

纽约公交出行时间预测

New York City Taxi Trip Duration

一、赛题介绍

背景介绍

竞赛数据集基于谷歌云平台 Big Query 中提供的 2016 年纽约市黄包车出行记录数据。这些数据最初由纽约市出租车和豪华轿车委员会 (TLC) 发布。这些数据经过采样和清理，用于此次游乐场竞赛。根据单个行程属性，参赛者应预测测试集中**每个行程的持续时间**。

评价指标

本文采用均方根对数误差：

$$\epsilon = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

其中， p_i 表示样本 i 的预测出行时间， a_i 表示样本 i 的实际出行时间。

提交文件

submission.csv提交格式如下：

id	trip_duration
id00001	978
id00002	546

二、数据情况

各字段介绍如下：

- **id** - 唯一标识
- **vendor_id** - 表示与行程记录相关的提供商的代码

- **pickup_datetime** - 出租车打表启动时间
- **dropoff_datetime** - 出租车打表停止时间
- **passenger_count** - 乘客人数
- **pickup_longitude** - 上车经度
- **pickup_latitude** - 上车纬度
- **dropoff_longitude** - 下车经度
- **dropoff_latitude** - 下车纬度
- **store_and_fwd_flag** - 这个记录表示是否在离线状态存储指标，Y表示存储并转发，N表示并没有存储和转发
- **trip_duration** - 出行时间

三、解决方案

原始数据共计11列，其中 `id` 与 `vendor_id`、`store_and_fwd_flag` 基本上没用。思考出行时间的影响因子，首先最重要的就是路程和速度，在这里我们速度不知道，路程不知道，只知道时间，而且需要时间。样本量大概在1.4m左右，不是特别大，我们直接读取进来就行。

经纬度是十分重要的信息，有了经纬度就可以通过距离构造方法近似构造路程，当然这是在高程未知、道路未知的情况下，在这里我们构造了两个距离，一个是地球椭圆距离Haversine，一个是街区距离Manhattan距离。如果能够获取到路网和高程数据，我相信对我们的距离构造是十分有帮助的。

关注信息

我们注意到，出行人数这个属性存在0人，这显然是不合理的，我们统一将其加到1人。然后是在验证集中的样本分布情况

pc	cnt
0	60
1	1033540
2	210318
3	59896
4	28404
5	78088
6	48333
7	3
8	1
9	1

这个分布肯定是不合理的，多集中分布在1人和2人，而3456789（甚至能9人）占比较少。于是我们将其修改下，除了将0人增加到1人外，设置其属性为：单人出行，双人出行和多人出行。

关注信息

对于出行时间，最大值甚至长达一个月，似乎有些不合理，最小值也是0s 1s，这似乎也不合理。我们可以采用三倍极差去除这些不合理的情况，过度关注极值会带来一些不必要的影响

本文的重点就是特征工程，如何构建特征工程，一向是比赛的重点。在这里，我们将特征分为时间特征、空间特征、环境特征三大类，分别构造如下：

时间特征

- 是否节假日
- 出行日期
- 出行时间段

空间特征

- Haversine距离
- Manhattan距离
- 方位角
- 方向
- 出行方式聚类

环境特征

- 湿度
- 大气压
- 温度
- 风向
- 风速

经过构建的并使用的特征有53个。建模方面，我们采用机器学习和深度学习并行的方式，当然，深度学习在本项目中表现不佳，应该是数据的特征的问题，能用的特征实在太少了，强行拟合也只会过拟合。

看了下其他人的解决方案，也大差不差，很多时候需要添加额外的信息来让模型变得更好，在另一场类似的比赛里，1st的方案采用了如下特征：

- 工作日
- 出发的小时
- 行程长度
- 出发点到城市中心的距离
- 出发点到城市中心的朝向
- 城市中心到中断点的距离
- 城市中心到中断点的朝向
- 出行速度中位数
- 阶段点的速度和朝向

至于本场比赛超过0.37的分数，大概是用了更好的数据集，从原始数据集来看，能到0.37似乎已经是上限了。

此时我们已经构造完了原始数据集特征，其特征如下所示：

```
<class 'pandas.core.frame.DataFrame'>  
Index: 1159983 entries, 324527 to 122720  
Data columns (total 53 columns):
```

