深圳北站周边交通拥堵指数预测大实验

队伍名：yxknmppjbjbu

171830635 俞星凯

171830583 穆朋朋

171860607 白晋斌

目录

1. 摘要（Abstract） *3*

2. 介绍（Introduction） *4*

3. 模型（Model） *4*

4. 基本理论和方法（Basic Theory and Method） *5*

4.1 集成学习 **5**

4.2 LSTM **7**

5. 方法（Method） *9*

5.1 搭建特征工程 **9**

5.2 应用回归模型 **11**

5.3 优化调整模型 **11**

6. 方法分析（Analysis of Method） *11*

6.1 缺失值处理 **12**

6.2 特征分组 **12**

6.3 参数调优 **12**

7. 算法（Algorithm） *13*

7.1 数据处理 **13**

7.2 回归模型 **13**

7.2.0 模型训练 13

7.2.1 线性模型 13

7.2.2 随机森林 13

7.2.3 xgboost 15

7.2.4 lightGBM 16

7.2.5 KNN 16

7.2.6 LSTM 17

8. 算法分析（Analysis of Algorithm） *18*

9. 数值结果（Numerical Result） *20*

10. 讨论（Discussion） *23*

11. 结论（Conclusion） *23*

12. 致谢（Acknowledgement） *23*

13. 参考文献（Reference） *24*

14. 附录（Appendix） *25*

15.其他说明 *27*

1. 摘要（Abstract）

交通拥堵预测是解决交通信息系统的一个关键点，交通信息系统可以为城市提供方便快捷的交通引导，极大地方便人们的出行。但是城市的交通都具有其自身特点，且交通流具有一定的复杂性，用代数表达式准确地来分析他的变化情况和规律是非常困难的。因此用强大的机器学习算法和模型来进行交通拥堵预测的研究具有十分重要的实际意义。

我们通过对不同地点的平均 TTI（Travel Time Index，通行时间指数）进行分析，并探究数据其他特征与 TTI 特征的相关系数，最终找到数据集中对因变量 TTI 有影响的全部自变量，其中包括前一小时的 TTI、平均速度、是否为工作日等特征。此外，我们使用华为云对 GPS 海量数据进行抽样处理，得到对应地点、时刻的网约车平均速度与订单密度，将这些数据与目标 TTI 进行相关性分析，最终与前述特征共同作为回归模型的自变量。通过对 TTI 分布的观察，我们发现 90%的 TTI 数据位于 1.0-2.0 之间，只有很少的 TTI 数据能达到 9 以上，因此，我们通过对数据集进行提取、删减，过滤掉这些离群值。

数据处理之后，我们并没有直接将其直接作为输入，而是根据之前对不同时刻、不同地点平均 TTI 的分析，尝试了数据四种分组方式，对每组数据单独训练一个模型，最终通过预测精度的对比，采用效果最好的分组方式训练模型并预测。预测方法上我们尝试了目前机器学习中已有的线性回归、kNN、LSTM、随机森林、XGBoost、lightGBM 等算法。在未使用 GPS 数据情况下，通过比较得出随机森林和 lightGBM 在本问题上表现更好的结论。并提出通过组合随机森林和 lightGBM 两种算法的方法，将两种预测结果加权平均，将单一模型无法通过调整参数提高的预测精度进一步提高，使 mae 进一步降低了 0.011。

最后进一步尝试使用 GPS 数据的得到的特征，但由于 GPS 数据量巨大，时间的限制让我们只得到了在 1/400 和 1/200 的采样率下的网约车特征，由于采样率极低，最终这些特征对模型的精度没有提升。但我们通过比较 1/200 采样率下和 1/400 采样率下的网约车特征与 TTI 的相关系数，发现采样率变为原来两倍使相关系数提升了 10 倍之多，所以我们认为提高采样率后一定可以很大程度上提高模型的预测精度。

2. 介绍（Introduction）

我们要考虑的问题是一种交通流预测问题，具体来说是使用过去时间的速度、交通拥塞指数(TTI)、网约车订单的 gps 数据预测未来的某 10 分钟的 TTI。完全相同的问题无法在现有文献中找到，只能找到一些类似问题的：

到目前为止，已经有多种用于交通流预测的模型与方法。例如传统的自回归差分移动平均（ARIMA）模型[1]、人工神经网络（ANN）模型及其变体[2-3]、K-近邻算法及其变体[4-5]、卡尔曼滤波模型[6]、支持向量回归（SVR）模型及其变体[7-8]、随机差分方程模型[9-10]。机器学习中的随机森林模型、回归树模型用于交通流预测时也能取得较好的预测精度[11]。近年来，深度学习中的卷积神经网络（CNN）模型[12]、长短时记忆（LSTM）神经网络模型[13]、深度神经网络（DNN）模型[14]、限制玻尔兹曼机（RBM）模型[15]、堆叠自编码（SAE）模型[16]在获得足够的样本训练的前提下，预测的效果往往优于传统模型。除此以外，考虑到单一预测模型的有效性与实用性难以同时满足，研究人员还通过多模型组合预测的方法来取得更好的预测效果[17-18]。综上所述，已有的交通流预测模型与方法主要分为四大类：①线性理论模型与方法；②传统机器学习模型；③新兴的深度学习方法；④多模型相结合的组合预测模型。第一类模型原理简单，且能满足流量预测的实时性，但预测精度不高；第二类和第三类模型预测精度较高，稳定性好，但需要大量的历史数据用于训练，且调参繁琐，这在实际中很难满足。

由于是一个全新的具体问题，且为了得到较好结果，我们选择使用多种模型和方法进行尝试，并将结果进行对比，尝试使用多模型组合的策略来改进预测效果。主要采用的机器学习模型有：线性回归模型、kNN、LSTM、随机森林、XGBoost、lightGBM 等。

3. 模型（Model）

本题是对道路的交通数据进行分析，希望利用以往的道路数据对未来道路某时刻的交通拥塞指数 TTI 进行预测。

表达成数学形式为：$y=f(\mathbf{x})$，其中 y 为某时刻 TTI，**x** 包含这个时刻以前的道路交通数据，本题就是让我们近似 f 这个函数。

本题实际处理的数据有：

train_TTI.csv: 训练集，数据结构为[id_road,time,speed,TTI]

toPredict_noLabel.csv： 测试集，数据结构为[id_road,time]，其中每连续的 3 条归为相同的半小时，即[t,t+30min)的 3 个时刻 t,t+10min,t+20min。

toPredict_train_TTI.csv: 提供的测试集的前一小时数据，可用于预测。数据结构为[id_road,time,speed,TTI]，其中每连续的 6 个时刻是[t-60min,t)内的以 10min 为间隔的时刻，且与测试集中某半小时的要预测数据相对应。

gps 文件: 网约车的 gps 数据，没有测试集相同时间的数据，只有与 train_TTI.csv 和 toPredict_train_TTI.csv 中记录相同时间的记录。

根据本题提供数据的特点，可以将 **y**=f(**x**)中做一些**简化：**

y=[y0;y1;y2],**x**=[…,xi,…]，**y** 为某半小时[t,t+30min)的 3 个时刻的 TTI，**x** 仅包含前一个小时[t-60min,t)的道路交通数据。

y=f(**x**)即利用前一个小时的各种数据预测后半个小时的 TTI。

我们利用数据的组合**将数据所处空间由 R^4 转换为了一个更高维的线性空间：R^n，**其中 n>=13，与提取的特征数量有关。

将前一个小时的 6 条数据组合，并利用时间、gps 等提取特征得到

x=[id_road,TTI0,…,TTI5,speed0,…,speed5，…其他特征…]

