所在组别	2021 年中国高校大数据挑战赛	参赛编号
本科生		bdc210819

基于 DBSCAN 和 LSMT 算法

对运营商基站 KPI 指标数据进行异常处理

摘要

智能运维技术以大数据平台和机器学习算法平台为核心，将监控、服务台、自动化系统联动，大大提升了 IT 系统的运营管理效率。异常检测、异常预测、趋势预测对智能运维而言至关重要。本文根据题目所提供的的运营商基站 KPI 的性能指标数据，针对小区内的平均用户数、小区 PDCP 流量、平均激活用户数三个指标，将基于 Kmeans 聚类、DBSCAN 聚类、LSTM 算法等机器学习算法，找出异常数值并建立预测模型。

针对问题一，利用专家知识和数据在时序上存在周期性的特点，对数据特征进行选择和衍生，对不同量纲的数据进行 Zscore 标准化，以增强模型的迭代速度与精度。采用 Kmeans 聚类方法，分别将平均用户数、PDCP 流量、平均激活用户数三个指标上表现相似的小区划分为同一类。在此基础上，针对每一类小区，通过 DBSCAN 聚类筛选出异常数值并统计其数量；根据异常值的时间分布确定异常周期数量：小区平均用户数的有 **202 个异常值、43 个异常周期**；小区 PDCP 流量有 **218 个异常值、36 个异常周期**；平均激活用户数有 **239 个异常值、33 个异常周期。**

针对问题二，在同时考虑输入数据时间跨度与输出数据时间跨度的情况下，使用 LSTM 模型对时间序列异常数值进行预测。根据问题一的结果，剔除数据集中的异常值点，对同类小区数据合并求得**均值**，得到每个指标下的三个数据集。随后进行数据预处理与特征工程得到训练数据，将三个指标下共 **9 个数据集**分别输入模型，得到三类指标的预测结果。通过参数调整和模型优化，得到 **RMSE** 最低可达 **0.08** 的 LSTM 模型，此时**神经网络层数为 50、输入数据时间跨度**为 **36 小时、输出数据时间跨度为 72 小时**。设定**阈值**，将大小位于后 95%的残差对应的数据判定为异常值，在三个指标下可以得到异常数值的数量为 **1914、1871、1861**。

针对问题三，利用问题二训练完成的模型，预测未来 72 个时刻三类小区的三类指标大小，以每一类小区的预测结果代表该类别下所有小区的指标，作为最终预测结果。同时，针对预测结果对于三类小区数据进行分析，并总结出三类小区用户**身份特征**，企业进行为用户画像提供了潜在方向。

关键词　智能运维 异常值预测 Kmeans 聚类 DBSCAN 聚类 LSTM 时间序列分析

一、问题重述

1.1 问题背景

《2021 年中国 ICT 技术成熟度曲线报告》显示，智能运维市场将持续增长并影响整个 IT 运营管理市。随着企业将更多的 IT 运营功能自动化以增强其灵活性，用户对智能运维平台功能的需求将不断增长。异常检测、异常预测、趋势预测，是智能运维中首先需要解决的重要问题。

虽然不同场景下运维的分析指标种类差异很大，但其数据均具有时序性特点。本赛题以运营商基站 KPI 性能指标为研究数据，选取了从 2021 年 8 月 28 日 0 时至 9 月 25 日 23 时共 29 天 5 个基站覆盖的 58 个小区对应的 67 个 KPI 指标，并选取了小区内平均用户数、小区 PDCP 流量、平均激活用户数作为核心指标进行分析。

1.2 问题提出

在本题中，我们需要根据运营商基站 KPI 性能指标的数据建立模型，解决下列问题：

1. 异常检测：对数据进行分析，找出三个核心指标的异常孤立点个数和异常周期个数。
2. 异常预测：选择恰当的输入时间跨度和输出时间跨度，基于异常数值前的数据建立模型，预测未来是否会发生异常数据。
3. 趋势预测：基于所建立的预测模型，对未来三天的三个核心指标取值进行预测。