#	Column	Non-Null Count	Dtype
0	f1	1159983 non-null	uint8
1	f2	1159983 non-null	uint8
2	f3	1159983 non-null	uint8
3	f4	1159983 non-null	uint8
4	Morning	1159983 non-null	uint8
5	Afternoon	1159983 non-null	uint8
6	DeepNight	1159983 non-null	uint8
7	Night	1159983 non-null	uint8
8	Noon	1159983 non-null	uint8
9	Other	1159983 non-null	uint8
10	distance	1159983 non-null	float64
11	p_month	1159983 non-null	int32
12	p_day	1159983 non-null	int32
13	p_hour	1159983 non-null	int32
14	pickup_latitude	1159983 non-null	float64
15	pickup_longitude	1159983 non-null	float64
16	dropoff_longitude	1159983 non-null	float64
17	dropoff_latitude	1159983 non-null	float64
18	p_minofday	1159983 non-null	int32
19	p_time	1159983 non-null	float64
20	d_0.0	1159983 non-null	uint8
21	d_1.0	1159983 non-null	uint8
22	d_2.0	1159983 non-null	uint8
23	d_3.0	1159983 non-null	uint8
24	d_4.0	1159983 non-null	uint8
25	d_5.0	1159983 non-null	uint8
26	d_6.0	1159983 non-null	uint8
27	d_7.0	1159983 non-null	uint8
28	d_8.0	1159983 non-null	uint8
29	distance_man	1159983 non-null	float64
30	One	1159983 non-null	uint8
31	Two	1159983 non-null	uint8
32	Multi	1159983 non-null	uint8
33	is_weekday?_0	1159983 non-null	uint8
34	is_weekday?_1	1159983 non-null	uint8

```

34  is_weekday?_1      1159983 non-null uint8
35  C_0                1159983 non-null uint8
36  C_1                1159983 non-null uint8
37  C_2                1159983 non-null uint8
38  C_3                1159983 non-null uint8
39  C_4                1159983 non-null uint8
40  C_5                1159983 non-null uint8
41  C_6                1159983 non-null uint8
42  C_7                1159983 non-null uint8
43  C_8                1159983 non-null uint8
44  C_9                1159983 non-null uint8
45  C_10               1159983 non-null uint8
46  C_11               1159983 non-null uint8
47  direction          1159983 non-null float64
48  humidity            1153123 non-null float64
49  pressure            1153123 non-null float64
50  temperature        1153123 non-null float64
51  wind_direction     1153123 non-null float64
52  wind_speed         1153123 non-null float64
dtypes: float64(13), int32(4), uint8(36)
memory usage: 181.4 MB

```

我们采用固定参数的树模型进行处理，请注意，树模型不需要进行one-hot编码，也不需要归一化。三种模型的得分为：

随机森林

Score in training 0.69
Score in val 0.69
Finall Score 0.43


XGBoost

Score in training 0.97
Score in val 0.78
Finall Score 0.36

Light GBM

Score in training 0.79
Score in val 0.77
Finall Score 0.37





主成分变换会更好吗？

 lgb_pca_sub.csv Complete (after deadline) · now	0.48561	0.48737	<input type="checkbox"/>
---	---------	---------	--------------------------

答案是否定的，当然也有可能是参数选择问题，我们保留前三十五个主成分再试试

 lgb_pca_sub (1).csv Complete (after deadline) · now	0.47881	0.48008	<input type="checkbox"/>
---	---------	---------	--------------------------

并没有什么作用

 submission (3).csv Complete (after deadline) · 35s ago · lgbm+rgb	0.39304	0.39675	<input type="checkbox"/>
 submission (2).csv Complete (after deadline) · 3m ago · rf	0.46057	0.46391	<input type="checkbox"/>
 submission.csv Complete (after deadline) · 4m ago · xgb	0.39948	0.40235	<input type="checkbox"/>
 submission (6).csv Complete (after deadline) · 8m ago · lgbm	0.39946	0.40332	<input type="checkbox"/>

我们进行模型融合，这里是平权投票，可以看到，两个性能相似的模型融合，进一步提升了性能。

那么，如果三个性能不接近的模型呢？

Submission and Description	Private Score 	Public Score 	Selected
 submission (4).csv Complete (after deadline) · now · $0.4 \cdot \text{lgb} + 0.4 \cdot \text{xgb} + 0.2 \cdot \text{rf}$	3.27739	3.27810	<input type="checkbox"/>

这里多除了2，也可以看到，对数均方根误差对数据的容忍性很强

 submission (5).csv Complete (after deadline) · now · $0.4 \cdot \text{l} + 0.4 \cdot \text{x} + 0.2 \cdot \text{f}$	0.39642	0.40014	<input type="checkbox"/>
---	---------	---------	--------------------------