希望得到的预测值：

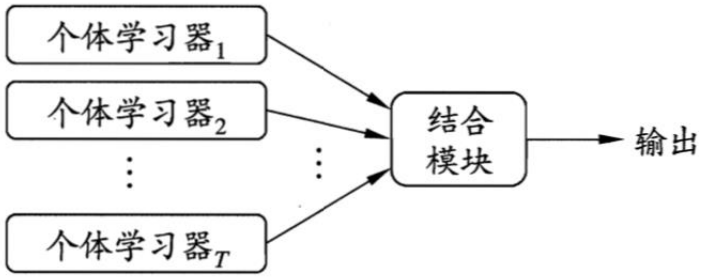
y=[TTI6,TTI7,TTI8]

所以本题被**建模成一个机器学习的回归问题**，可以通过常用的机器学习方法来进行训练，从训练集中学习得到 **y**=f(**x**)的函数关系 f，然后利用 toPredict_train_TTI.csv 中的前一小时数据进行预测。

4. 基本理论和方法（Basic Theory and Method）

4.1 集成学习

在机器学习的有监督学习算法中，我们的目标是学习出一个稳定的且在各个方面表现都较好的模型，但实际情况往往不这么理想，有时我们只能得到多个有偏好的模型（弱监督模型，在某些方面表现的比较好）。集成学习就是组合这里的多个弱监督模型以期得到一个更好更全面的强监督模型，集成学习潜在的思想是即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来。



集成学习在各个规模的数据集上都有很好的策略。当数据集比较大的时候,我们可以划分成多个小数据集，学习多个模型进行组合。当数据集比较小的时候,我们可以利用 Bootstrap 方法进行抽样，得到多个数据集，分别训练多个模型再进行组合。

根据个体学习器的生成方式，目前的集成学习方法大致可分为两大类，即个体学习器间存在强依赖关系、必须串行生成的序列化方法，以及个体学习器间不存在强依赖关系、可同时生成的并行化方法；前者的代表是 Boosting，后者的代表是 Bagging 和“随机森林（Random Forest）”。

提升方法（Boosting）是一种可以用来减小监督学习中偏差的机器学习算法。主要也是学习一系列弱分类器，并将其组合为一个强分类器。Boosting 中有代表性的是 AdaBoost（Adaptive boosting）算法：刚开始训练时对每一个训练例赋相等的权重，然后用该算法对训练集训练 t 轮，每次训练后，对训练失败的训练例赋以较大的权重，也就是让学习算法在每次学习以后更注意学错的样本，从而得到多个预测函数。另一方面，GBDT（Gradient Boost Decision Tree)也是一种 Boosting 的方法，与 AdaBoost 不同，GBDT 每一次的计算是为了减少上一次的残差，GBDT 在残差减少（负梯度）的方向上建立一个新的模型。

相比于前面的 Boosting，Bagging 和随机森林就简洁了许多。上面已经提到了，集成学习的核心是产生并组合“好而不同”的个体学习器，所以是在保证准确性的前提下，尽量的提高多样性。而 Bagging 和 随机森林（Random Forest），都是通过“自主采样”的方法来增加多样性。

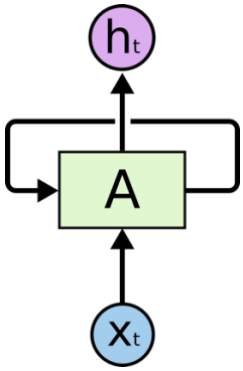
与 Boosting 不同的是，Bagging 是一种并行的集成学习方法，基学习器的训练没有先后顺序，同时进行。Bagging 采用“有放回”采样，对于包含 m 个样本的训练集，进行 m 次有放回的随机采样操作，从而得到 m 个样本的采样集，按照这样的方式重复进行，我们就可以得到 T 个包含 m 个样本的训练集，训练出来 T 个基学习器，然后对这些基学习器的输出进行结合。

随机森林（Random Forest）是 Bagging 的一个变体。它的基学习器固定是决策树，所以多棵树就叫做森林。而“随机” 体现在属性选择的随机性上。随机森林（Random Forest）在训练基学习器时候，也采用了自助取样法增加样本扰动；除此之外，随机森林还引入了一种属性扰动：对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含 K 个属性的子集，然后在子集中选取一个最优的属性用于该结点的划分。而这里的参数 K 控制了随机性的程度，令 k=d，则就是传统的决策树；令 k=1，就是随机选择一个属性用于结点划分。一般情况下，推荐的 K 值是 k=log2d。与 Bagging 相比，随机森林由于随机属性引入的多样性，使得其在多样性上有了进一步提升。相比于 Bagging，由于属性扰动的加入，其初始泛化性能较差（基决策树的准确度有一定的下降），但是随着集成数目的增多，其往往可以收敛到更低的泛化误差。同时随机森林由于属性选择，训练效率更高。

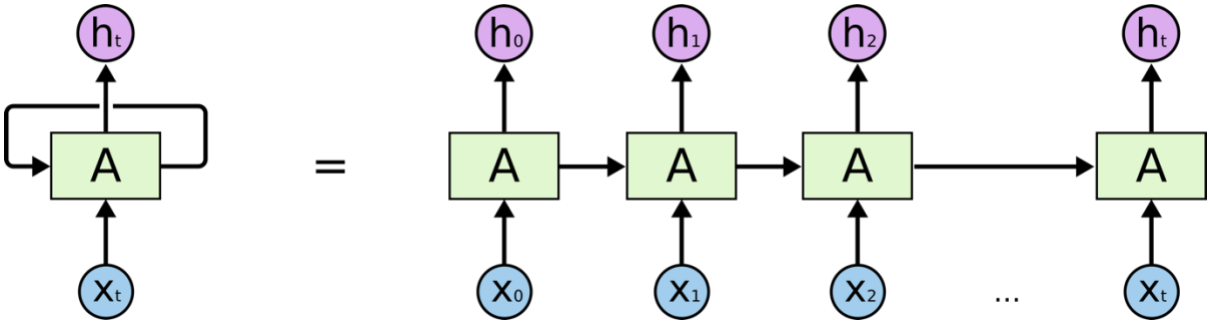
4.2 LSTM

LSTM(Long Short-Term Memory) 长短期记忆网络，是一种时间递归神经网络，适合于处理和预测时间序列中间隔和延迟相对较长的重要事件。

一般神经网络没有考虑数据的持续影响。通常，前面输入神经元的数据对后输入的数据有影响。考虑到这点或者说为了解决传统神经网络不能捕捉/利用 `previous event affect the later ones`，提出了 RNN，网络中加入循环。下图是 RNN 网络示图。



RNN 网络实质上是多个普通神经网络的连接，每个神经元向下一个传递信息，如下图所示:

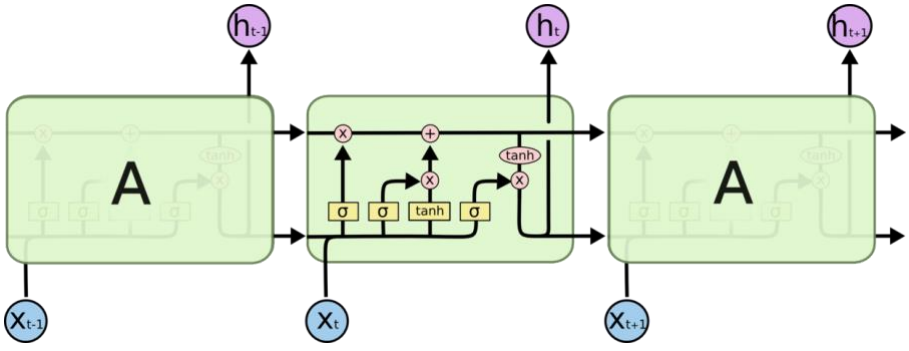


有时，我们只需要查看最近的信息来执行当前的任务。 例如，考虑一个语言模型试图根据以前的单词预测下一个词。 如果我们试图预测“the clouds are in the sky ”的最后一个词，我们不需要任何进一步的背景(上下文) - 很明显，下一个词将是 sky。 在这种情况下，当前任务训练时 RNNs 模型需要过去 n 个信息且 n 很小。