二、问题分析

2.1 问题一分析

对于问题一，我们需要对附件一中三个核心特征的总体分布进行描述性统计，了解其总体分布。考虑到不同小区的数据在时序性上呈现出不同的规律特点，首先采用 Kmeans 聚类，对不同小区进行分类，使在三个核心特征上表现相似的小区归为同一类。其次，采用 DBSCAN 聚类算法，对每个特征每个类别的小区数据建立并拟合模型，找到异常数值。

2.2 问题二分析

对于问题二，我们利用三个核心指标将 58 个小区分为三类，并剔除问题一中预测的异常值点。通过将同类小区数据合并求得均值，得到每个指标下的三个数据集，长度为 29 天 24 个时刻的时间跨度。随后建立 LSTM 模型，并输入得到的九个数据集。通过输入数据学习长期依赖信息，可以有效预测在未来一定时间内的三个指标的数值。最后通过设定阈值，即可筛选出三个核心指标的异常值点。

2.3 问题三分析

问题三沿用问题二的模型和思路，将 LSTM 模型的步长设置为 72，并将测试集数据按照小区类别分为三类，对三个指标共计九个数据集分别预测，得到每类的结果作为该类别下所有小区的预测结果。

三、模型基本假设

1. 假设不同小区间用户使用产品的习惯存在差异，且不同的小区可被划分为有限的类别。
2. 异常周期假设为:若一段时间内，异常值发生次数≥2，且任意相邻两次异常发生间隔都≤1 小时，则该段时间视为一个异常周期。

四、符号说明

表 1 符号说明

符号	符号说明
μ	数据均值
σ	数据方差
Xi	第 i 个样本
Xit	第 i 个样本的第 t 个属性
k	Kmeans 聚类中心个数
Ci	Kmeans 的各聚类中心
Cjt	Kmeans 的第 j 个聚类中心的第 t 个属性
Si	Kmeans 的各聚簇
eps	DBSCAN 聚类的领域半径
min_samples	DBSCAN 聚类的最小邻居个数
stepin	LSTM 输入数据时间跨度
stepout	LSTM 输出数据时间跨度
layer	神经网络层数
Y_predict	目标变量预测值
Y_actual	目标变量实际值
threshold	阈值

五、问题一的模型建立与求解

5.1 建模思路

不同小区在 KPI 业务指标的表现存在差异，因此本题的思路可分为两部分：第一部分使用 Kmeans 聚类法对小区进行分类，对于每个核心指标，将表现相似的小区划分为同一类。第二部分，针对 Kmeans 聚类法的划分结果，对每一类使用 DBSCAN 聚类法找出噪声点。

5.2 数据预处理

数据预处理是建立模型前至关重要的一步，对模型的最终结果有决定性作用。针对问题一的数据需求，主要进行以下预处理。

5.2.1 特征选择和变量衍生

模型中特征的数量往往决定了泛化能力。在模型中使用过多的特征会使模型变得更加复杂，从而增大过拟合的可能性，减低其泛化能力。在建立模型时只包含较为有用的特征，适当删除其余特征，以提高模型性能、得到解释性更强的模型。

此外，在特征工程中利用专家知识[1]，我们发现一天内的不同时段是影响运营商基站 KPI 性能指标的重要因素。三个核心指标与日期之间的相关性较弱、但随时段存在强周期性，如图 1。因此，在附件 1 的“时间”特征中仅提取“时段”信息作为衍生变量。