还可以啊，比任意单模强

距离剔除也不合理


```
3]:
```

```
A.corrwith(B,axis=0)
```

```
... f1          1.000000
    f2          1.000000
    f3          1.000000
    f4          1.000000
    distance    0.999999
    p_month     1.000000
    p_day       1.000000
    p_hour      1.000000
    pickup_latitude 1.000000
    pickup_longitude 1.000000
    dropoff_longitude 1.000000
    dropoff_latitude 1.000000
    p_minofday  0.999999
    p_time      1.000000
    distance_man 1.000000
    is_weekday?_0 1.000000
    is_weekday?_1 1.000000
    direction   1.000000
    passenger_count 1.000000
    humidity    1.000000
    pressure    1.000000
    temperature 1.000000
    wind_speed  1.000000
    wind_direction 1.000000
    precipitation 1.000000
    snow fall   1.000000
    snow depth  1.000000
    Afternoon   1.000000
    DeepNight   1.000000
    Morning     1.000000
    Night       1.000000
    Noon        1.000000
    Other       1.000000
    dtype: float64
```

相关性判别

	4	8	9	10	11	14
count	1425094	1425094	1425094	1425094	1425094	1425094
min	-0.895600796	-12.77976799	-12.68373203	-18.27015686	-13.53777504	-0.848189712
25%	-0.566904128	-0.495177895	-0.485669076	-0.509659886	-0.501682937	-0.542581558
50%	-0.341814399	0.109664001	-0.213477165	-0.173936948	0.085000865	-0.333218962
75%	0.121496206	0.626224935	0.170346335	0.306585908	0.568783045	0.093408026
max	20.11207581	14.71290684	17.3256321	23.3401947	21.1493454	20.05653191
	distance	pickup_latitude	pickup_longitude	dropoff_longitude	dropoff_latitude	distance_man
count	625134	625134	625134	625134	625134	625134
min	-0.803656913	-112.6159159	-653.4981682	-660.9242439	-115.8583078	-0.764963594
25%	-0.51529977	-0.453466441	-0.248504656	-0.246120232	-0.441493058	-0.495972372
50%	-0.313576698	0.106063284	-0.110759748	-0.087044951	0.07612939	-0.306996315
75%	0.105132267	0.585203746	0.084682084	0.143945395	0.503452756	0.079626765
max	241.7025679	69.15111118	64.37891109	89.253439	226.2675556	243.1913945

总结

作为我实际上全身心投入的第一场比赛，我们遇到过许多挫折，不过也得到了很大程度的成长。从一开始写Baseline都费时费力，还需要随时查阅技术文档，到后面比较得心应手，也算是成就感满满。

虽然遇到了某些问题，比如在这里数据清洗不如不做，我之前做了一些数据清洗，把出行速度太短的剔除了，结果导致效果大不如前，我还不自知。最后进行测试的时候才发现，训练集和测试集具有相同的分布，在训练集上过拟合(0.27)，但是在测试集上直接炸了(0.43)，关键用速度剔除会出现如下问题：测试集无法构建速度属性，也就无法使用掩码处理。所以在数据清洗的时候得想一想了，这个到底是异常点，还是就是数据分布的一部分？虽然从真实值的情况来讲，我们做剔除肯定要更好的(0.27)，但遗憾的是，这是数据比赛，所以啊，两个都是取自同一个Dirty数据，就别装清高了。

然后还有深度学习，一开始训练的慢，训练没有成效，然后看了下之前蘑菇预测的比赛，他们的MLP是怎么写的。然后我发现，这种1D数据有可能直接放进去训练，而不需要构建数据集分批训练。当然主要还是看显存能否吃下，在蘑菇预测那里，人家训练了一万次，所以我们也改进了策略，选择训练多次，按照这种方式，确实大大提升了训练速度，并且成功向更好的方向收敛。

关于深度学习预测值问题，我是看了LEAP那场，他们获取原始标签分布，并将其归一化，保留均值和方差，预测的结果就是一个归一化后的值，再反归一化，这样标签分布会更偏向原始标签分布。

特征构建聚类这个信息，一开始用的是上下车点构建的向量[n,4]，但是出现了训练集和测试集分布异常问题，随后我们综合了训练集和测试集的上下车点来做聚类，密度聚类DBSCAN这种爆内存了，做的KMEANS，上车点中心，下车点中心。效果有没有不好说，反正有这个特征。

然后再说就是一些技术层面的事情了，我们放在Knowledge里面说。