但是也有需要很多上下文信息的情况。如果我们试图预测长句的最后一个单词：Consider trying to predict the last word in the text “I grew up in France… I speak fluent French”，最近的信息 I speak fluent French 表示/提示下一个单词可能是某种语言的名称，但是如果我们缩小范围到具体某种语言时，我们需要关于 France 的背景信息。那么使用 RNNs 训练时需要过去 n 个信息，且 n 要足够大。

换句话说，传统的 RNN 模型，在训练的过程中的梯度下降过程中，更加倾向于按照序列结尾处的权值的正确方向进行更新。也就是说，越远的序列输入的对权值的正确变化所能起到的“影响”越小，所以训练的结果就是往往出现偏向于新的信息，即不太能有较长的记忆功能。

LSTM 通过刻意的设计来避免长期依赖问题，不同于 RNN 单一神经网络层，这里是有四个，以一种非常特殊的门机制进行交互。换句话说，LSTM 把原本 RNN 的单元改造成一个叫做 CEC 的部件，这个部件保证了误差将以常数的形式在网络中流动，并在此基础上添加输入门和输出门使得模型变成非线性的，并可以调整不同时序的输出对模型后续动作的影响。



5. 方法（Method）

我们的方法主要包括三个部分：

5.1 搭建特征工程

以交通流量预测知识体系为背景作为原始数据源的基础分析，找到数据集中对因变量有影响的全部自变量，并根据结果进行生成、提取、删减或者组合。

这一部分我们使用的是本题提供的原始文件：

train_TTI.csv: 训练集，数据结构为[id_road,time,speed,TTI]

toPredict_noLabel.csv： 测试集，数据结构为[id_road,time]，其中每连续的 3 条归为相同的半小时，即[t,t+30min)的 3 个时刻 t,t+10min,t+20min。

toPredict_train_TTI.csv: 提供的测试集的前一小时数据，可用于预测。数据结构为[id_road,time,speed,TTI]，其中每连续的 6 个时刻是[t-60min,t)内的以 10min 为间隔的时刻，且与测试集中某半小时的要预测数据相对应。

gps 文件: 网约车的 gps 数据，没有测试集相同时间的数据，只有与 train_TTI.csv 和 toPredict_train_TTI.csv 中记录相同时间的记录。

其中从文件 **train_TTI.csv 和 toPredict_train_TTI.csv** 中很容易提取出：道路 id_road、时间 time、平均速度 speed、交通拥塞指数 TTI 等特征，而 gps 文件的处理就非常复杂了，这也是本竞赛的一大难点，一方面数据量极大，四个文件总计 80G，另一方面价值密度很低，gps 仅仅记录时间地点速度等信息。为了利用 GPS 数据达到更好的模型效果，我们尝试将 GPS 数据进行分析处理得到网约车的平均速度与订单密度，我们设计了一种方法对之处理，如下：

GPS 数据以 5 个字段为一组，[lng,lat, speed,direction,time]，分别是传感器记录的 gps 经度、纬度、速度、方向，以及该记录的时间。题目一共有 6 个路段，每个路段包含 2 个方向，我们为每个路段选择 4 个“标记点”，记录经度纬度，一共 24 个“标记点”。对于每组数据，我们根据经纬度算出它与 24 个“标记点”的直线距离，并把它分配给最近的“标记点”所在路段。接着根据方向角度与该路段两个方向的差异，确定它的路段和方向。然后将时间戳的秒钟和分钟向下舍去，获得它的所属时间。最终，我们得到了两张以为路段和时间作为行列索引的表格，分别表示平均速度和订单个数。

至此，我们有道路 id_road、时间 time、平均速度 speed、交通拥塞指数 TTI、网约车平均速度 carSpeed、网约车订单量 count，但是这些特征中有些不太明显或不容易量化，尤其是时间需要进一步处理，方法如下：

我们将时间中小时和分钟合成为一个浮点数 **hour**，例如 7:30 转化为 7.5，用浮点数来表示一天内的时间。

根据时间中年月日来计算本日是否为节假日 **isHoliday** 和周几 **weekday**。节假日很明显会影响出行车辆，而周几可能会影响限号，也会影响车辆多少。

然后我们发现无论是训练集提供的数据，还是 GPS 数据，有些时间的记录存在数据缺失，我们就要做一个**缺失值处理，采用的方法是就近填充。**

接着就只需要根据问题建模中提供的思路，**将数据进行组合**，我们要做的是用前一个小时的数据预测后半个小时的 TTI，所以需要**将所有相邻的 9 条数据合并为一条数据。表达如下：**

[id_road,time0,speed0,TTI0,carSpeed0,count0；

id_road,hour1,speed1,TTI1,carSpeed1,count1；

id_road,hour2,speed2,TTI2,carSpeed2,count2；

id_road,hour3,speed3,TTI3,carSpeed3,count3；

id_road,hour4,speed4,TTI4,carSpeed4,count4；

id_road,hour5,speed5,TTI5,carSpeed5,count5；

id_road,hour6,speed6,TTI6,carSpeed6,count6；

id_road,hour7,speed7,TTI7,carSpeed7,count7；

id_road,hour8,speed8,TTI8,carSpeed8,count8]

(其中 hour0-hour8 为相邻的以 10 分钟为间隔的时间)

转换为：

x=[id_road,hour6,TTI0,TTI1,TTI2,TTI3,TTI4,TTI5,speed0,speed1,speed2,speed3,speed4,speed5,aveCarSpeed,aveCount]，

其中 aveCarSpeed 为 carSpeed0 到 carSpeed5 的平均值，avecount 为 count0 到 count5 的平均值。

y=[TTI6,TTI7,TTI8]

得 X 和 Y 矩阵，对测试集进行相似处理可得 X_test.

特征数据得到后，还有很多分组方法训练模型:

1)前述得到的数据整体训练一个模型

2)按 id_road 分类,训练 id_road 数量个模型

3)按 id_road 和 hour 分类,训练 id_road*hour 数量个模型

4)按 id_road、hour 和 weekday 分类,训练 id_road*hour*weekday 数量个模型

这四种方法我们均有尝试。

5.2 应用回归模型

依据分析处理后的样本数据及特征，对线性回归、随机森林、LSTM 神经网络、XGBoost、lightGBM 五种算法进行比较，验证每种算法是否适合此样本下短时交通流量的预测。只需构造不同的 model，进行 model.fit(X,Y)就可完成训练，用 model.predict(X_test)就可以得到结果。

5.3 优化调整模型

利用第一部分得到的特征和第二部分进行初步验证得到的预测结果,调整合适参数,进行模型的优化，并尝试使用不同模型进行组合，比较最后的预测结果，看是否能够提高各个模型的精确度，验证对各个模型经过一定程度的优化后,哪些模型更适合短时交通流量预测。

我们最开始使用**简单的交叉验证方法**，每月分别作为测试集，其他月作为训练集，手动寻找较好的参数和验证一些特征是否有用，发现这样调优的效果不是很明显。之后采用 **GridSearchCV 方法**和 **k 折交叉验证**对最优参数进行搜索，成绩有了一小步提升。

最终发现从 gps 中得到的特征效果不是很好，而 isHoliday 和 weekday 对模型只有非常微小的提升。得到最好成绩的是随机森林和 lightGBM，我们发现这两个算法预测的结果有一些差距，但是成绩却非常接近，非常符合集成学习中“好而不同”的原则，所以我们对两种算法进行组合，构造出的组合预测模型效果很好，相比随机森林和 lightGBM 都有较大提升。

6. 方法分析（Analysis of Method）

6.1 缺失值处理

对于缺失值处理方法，我们采用的是就近填充，还可以选择的方法有删除法、前向填充、后向填充和利用最近的两个的均值填充等等。就近填充的运行速度很快，适用性强，非常容易实现，但填充效果并不是非常好，可能不如均值填充。但由于缺失值相对很少，所以这里我们采用的就近填充是完全可以的，影响很小。如果需要改进可以尝试使用均值填充或更复杂但适合本应用问题的方法。