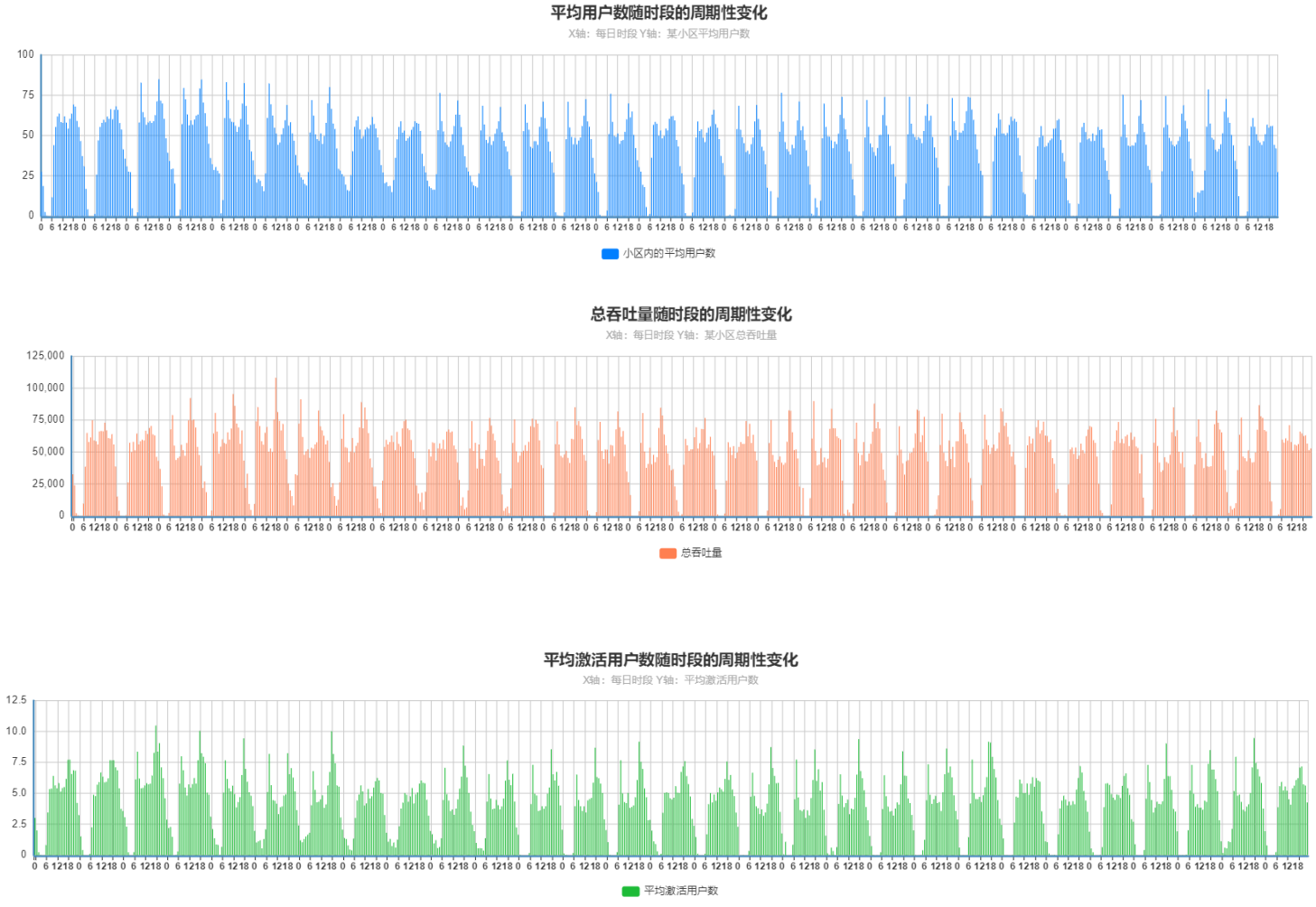


图 1 三种指标随时间变化图

5.2.2 数据标准化

不同数据的量纲存在较大差异，为提后续模型的迭代速度与精度，对数据采用基于数据均值和标准差的 Z-score 法进行缩放。

$$\frac{x-\mu}{\sigma}$$

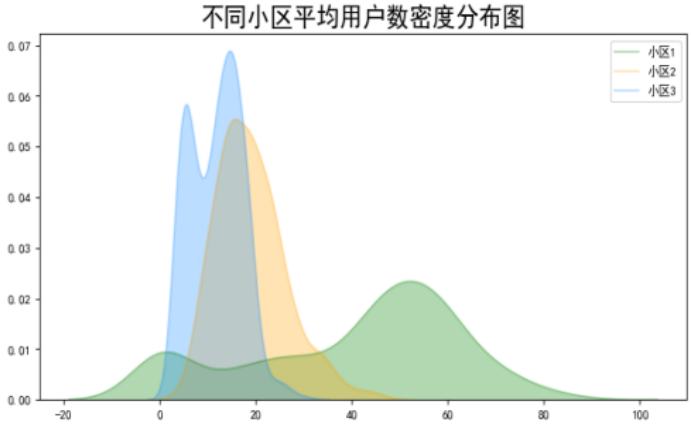
其中 μ 是数据均值， σ 是数据的标准差。

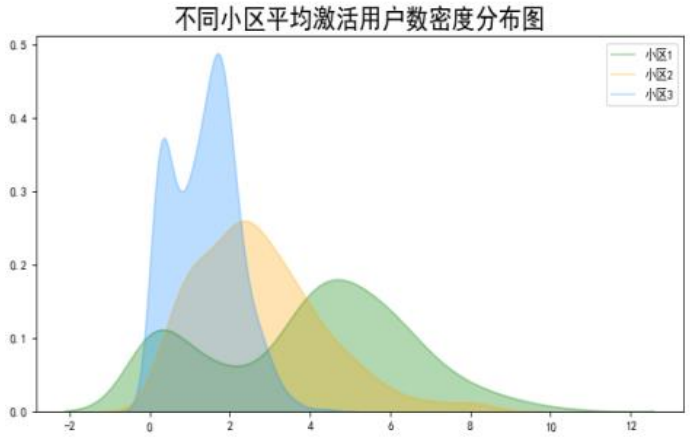
5.3 数据分析

为了更好地建立模型找出异常点，下面通过 Kmeans 聚类法和 DBSCAN 聚类法，对数据进行了进一步分析。

5.3.1 三个核心指标的数据分布

通过绘制三个核心指标在不同小区的数据密度分布图，可以观察到不同小区的数据分布存在显著差异。这可能与不同小区的地段繁华程度、住户社会层次、用户平均年龄等多种因素有关。若将所有小区全部放入同一模型中，模型的拟合度欠佳。因此，我们首先使用 Kmeans 聚类算法，将数据表现相似的小区划分为同一聚簇。





