:56:00,2017-07-11 18:58:00)#10.6913046
4377906288063800514#2017-07-11#[2017-07-11 18:58:00,2017-07-11 19:00:00)#10.664521400000002
4377906288063800514#2017-07-12#[2017-07-12 08:00:00,2017-07-12 08:02:00)#14.157918599999999
4377906288063800514#2017-07-12#[2017-07-12 08:02:00,2017-07-12 08:04:00)#13.074872199999998
4377906288063800514#2017-07-12#[2017-07-12 08:04:00,2017-07-12 08:06:00)#12.147231999999999
4377906288063800514#2017-07-12#[2017-07-12 08:06:00,2017-07-12 08:08:00)#12.0124328
4377906288063800514#2017-07-12#[2017-07-12 08:08:00,2017-07-12 08:10:00)#11.799725200000001
4377906288063800514#2017-07-12#[2017-07-12 08:10:00,2017-07-12 08:12:00)#11.7864434
4377906288063800514#2017-07-12#[2017-07-12 08:12:00,2017-07-12 08:14:00)#11.7764504
4377906288063800514#2017-07-12#[2017-07-12 08:14:00,2017-07-12 08:16:00)#11.7389466
4377906288063800514#2017-07-12#[2017-07-12 08:16:00,2017-07-12 08:18:00)#11.659346799999998
```

附录：

Github 代码链接：https://github.com/canlanqianyang/TrafficFlowPredict

部分代码截图如下：

```python
def XGB_train(t1,t2,on_or_off,i,mode):
    train = pd.read_csv('/home/wangxuefei/Downloads/data/%s.txt'%(t1),low_memory=False)
    train_label = np.log1p(train.pop('travel_time'))
    print(train.shape[1])
    test = pd.read_csv('/home/wangxuefei/Downloads/data/%s.txt'%(t2),low_memory=False)
    test_lable = np.log1p(test.pop('travel_time'))
    print(test.shape[1])
    col_2=list(train.columns.values)
    list_2=[name for name in col_2 if name.startswith('median_')]
    train.drop(['time_interval_begin','time_interval_week'],inplace=True,axis=1)
    #train.drop(list_2, inplace=True, axis=1)
    test.drop(['time_interval_begin','time_interval_week'],inplace=True,axis=1)
    print("训练样本和测试样本行列数")
    print(train.shape,test.shape)

    if mode == 1:
        xlf = xgb.XGBRegressor(max_depth=11,
                               learning_rate=0.01,
                               n_estimators=425,
                               silent=True,
                               objective=mape_object,
                               gamma=0,
                               min_child_weight=6,
                               max_delta_step=0,
                               subsample=0.8,
                               colsample_bytree=0.8,
                               colsample_bylevel=1,
                               reg_alpha=1e0,
                               reg_lambda=0,
                               scale_pos_weight=1,
                               seed=12,
                               missing=None)
        xlf.fit(train.values, train_label.values, eval_metric=mape_ln, verbose=True, eval_set=[(test.values, test_lable.values)])
        print("正在预测...")
        result = xlf.predict(test.values)
    elif mode ==2:
        clf = RandomForestRegressor(n_estimators=20,
                                    max_depth=9,
                                    min_samples_split=3,
                                    min_samples_leaf=5,
                                    n_jobs=-1,
                                    random_state=0)
        clf.fit(train.values, train_label.values)
        result = clf.predict(test.values)
    if on_or_off == "off":
        print("写入文件...")
        #写入文件
```

```python
def runLGBM(train_X, train_y, test_X, test_y,on_off):
    """
    :param train_X:  训练集数据
    :param train_y:  训练集标签 log
    :param test_X:   测试集数据
    :param test_y:   测试集标签 log
    :return:
    """
    params = {
        'task': 'train',
        'boosting_type': 'gbdt',
        'objective': 'regression_l1',
        # 'metric': 'l1',
        'num_leaves': 127,
        'learning_rate': 0.01,
        #'feature_fraction': 0.9,
        #'bagging_fraction': 0.8,
        'bagging_freq': 5,
        'verbose': 0,
        'min_data_in_leaf': 600
    }

    if on_off == "off":

        lgb_train = lgb.Dataset(train_X, train_y, silent=True)
        lgb_eval = lgb.Dataset(test_X, test_y, reference=lgb_train)
        model_ = lgb.train(params,
                           lgb_train,
                           num_boost_round=2000,
                           fobj=mape_object,
                           feval=mape_ln,
                           valid_sets=lgb_eval,
                           verbose_eval=True,
                           early_stopping_rounds=5)

        pred_test_y = model_.predict(test_X, num_iteration=model_.best_iteration)

    if on_off == "on":

        lgb_train = lgb.Dataset(train_X, train_y, silent=True)

        model_ = lgb.train(params,
                           lgb_train,
                           num_boost_round=296,
                           fobj=mape_object,
                           feval=mape_ln
                           )
```