6.2 特征分组

对于特征分组方法，我们尝试了

1)前述得到的数据整体训练一个模型

2)按 id_road 分类,训练 id_road 数量的模型

3)按 id_road 和 hour 分类,训练 id_road*hour 数量的模型

4)按 id_road、hour 和 weekday 分类,训练 id_road*hour*weekday 数量的模型

方法 1)比较简单，但由于数据集非常大，复杂模型的训练性能非常差，需要耗费大量时间；方法 4)分非常多组，训练很多模型，数据集小且模型可以比较简单，训练速度很快，但是由于数据量很小，可能无法保留整体数据集的信息，学到的只是局部分布的信息，从而预测不准确，而且选用复杂模型的话非常容易过拟合。较好的方法是方法 2)和方法 3)，既可以保证比较好的训练效果，训练时间也不会很长，而且通过相关性分析得到 id_road、hour 和 TTI 的相关系数是比较小的，在 0.1 和 0.3 之间，可以作为分组依据(相关性特别高的特征携带重要信息，不适合分组)。

6.3 参数调优

对于参数调优，我们最初采用的是简单的交叉验证，只按月来划分训练集和测试集，而且手动改参数验证，效率低下且效果比较差；而之后采用的 k 折交叉验证方法更好，k 折交叉验证在划分训练集和测试集时会随机地分成 k 份，这里会引入随机性，多次运行选取表现都较好地最优参数，此方法得到的参数具有更好的泛化性能，而且使用 GridSearchCV 网格搜索最优参数非常方便，适用性强。

7. 算法(Algorithm)

7.1 数据处理

对整个 toPredict_train_TTI.csv 的数据进行处理后,得到了 x[i]=[id_road,TTI0,TTI1,TTI2,TTTI3,TTI4,TTI5,speed0,speed1,speed2,speed3,sppeed4,speed5,hour,isHoliday,weekday]，y[i]=[TTI6,TTI7,TTI8]的数据格式。

7.2 回归模型

7.2.0 模型训练

我们按四种方式对数据进行分组训练模型:

1)前述得到的数据整体训练一个模型

2)按 id_road 分类,训练 id_road 数量个模型

3)按 id_road 和 hour 分类,训练 id_road*hour 数量个模型

4)按 id_road、hour 和 weekday 分类,训练 id_road*hour*weekday 数量个模型

7.2.1 线性模型

通过构建损失函数，来求解损失函数最小时的参数 w 和 b。通常我们可以表达成如下公式：

$$\hat{\boldsymbol{y}} = \boldsymbol{wx} + b$$

其中,$\hat{\boldsymbol{y}}$为预测值，自变量 **x** 和因变量 **y** 是已知的，而我们想实现的是预测新增一个 x，其对应的 y 是多少。因此，为了构建这个函数关系，目标是通过已知数据点，求解线性模型中 w 和 b 两个参数。

因此该模型可以被认为是求解$\boldsymbol{w}$和 b.我们可用最小二乘法来求解$\boldsymbol{w}$和 b.

给定训练集 **x** 和 **y**,可以通过以下公式:

$$\boldsymbol{w} = \frac{\sum_{i=1}^{m} y_i(\boldsymbol{x_i} - \bar{\boldsymbol{x}})}{\sum_{i=1}^{m} \boldsymbol{x_i^2} - \frac{1}{m}(\sum_{i=1}^{m} \boldsymbol{x_i})^2}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - \boldsymbol{wx_i})$$

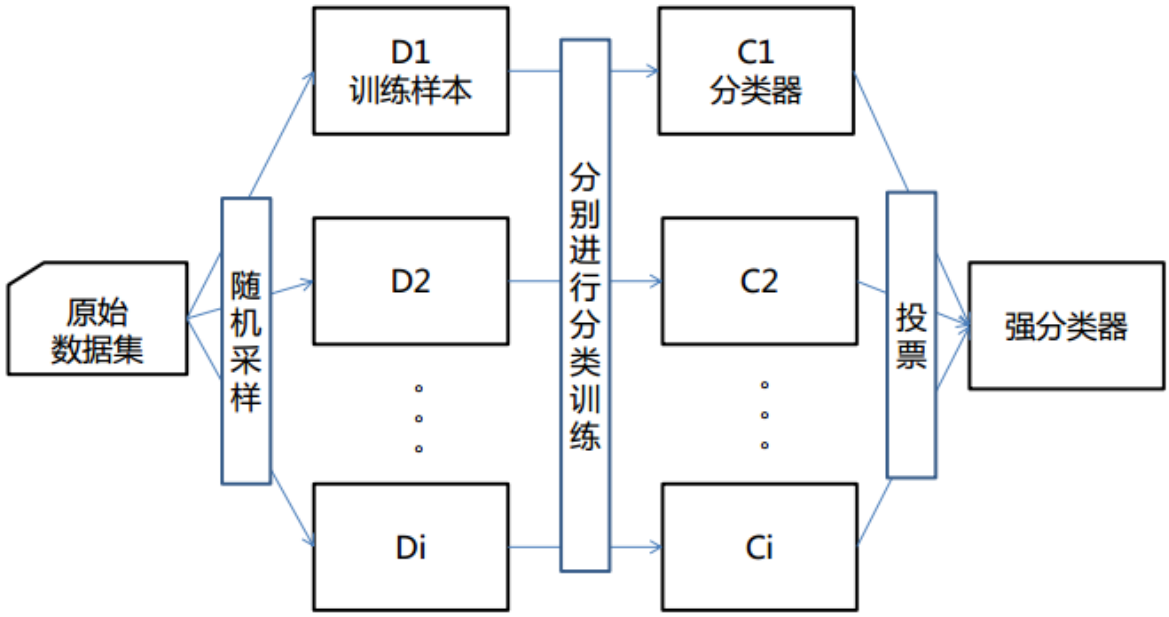
其中,$\bar{\boldsymbol{x}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x_i}$为 x 的均值.

之后在测试集通过给定的 x,带入上述求得的$\boldsymbol{w}$和 b 即可预测出测试集的 y.

7.2.2 随机森林

随机森林是一种集成算法（Ensemble Learning），它属于 Bagging 类型，通过组合多个弱分类器，最终结果通过投票或取均值，使得整体模型的结果具有较高的精确度和泛化性能。其可以取得不错成绩，主要归功于“随机”和“森林”，一个使它

具有抗过拟合能力，一个使它更加精准。



随机森林有几个主要特点:

【弱分类器】首先，RF 使用了 CART 决策树作为弱学习器。换句话说，其实我们只是将使用 CART 决策树作为弱学习器的 Bagging 方法称为随机森林。

【随机性】同时，在生成每棵树的时候，每个树选取的特征都仅仅是随机选出的少数特征，一般默认取特征总数 m 的开方。而一般的 CART 树则是会选取全部的特征进行建模。因此，不但特征是随机的，也保证了特征随机性。

【样本量】相对于一般的 Bagging 算法，RF 会选择采集和训练集样本数 N 一样个数的样本。

【特点】由于随机性，对于降低模型的方差很有作用，故随机森林一般不需要额外做剪枝，即可以取得较好的泛化能力和抗过拟合能力（Low Variance）。当然对于训练集的拟合程度就会差一些，也就是模型的偏倚会大一些（High Bias），仅仅是相对的。

CART 树:

随机森林的弱分类器使用的是 CART 数，CART 决策树又称分类回归树。当数据集的因变量为连续性数值时，该树算法就是一个回归树，可以用叶节点观察的均值作为预测值；当数据集的因变量为离散型数值时，该树算法就是一个分类树，可以很好的解决分类问题。

但需要注意的是，该算法是一个二叉树，即每一个非叶节点只能引伸出两个分支，所以当某个非叶节点是多水平(2 个以上)的离散变量时，该变量就有可能被多次使用。同时，若某个非叶节点是连续变量时，决策树也将把他当做离散变量来处理（即在有限的可能值中做划分）

7.2.3 xgboost

该算法思想就是不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到 k 棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

算法流程:

首先，对所有特征按数值进行预排序。

其次，在每次的样本分割时，用 O(# data)的代价找到每个特征的最优分割点。

最后，找到最后的特征以及分割点，将数据分裂成左右两个子节点。

xgboost 的核心在于建树,建树的主要流程包括:

枚举可能的树结构

计算结构分数 $J(\theta) = -\frac{1}{2}\sum_{t=1}^{T}[\frac{G_t^2}{H_t+\lambda}] + \gamma T$

选择分数最小树结构，并且运用最优的权重

树结构有很多可能，所以对精确搜索的情况，可采用贪心算法。

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i\in I} g_i$, $H \leftarrow \sum_{i\in I} h_i$
for $k = 1$ *to* m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

当数据太多不能装载到内存时，不能进行精确搜索分裂，可以使用近似搜索算法:

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ *to* m do
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ *to* m do
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.