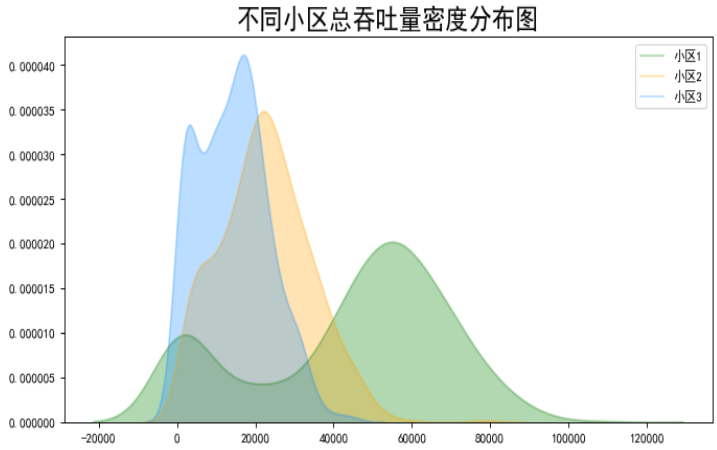


图 2 三种指标不同小区分布图

5.3.2 Kmeans 聚类

我们采用 Kmeans 聚类的方法，分别基于三个核心特征，对小区进行分类。

1）理论基础

在给定 k 值和 k 个初始类簇中心点的情况下，算法交替执行以下两个步骤：将每个数据点分配给最近的簇中心；将每个簇中心设置为所分配的所有数据点的平均值。当簇的分配不再发生变化或达到指定迭代次数时，算法结束。

我们将每个小区作为一条样本数据，将其各日各时段的平均用户数、总吞吐量、平均激活用户数作为其属性。Kmeans 算法的目的是将 59 个小区依据小区间的相似性聚集到指定的 k 个类簇中，使得每个簇内的小区非常相似且不同簇内的小区非常不同。每个小区属于且仅属于一个其到类簇中心距离最小的类簇。