7.2.4 lightGBM

常用的机器学习算法，例如神经网络等算法，都可以以 mini-batch 的方式训练，训练数据的大小不会受到内存限制。而 GBDT 在每一次迭代的时候，都需要遍历整个训练数据多次。如果把整个训练数据装进内存则会限制训练数据的大小；如果不装进内存，反复地读写训练数据又会消耗非常大的时间。尤其面对工业级海量的数据，普通的 GBDT 算法是不能满足其需求的。

LightGBM 提出的主要原因就是为了解决 GBDT 在海量数据遇到的问题，让 GBDT 可以更好更快地用于工业实践。

直方图算法的基本思想是：先把连续的浮点特征值离散化成 k 个整数，同时构造一个宽度为 k 的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了需要的统计量，然后根据直方图的离散值，遍历寻找最优的分割点。

LightGBM 是基于直方图的决策树算法, 可以简单理解为：首先确定对于每一个特征需要多少个箱子（bin）并为每一个箱子分配一个整数；然后将浮点数的范围均分成若干区间，区间个数与箱子个数相等，将属于该箱子的样本数据更新为箱子的值；最后用直方图（#bins）表示。看起来很高大上，其实就是直方图统计，将大规模的数据放在了直方图中。

7.2.5 KNN

KNN(K- Nearest Neighbor) 即 K 最邻近法,指的是在训练集中数据和标签已知的情况下，输入测试数据，将测试数据的特征与训练集中对应的特征进行相互比较，找到训练集中与之最为相似的前 K 个数据，则该测试数据对应的类别就是 K 个数据中出现次数最多的那个分类，其算法的描述为：

计算测试数据与各个训练数据之间的距离；

按照距离的递增关系进行排序；

选取距离最小的 K 个点；

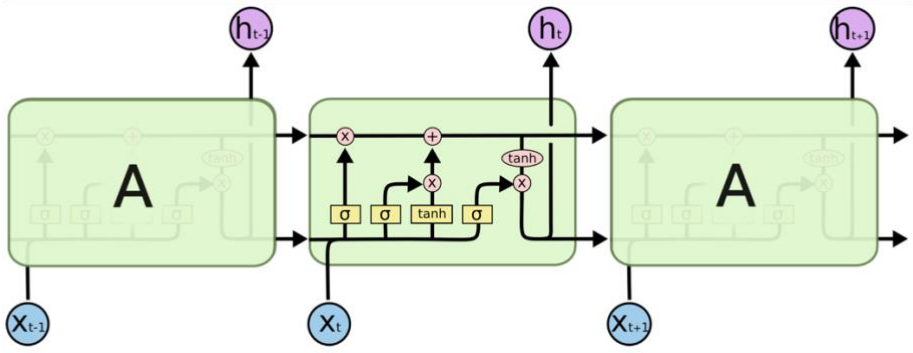
确定前 K 个点所在类别的出现频率；

返回前 K 个点中出现频率最高的类别作为测试数据的预测分类。

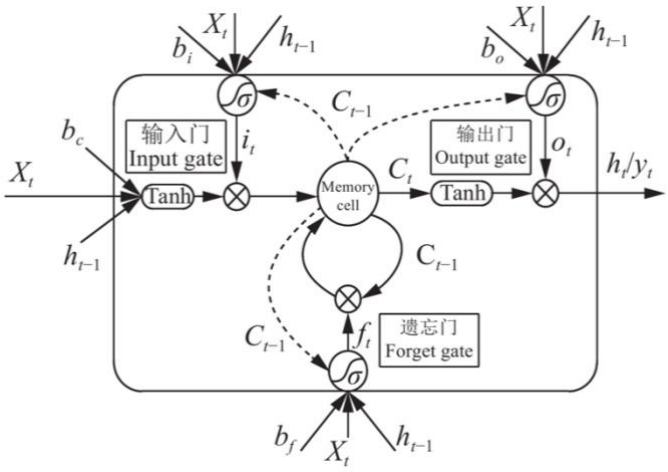
7.2.6 LSTM

传统 RNN 随着时间间隔的增加, 容易出现梯度爆炸或梯度弥散, 为解决这一问题, 提出了长短时记忆神经网络(LSTM). LSTM 采用了门限机制来控制信息的累计速度, 并可以选择性的遗忘之前的累计信息. 该模型主要包括输入门 i、输出门 o、遗忘门 f 和细胞更新状态 c. 输入门决定更新哪些信息到细胞状态中; 输出门决定细胞状态中将要输出哪些信息; 遗忘门决定细胞状态中要忘记哪些信息. 这 3 个门是控制信息流的关键, 进而解决梯度消失的问题. LSTM 模型的控制特点使其能够长时间的记忆历史数据的状态及自动匹配最佳的时间间隔.

LSTM 神经网络的结构如下图所示:



其中,LSTM 的基本储存单元架构如下图所示:



上图中, x_t , y_t分别为输入序列和输出序列, i_t, o_t, f_t分别是 t 时刻的输入门, 输出门和遗忘门, $\int$ 为激活函数. 其整个存储单元计算过程可以用如下公式表示:

$$i_t = \sigma(w_{x_i}x_t + w_{h_i}h_{t-1} + w_{c_i}c_{t-1} + b_i)$$

$$o_t = \sigma(w_{x_o}x_t + w_{h_o}h_{t-1} + w_{c_o}c_{t-1} + b_o)$$

$$f_t = \sigma(w_{x_f}x_t + w_{h_f}h_{t-1} + w_{c_f}c_{t-1} + b_f)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(w_{x_c}x_t + w_{k_c}h_{t-1} + b_c)$$

$$h_t = o_t \odot tanh(c_t)$$

式中, i_t, o_t, f_t分别是 t 时刻的输入门, 输出门和遗忘门, c_t 是 t 时刻的存储单元的计算公式, h_t是 t 时刻 LSTM 单元的所有输出. σ 和 tanh 分别代表 Sigmoid 和双曲线正切函数. w 和 b 表示相应的权重系数矩阵和偏置。

8. 算法分析 (Analysis of Algorithm)

在基于决策树的算法模型中,我们主要对比一下随机森林(RF), LightGBM 和 XGBoost 三种算法.

（1）LightGBM 使用了基于 histogram 的决策树算法，这一点不同于 XGBoost 中的贪心算法和近似算法，histogram 算法在内存和计算代价上都有不小优势。

1）内存上优势：很明显，直方图算法的内存消耗为 (#data∗#features∗1Bytes) (因为对特征分桶后只需保存特征离散化之后的值)，而 XGBoost 的贪心算法内存消耗为：(2∗#data∗#features∗4Bytes)，因为 XGBoost 既要保存原始 feature 的值，也要保存这个值的顺序索引，这些值需要 32 位的浮点数来保存。

2）计算上的优势：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 O(#data∗#feature)，而直方图算法只需要遍历桶就行了，时间为 O(#bin∗#feature)。

3）LightGBM 使用直方图做差加速，一个子节点的直方图可以通过父节点的直方图减去兄弟节点的直方图得到，从而加速计算。

4）LightGBM 支持类别特征，不需要进行独热编码处理。

5）LightGBM 优化了特征并行和数据并行算法，除此之外还添加了投票并行方案。

6）LightGBM 采用基于梯度的单边采样来减少训练样本并保持数据分布不变，减少模型因数据分布发生变化而造成的模型精度下降。

（2）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，带来了不必要的开销。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。

XGBoost 在每一层都动态构建直方图，因为 XGBoost 的直方图算法不是针对某个特定的特征，而是所有特征共享一个直方图(每个样本的权重是二阶导)，所以每一层都要重新构建直方图，而 LightGBM 中对每个特征都有一个直方图，所以构建一次直方图就够了。

（3）RF 的起始性能较差，特别当只有一个基学习器时，随着学习器数目增多，随机森林通常会收敛到更低的泛化误差。随机森林的训练效率也会高于 Bagging，因为在单个决策树的构建中，Bagging 使用的是‘确定性’决策树，在选择特征划分结点时，要对所有的特征进行考虑，而随机森林使用的是‘随机性’特征数，只需考虑特征的子集.

RF 的优点包括:

1) 由于采用了集成算法，本身精度比大多数单个算法要好；

2)在测试集上表现良好，由于两个随机性（样本随机、特征随机）的引入，使得随机森林不容易陷入过拟合（样本随机，特征随机）；

3)在工业上，由于两个随机性的引入，使得随机森林具有一定的抗噪声能力，对比其他算法具有一定优势；

4)由于树的组合，使得随机森林可以处理非线性数据，本身属于非线性分类（拟合）模型；

5)它能够处理很高维度（feature 很多）的数据，并且不用做特征选择，对数据集的适应能力强：既能处理离散型数据，也能处理连续型数据，数据集无需规范化；

6)训练速度快，可以运用在大规模数据集上；

7)可以处理缺省值（单独作为一类），不用额外处理；

8)由于有袋外数据（OOB），可以在模型生成过程中取得真实误差的无偏估计，且不损失训练数据量；

9)在训练过程中，能够检测到 feature 间的互相影响，且可以得出 feature 的重要性，具有一定参考意义；

10)由于每棵树可以独立、同时生成，容易做成并行化方法；

11)由于实现简单、精度高、抗过拟合能力强，当面对非线性数据时，适于作为基准模型。

RF 的缺点包括:

在噪声较大的分类或者回归问题上会过拟合。

9. 数值结果（Numerical Result）

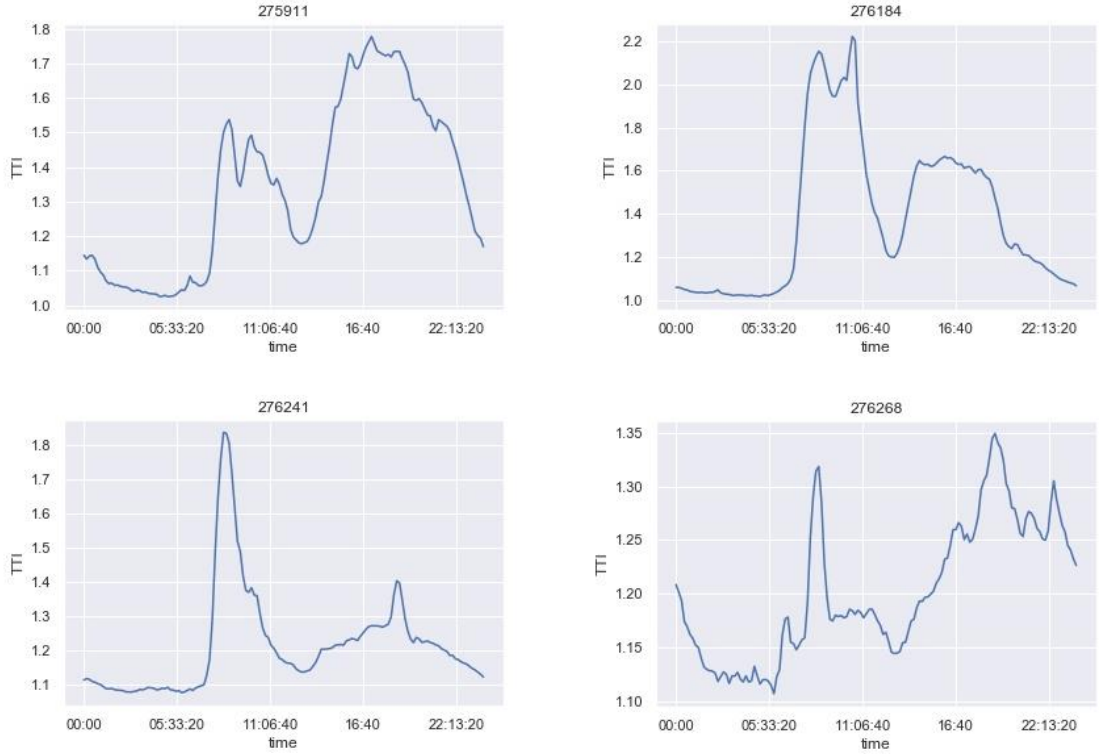
软件方面，我们选择 Python 作为编程语言，使用 Jupyter Lab 作为开发环境，用到 numpy, pandas, matplotlib, sklearn, xgboost,等科学计算与机器学习库。硬件方面，我们同时使用本地电脑以及华为云 OBS 和 ModelArts 服务，其中云服务器主要用来运行大规模数据（GPS）处理和大型模型训练（LSTM），而本机用来执行其他所有任务。原始数据有四列，包含路段、TTI、速度、时间。

	id_road	TTI	speed	time
0	276183	1.03472	69.6197	2019-01-01 00:00:00
1	276183	1.03506	69.6275	2019-01-01 00:10:00
2	276183	1.04166	69.1003	2019-01-01 00:20:00
3	276183	1.02642	70.1266	2019-01-01 00:30:00
4	276183	1.01442	70.9565	2019-01-01 00:40:00
...	...	...	...	...
296193	276738	1.44719	28.3267	2019-12-21 23:10:00
296194	276738	1.35854	30.1750	2019-12-21 23:20:00
296195	276738	1.37941	29.7186	2019-12-21 23:30:00
296196	276738	1.21284	33.7999	2019-12-21 23:40:00
296197	276738	1.21629	33.7043	2019-12-21 23:50:00

探索性数据分析，观察 TTI 和速度的分布，发现 TTI 的分布很不均匀，有极少部分特别大的值，绝大多数都处于 0.7~2 之间。

	TTI	speed
count	296198.000000	296198.000000
mean	1.384004	44.075114
std	0.663539	14.193853
min	0.661631	1.073000
25%	1.096500	37.090150
50%	1.182850	43.149900
75%	1.433515	49.822425
max	48.910300	80.207900

对不同路段的不同时刻的 TTI 进行观察，其中 TTI 根据每天该路段该时刻的平均值得到。下面选择了四个例子，容易发现不同路段的 TTI 模式存在明显差异，这提示我们不同路段应当分别处理。