对于 Kmeans，首先需要初始化 k 个聚类中心（C1,C2,……CK），然后通过计算每一个对象到聚类中心的欧式距离：

$$\mathrm{dis}\left(X_i, C_j\right) = \sqrt{\sum_{t=1}^{m}\left(X_{it} - C_{jt}\right)^2}$$

上式中，Xi 表示第 i 个对象，1≤i≤n；Cj 表示第 j 个聚类中心，1≤j≤k ，Xit 表示第 i 个对象的第 t 个属性，1≤t≤m ，Cjt 表示第 j 个聚类中心的第 t 个属性。

依次比较每一个对象到每一个聚类中心的距离，将对象分配到距离最近的聚类中心的类簇中，得到 k 个类簇（S1,S2,……SK）。 KMeans 算法用中心定义了类簇的原型，类簇中心就是类簇内所有对象在各个维度的均值，其计算公式如下

$$C_l = \frac{\sum_{X_i \in S_l} X_i}{|S_l|}$$

上式中，Cl 表示第 l 个聚类的中心，1⩽l⩽k，|Sl|表示第 l 个类簇中对象的个数，Xi 表示第 l 个类簇中第 i 个对象，1⩽i⩽|Sl|。

2）模型结果

通过分析数据并结合经验，我们将小区划分为三类。利用 Kmeans 算法分别针对三个核心指标对小区进行聚类的结果如下。

可以发现三个核心指标的三类小区在峰值高度、峰的宽度、高峰时段、高峰

持续时长等方面均存在明显差异。

对于平均用户数，大多数小区的高峰时段相同，但其中第三类小区比其他两类小区在高峰值处持续时间更长、低谷期下降幅度更小。

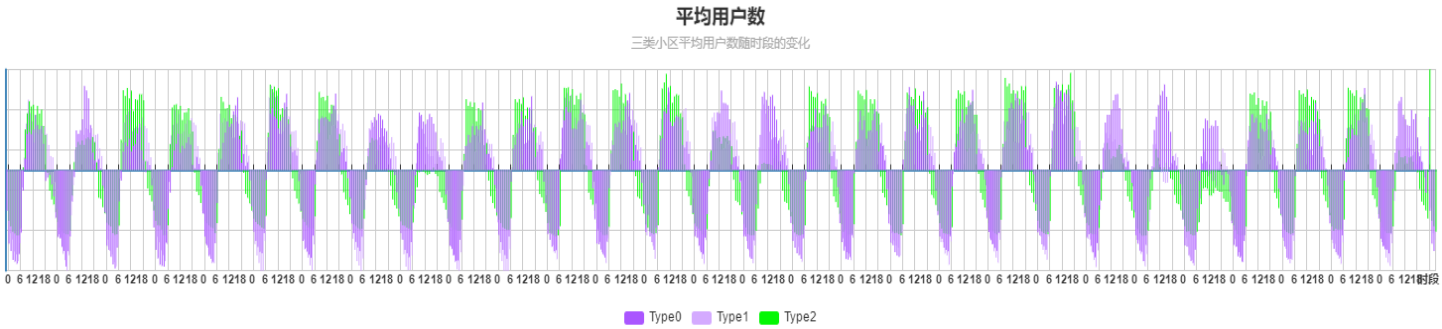


图 3 平均用户数聚类结果