经过我们的数据变换，格式变成如下样子，并且使用多重索引，非常灵活方便。

id_road	time	date	TTI0	TTI1	TTI2	TTI3	TTI4	TTI5	speed0	speed1	speed2	speed3	speed4	speed5	TTI6	TTI7	TTI8	isHoliday	weekday
275911	00:00:00	2019-01-02	1.14105	1.13699	1.12393	1.15026	1.13117	1.11357	63.1037	63.3287	64.0646	62.5981	63.6550	64.6607	1.09580	1.10014	1.11471	0.0	2
		2019-01-03	1.53843	1.29012	1.16313	1.13825	1.37852	1.91348	46.8037	55.8121	61.9055	63.2586	52.2333	37.6436	2.08315	2.04741	2.08087	0.0	3
		2019-01-04	1.20683	1.29577	1.24171	1.15601	1.14226	1.15313	59.6641	55.5687	57.9878	62.2869	63.0367	62.4427	1.15265	1.30350	1.86703	0.0	4
		2019-01-05	1.58394	1.47470	1.25195	1.15187	1.16729	1.18005	45.4591	48.8351	57.5137	62.5106	61.6852	61.0182	1.15340	1.14601	1.15873	1.0	5
		2019-01-06	2.32614	1.51375	1.22063	1.20102	1.20443	1.12525	30.9755	47.5669	58.9894	59.9527	59.7828	63.9898	1.16382	1.16347	1.14915	1.0	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
276738	23:50:00	2019-12-16	1.28309	1.36590	1.37705	1.38399	1.20483	1.34748	31.9496	30.0126	29.7696	29.6202	34.0249	30.4227	1.42428	1.43791	1.36372	0.0	0
		2019-12-17	1.21852	1.42880	1.36848	1.26464	1.25062	1.32827	33.6425	28.6912	29.9559	32.4157	32.7790	30.8628	1.24253	1.26758	1.31894	0.0	1
		2019-12-18	1.39183	1.42769	1.41937	1.32931	1.29322	1.17574	29.4534	28.7136	28.8818	30.8386	31.6993	34.8667	1.28191	1.30337	1.18777	0.0	2
		2019-12-19	1.26962	1.34787	1.25425	1.27220	1.23676	1.27166	32.2884	30.4140	32.6842	32.2230	33.1464	32.2367	1.22254	1.28557	1.17445	0.0	3
		2019-12-20	1.35850	1.50126	1.84028	1.62766	1.31815	1.28073	30.1759	27.3064	22.2760	25.1859	31.0996	32.0085	1.38232	1.30534	1.19134	0.0	4

并且我们为维度两两之间计算了相关系数，得到相关系数矩阵，容易看出前一小时的 TTI 和速度都与需要预测的后半小时 TTI 很大相关，而是否假日和周几却关联较小。

	TTI0	TTI1	TTI2	TTI3	TTI4	TTI5	speed0	speed1	speed2	speed3	speed4	speed5	TTI6	TTI7	TTI8	isHoliday	weekday
TTI0	1.000000	0.913943	0.825248	0.762461	0.715100	0.674471	-0.646792	-0.617892	-0.585991	-0.559919	-0.537530	-0.517653	0.636325	0.598135	0.559219	-0.051272	-0.008241
TTI1	0.913943	1.000000	0.914789	0.826398	0.763876	0.715793	-0.621196	-0.649624	-0.620813	-0.588929	-0.562889	-0.540310	0.675446	0.636764	0.596449	-0.052615	-0.008486
TTI2	0.825248	0.914789	1.000000	0.914602	0.826465	0.763680	-0.589862	-0.622021	-0.650725	-0.621969	-0.590149	-0.564030	0.715736	0.674705	0.633910	-0.053274	-0.008279
TTI3	0.762461	0.826398	0.914602	1.000000	0.914404	0.826163	-0.562989	-0.589959	-0.622274	-0.651186	-0.622446	-0.590596	0.763292	0.714722	0.672087	-0.053408	-0.007883
TTI4	0.715100	0.763876	0.826465	0.914404	1.000000	0.914370	-0.538843	-0.562997	-0.590085	-0.622564	-0.651559	-0.622814	0.826007	0.762357	0.712298	-0.053680	-0.007840
TTI5	0.674471	0.715793	0.763680	0.826163	0.914370	1.000000	-0.516412	-0.538482	-0.562764	-0.589994	-0.622539	-0.651592	0.914214	0.825035	0.760211	-0.053786	-0.007731
speed0	-0.646792	-0.621196	-0.589862	-0.562989	-0.538843	-0.516412	1.000000	0.978521	0.956265	0.937033	0.919127	0.902554	-0.494271	-0.471746	-0.447997	0.070694	0.020180
speed1	-0.617892	-0.649624	-0.622021	-0.589959	-0.562997	-0.538482	0.978521	1.000000	0.978522	0.956274	0.937068	0.919153	-0.515670	-0.493066	-0.468666	0.071230	0.020310
speed2	-0.585991	-0.620813	-0.650725	-0.622274	-0.590085	-0.562764	0.956265	0.978522	1.000000	0.978519	0.956290	0.937076	-0.537822	-0.514480	-0.489845	0.071617	0.020318
speed3	-0.559919	-0.588929	-0.621969	-0.651186	-0.622564	-0.589994	0.937033	0.956274	0.978519	1.000000	0.978524	0.956287	-0.562210	-0.536719	-0.511325	0.071838	0.020253
speed4	-0.537530	-0.562889	-0.590149	-0.622446	-0.651559	-0.622539	0.919127	0.937068	0.956290	0.978524	1.000000	0.978520	-0.589502	-0.561134	-0.533523	0.072040	0.020283
speed5	-0.517653	-0.540310	-0.564030	-0.590596	-0.622814	-0.651592	0.902554	0.919153	0.937076	0.956287	0.978520	1.000000	-0.622068	-0.588422	-0.557786	0.072224	0.020312
TTI6	0.636325	0.675446	0.715736	0.763292	0.826007	0.914214	-0.494271	-0.515670	-0.537822	-0.562210	-0.589502	-0.622068	1.000000	0.913962	0.823352	-0.053674	-0.007563
TTI7	0.598135	0.636764	0.674705	0.714722	0.762357	0.825035	-0.471746	-0.493066	-0.514480	-0.536719	-0.561134	-0.588422	0.913962	1.000000	0.913163	-0.053387	-0.007687
TTI8	0.559219	0.596449	0.633910	0.672087	0.712298	0.760211	-0.447997	-0.468666	-0.489845	-0.511325	-0.533523	-0.557786	0.823352	0.913163	1.000000	-0.052727	-0.007747
isHoliday	-0.051272	-0.052615	-0.053274	-0.053408	-0.053680	-0.053786	0.070694	0.071230	0.071617	0.071838	0.072040	0.072224	-0.053674	-0.053387	-0.052727	1.000000	0.637761
weekday	-0.008241	-0.008486	-0.008279	-0.007883	-0.007840	-0.007731	0.020180	0.020310	0.020318	0.020253	0.020283	0.020312	-0.007563	-0.007687	-0.007747	0.637761	1.000000

我们使用这些数据作为训练集，根据 7.2.0 中提出的四种数据选择办法，以及模型所采用的机器学习算法，得到了多组结果。

数据选择方法	机器学习算法	测试得分
使用全部数据	均值预测	0.2266
使用全部数据	K 近邻	0.1365
按 id_road、time 分组	随机森林	0.1324
按 id_road、hour, weekday 分组	LSTM	0.2037
按 id_road 分组	线性模型	0. 1445
按 id_road 分组	随机森林	0.1216

按 id_road 分组	lightGBM	0.1220
按 id_road 分组	lightGBM 和随机森林组合预测模型	0.1205

10. 讨论（Discussion）

我们对上面的结果进行分析，最好的结果由 lightGBM 和随机森林结合产生，这不是偶然的。因为我们发现这两者本身与真实值的 MAE 都已经达到 0.12 多，但是这两者之间的 MAE 却也有 0.08，这恰恰符合了集成学习“好而不同”的思想，将它们结合很可能得到更加出色的结果，事实也印证了我们的想法。同时，可以发现按 id_road 分组产生的模型较好，这是因为如果进一步按 id_road 和 time 分组，那么每组仅有 172 个数据，规模较小，稍微复杂的学习算法很容易就对其过拟合，导致泛化误差较大。而且我们通过实验表明，粗粒度分组的模型训练时间少于多个细粒度分组的模型训练时间之和。这两点表明按 id_road 分组更为合适。