在 PDCP 流量指标上，不同小区的高低峰时刻存在差异，且第二类小区高峰峰值高、高峰峰值持续时间短，低峰下降幅度小。

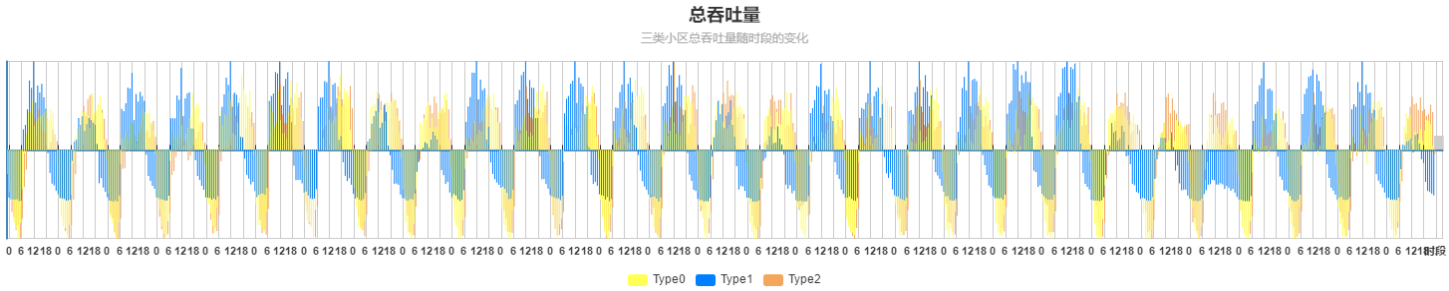


图 4 小区 PDPC 流量聚类结果

平均激活用户数在不同小区的表现中，其中第三类小区的高峰期开始时刻比其他两类小区更早，低谷期下降幅度也较低。

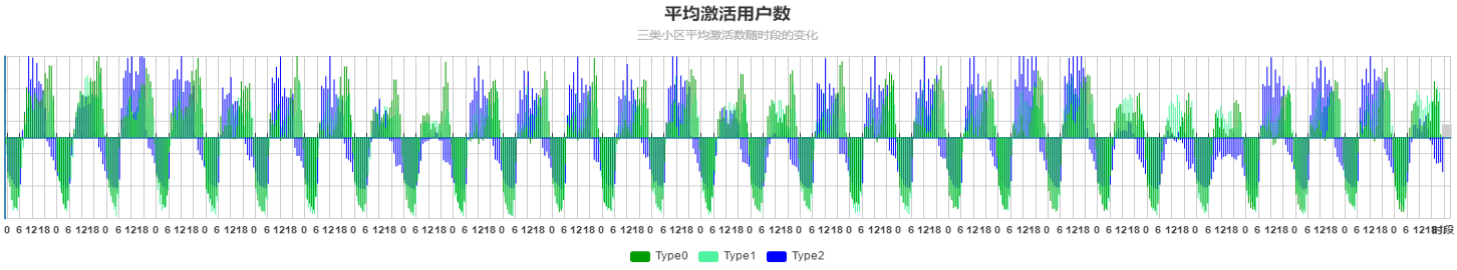


图 5 平均激活用户数聚类结果

5.3.3 DBSCAN 聚类

基于上述 Kmeans 算法对小区的分类，对每个指标中的每一类采用 DBSCAN 算法，筛选出异常数据。

1）理论基础

DBSCAN 的原理是识别特征空间中的“拥挤”区域中的点，在这些区域中许多数据点靠近在一起，这些区域被称为特征空间中的密集区域。在密集区域内的点被称为核心样本（或核心点）。DBSCAN 有两个参数：eps 和 min_samples，如果

在距离一个给定数据点 eps 的距离内至少有 min_samples 个数据点，那么这个数据点就是核心样本。DBSCAN 将彼此距离小于 eps 的核心样本放到同一个簇中。

算法首先任意选取一个点，然后找到到这个点的距离小于等于 eps 的所有点。如果距离起始点的距离在 eps 距离之内的数据点小于 min_samples，那么这个点就会被标记为噪声。如果这个点是核心样本，则为其分配一个新的簇标签，然后访问距离该点 eps 距离内的所有邻居，如果邻居没有被分配簇标签，则将该核心样本的簇标签分配给该邻居，如果邻居也是核心样本点，则依次访问其邻居，以此类推。随着簇逐渐增大，直到在簇 eps 距离内没有更多核心样本为止。