11. 结论（Conclusion）

总的来说，我们首选对数据进行了探索和清洗，然后将问题建模成基于少量历史数据预测三个实数值，接着对训练数据的选择方法分为四类，并且结合不同的学习算法进行预测，得到多种结果，最终我们集成较好结果得到最终成果。我们发现对于不同路段分组，并且使用 lightGBM 和随机森林的综合结果效果最佳，这在测试集达到了 0.1205 的 mae，可谓相当准确。

尽管我们的方法已经取得了不错的结果，但是仍然有一些可以改进之处。一是由于时间限制，我们对 GPS 数据进行了非常稀疏的抽样，这使得我们提取的特征不太准确。二是很多专门用于时间序列的算法我们使用较少（例如 ARIMA），而是更加倾向于将该任务当作具有三个输出值的回归任务。三是数据清洗部分，对于缺失数据我们使用近邻填充的方式，如果使用其他插值方法也许更好。

12. 致谢（Acknowledgement）

感谢比赛主办方，感觉为 sklearn 库作出贡献的各位程序员，感谢机器学习这门课的任课老师与助教们！

13. 参考文献（Reference）

[1]Khandelwal I , Adhikari R , Verma G . Time Series Forecasting Using Hybrid ARIMA and ANN Models Based on DWT Decomposition[J]. Procedia Computer Science, 2015, 48:173-179.

[2]Kumar K , Parida M , Katiyar V K . Short Term Traffic Flow Prediction for a Non Urban Highway Using Artificial Neural Network[J]. Procedia - Social and Behavioral Sciences, 2013, 104:755-764.

[3]Zhang L D , Jia L , Zhu W X . Overview of traffic flow hybrid ANN forecasting algorithm study[C], International Conference on Computer Application & System Modeling. IEEE, 2010.

[4]Ryu U , Wang J , Kim T , et al. Construction of traffic state vector using mutual information for short-term traffic flow prediction[J]. Transportation Research Part C: Emerging Technologies, 2018, 96:55-71.

[5]Shifen C , Feng L , Peng P , et al. Short-term traffic forecasting: An adaptive ST-KNN model that considers spatial heterogeneity[J]. Computers, Environment and Urban Systems, 2018:S0198971518300140-.

[6]杨兆升, 朱中. 基于卡尔曼滤波理论的交通流量实时预测模型[J]. 中国公路学报, 1999(3):63-67.

[7]Lippi M , Bertini M , Frasconi P . Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning[J]. IEEE Transactions on Intelligent Transportation Systems, 2013, 14(2):871-882.

[8]Castroneto M , Jeong Y S , Jeong M K , et al. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions[J]. Expert Systems with Applications An International Journal, 2009, 36(3):6164-6173.

[9]Tahmasbi R , Hashemi S M . Modeling and Forecasting the Urban Volume Using Stochastic Differential Equations[J]. IEEE Transactions on Intelligent Transportation Systems, 2014, 15(1):250-259.

[10]Rajabzadeh Y, Rezaie A H, Amindavar H. Short-term traffic flow prediction using time-varying Vasicek model[J]. Transportation Research Part C: Emerging Technologies, 2017, 74: 168-181.

[11]Hou Y , Edara P , Sun C . Traffic Flow Forecasting for Urban Work Zones[J]. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(4):1761-1770.
[12]张伟斌. 基于时空分析和 CNN 的交通流量短时预测方法[A]. 中国智能交通协会.第十 三届中国智能交通年会大会论文集[C].中国智能交通协会:中国智能交通协会,2018:13.
[13]Yang B, Sun S, Li J, et al. Traffic Flow Prediction Using LSTM with Feature Enhancement[J]. Neurocomputing, 2018.
[14]Wu Y, Tan H, Qin L, et al. A hybrid deep learning based traffic flow prediction method and its understanding[J]. Transportation Research Part C: Emerging Technologies, 2018, 90: 166-180.
[15]Kong F , Li J , Jiang B , et al. Short-term traffic flow prediction in smart multimedia system for Internet of Vehicles based on deep belief network[J]. Future Generation Computer Systems, 2018.
[16]Zhou T , Han G , Xu X , et al. δ -agree AdaBoost stacked autoencoder for short-term traffic flow forecasting[J]. Neurocomputing, 2017, 247.
[17]Zulong D , Dafang Z , Xin W , et al. A Hybrid Model For Short-Term Traffic Volume Prediction In Massive Transportation Systems[J]. IEEE Transactions on Intelligent Transportation Systems, 2018:1-12.
[18]Guo J , Huang W , Williams B M . Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification[J]. Transportation Research Part C, 2014, 43:50-64.

14. 附录（Appendix）

代码文件说明：

main2.py:

完成简单预测，使用 7 到 22 点的数据和线性回归模型、决策树模型，包括简单的数据预处理。

code2.ipynb:

本文件是核心代码。

进行了比较复杂的数据预处理、特征提取，划分训练集为训练集和验证集并简单调参，训练和输出结果。

使用复杂的集成学习算法——随机森林和 lightGBM。

code3.ipynb:

对 lightGBM 使用交叉验证和 GridSearchCV 调参，使用模型成绩提升。

code4.ipynb:

使用由华为云处理 gps 数据得到网约车的订单数和平均速度特征，重新进行数据预处理、特征提取，训练和输出结果。

process.ipynb:

对数据探索性分析，进行数据清洗，把数据变换成需要的格式。

predict.ipynb:

使用不同学习算法进行预测，包括 K 近邻、随机森林和 XGBoost。

gps.ipynb:

跟 GPS 数据有关的代码，放置华为云端运行处理。

split2weekday0.py:

对之前生成的文件 train_TTI.csv 做了一个小处理按周分成了 7 份表格,便于后续处理.

split2weekday1.py:

对之前生成的文件 train_TTI_6plus3.csv 做了一个小处理按周分成了 7 份表格,便于后续处理.

lstm000.py:

应用 LSTM 方法对训练集进行训练并预测结果.其中,模型输入为前六个时刻的 TTI,输出为后三个时刻的 TTI.每个模型所用到的数据为同线路同时刻同星期数的 24 条数据.其中,preprocess()函数主要是对 **split2weekday0.py** 和 **split2weekday1.py** 生成的文件格式进一步处理为输入输出序列的标准格式,load()函数为读取给定文件名的文件,并按照给定比例划分训练集和验证集, lstm()函数为文件的主要部分,实现了从读取数据到模型训练再到预测并写入文件的全过程.

lstm001.py:

应用 LSTM 方法对训练集进行训练并预测结果.其中,模型输入为前六个时刻的 TTI,输出为后三个时刻的 TTI. 每个模型所用到的数据为同线路同时刻的 170 余条数据.

lstm002.py:

应用 LSTM 方法对训练集进行训练并预测结果.其中,模型输入为前六个时刻的 TTI,输出为后一个时刻的 TTI. 每个模型所用到的数据为同线路同时刻同星期数的 24 条数据.

lstm003.py:

应用 LSTM 方法对训练集进行训练并预测结果.其中,模型输入为前六个时刻的 TTI,输出为后一个时刻的 TTI. 每个模型所用到的数据为同线路同时刻的 170 余条数据.

15.其他说明

小组队员与分工

穆朋朋：数据预处理；特征提取；主要使用按路段分组的方法，主要使用随机森林和 lightGBM 算法进行预测；使用交叉验证和 GridSearchCV 对模型进行调参；尝试组合多种预测算法进行预测；写报告。

俞星凯：主要的数据预处理；主要使用按道路和时刻分组的方法，主要使用随机森林和 xgboost 算法进行预测；使用华为云计算 gps 相关特征：订单数和平均速度；写报告。

白晋斌：数据预处理；主要使用按照按道路、时间和周几分组的方法，主要使用 LSTM 模型进行预测；写报告。