2）模型结果

将 DBSCAN 聚类结果可视化为如下图：

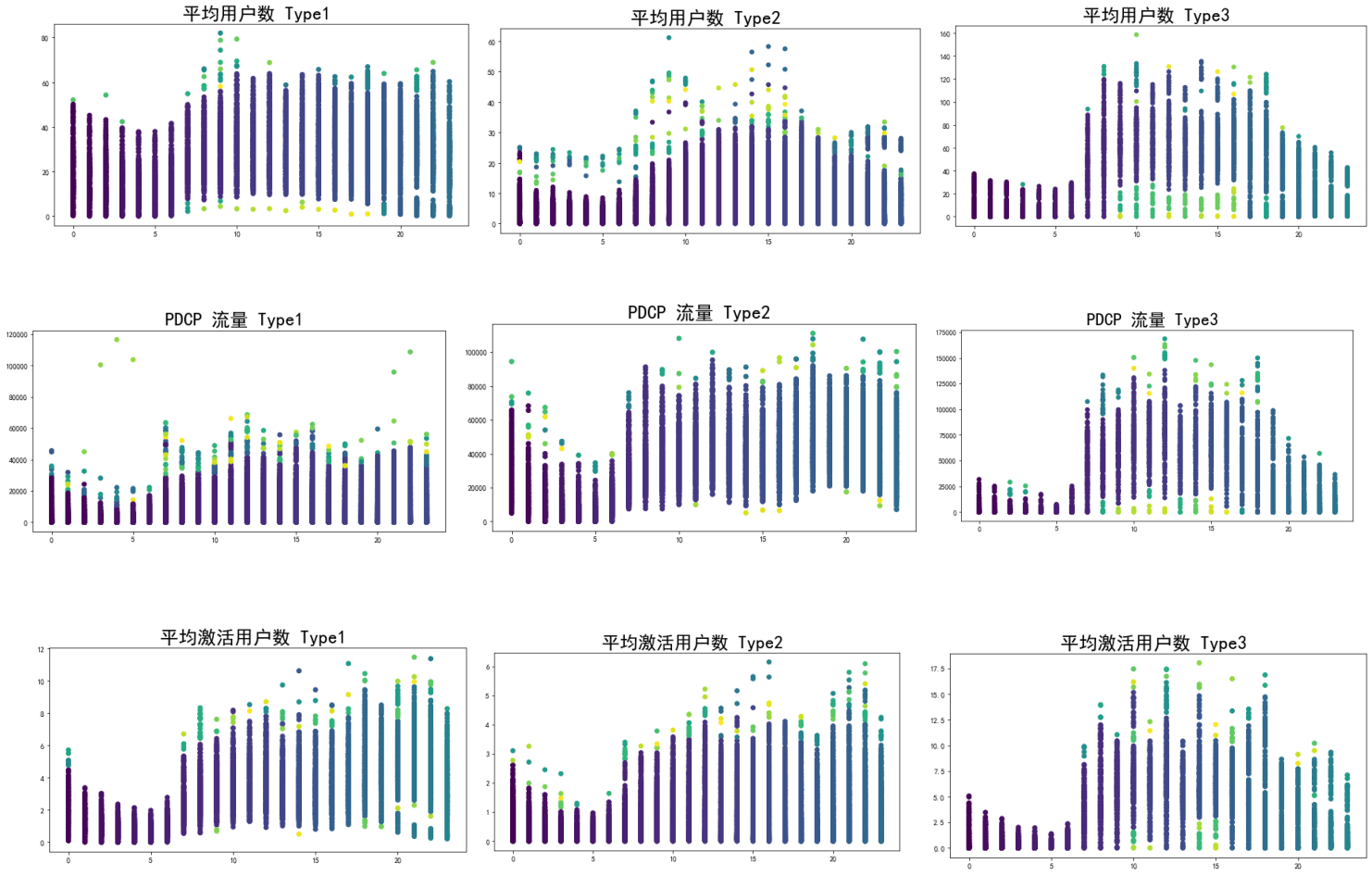


图 6 DBSCAN 聚类结果

根据 DBSCAN 聚类模型的结果，我们得出以下结论。

表 2 异常检测汇总表

	时间周期选择标准	异常孤立点的个数	异常周期个数
小区内的平均用户数	一段时间内任意相邻两次异常发生间隔≤1 小时	202	43
小区 PDCP 流量	一段时间内任意相邻两次异常发生间隔≤1 小时	218	36
平均激活用户数	一段时间内任意相邻两	239	33

	次异常发生间隔≤1 小时		

*注：上表中的异常周期假设为：若一段时间内，异常值发生次数大于等于 2，且任意相邻两次异常发生间隔都小于等于 1 小时，则该段时间视为一个异常周期。

六、问题二模型的建立与求解

6.1 建模思路

问题二要求利用异常数值前的数据建立预测模型，预测未来是否会发生异常数值，并综合考虑模型的输入和输出时间跨度。经分析发现，问题二是对时间序列异常数值的预测，因此本组使用 LSTM 模型求解。

首先利用根据三个核心指标将小区分为三类，并剔除问题一中预测的异常值点。我们假设在相同指标下，每类小区的分布大致相同，因此将同类小区数据合并求得均值，得到每个指标下的三个数据集。随后进行数据预处理与特征工程得到九个数据集，针对三类指标分别输入模型。通过输入数据学习长期依赖信息，可以有效预测在未来一定时间内的三个指标的数值。最后通过设定阈值，即可筛选出异常值点。具体流程如下图所示：

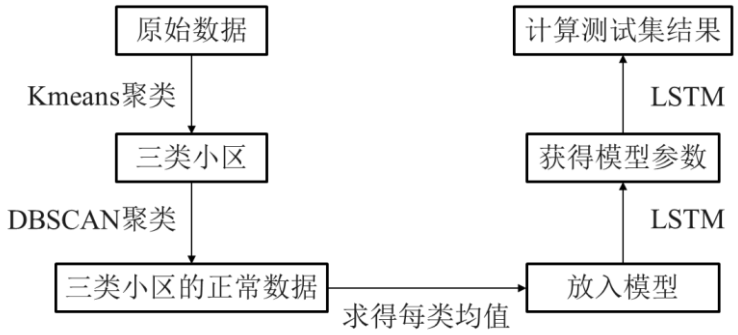


图 7 实验流程

6.2 数据预处理与特征工程

由于需要确保输入数据的准确性，我们针对三个指标分别剔除了问题一中预测得到的异常值点，得到的数据规模如下：

表 3 三个指标下用于训练模型的数据规模

	指标一	指标二	指标三
数据量	40166	40150	40129

针对问题一中的聚类结果，我们将 58 个小区在三个指标上分为如下九类：

表 4 三个指标下不同类别小区个数

	指标一	指标二	指标三
第一类小区	16	37	16
第二类小区	34	15	35
第三类小区	8	6	7

随后对每个指标下的三类小区内部求均值，作为每个指标下不同类小区的模型输入，得到 3×3=9 个数据集，每个数据集样品数量为总时间跨度，即 24×

29=696 条。

在选择数据特征时，我们对指标内容相近的特征计算求得相关系数，得到“eNodeB 内异频切换出成功次数”和“eNodeB 内异频切换出执行次数”两个指标的相关系数达到 0.99，具有完全多重共线性，因此剔除“eNodeB 内异频切换出执行次数”，并保留“eNodeB 内异频切换出成功次数”。

对于取值较少且离散的变量，我们认为数值本身对于预测结果影响较小，可以当作分类变量处理。因此我们选择使用目标编码，即利用目标变量和当前变量的特征共同编码，将离散值通过映射成为与目标变量存在数量关系的连续变量，作为最终的编码结果。通过初步观察，我们发现“上行可用的 PRB 个数”“下行可用的 PRB 个数”取值均仅有 0、100、200 三种取值，因此对其进行了目标编码处理。

此外，由于 LSTM 模型的长短期记忆机制是建立在输入数据顺序上的，数据发生的时间本身并不会对预测结果产生影响，因此剔除“时间”这一变量。同时，由于已经将目标变量分布相似小区归为同类，小区的“基站编号”“小区编号”“本地小区标识”对预测结果并无影响，同样进行剔除处理。最终我们利用 64 个变量对指标一、三进行预测，用 64 个变量对指标二进行预测。

为了消除量纲数据之间的量纲影响，将所有特征统一到大致相同的数据区间内，我们对特征进行了线性函数归一化处理，将其等比放缩到[0,1]。计算方式如下：

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

特别的，我们在实验中发现，目标变量由于小数位数限制且不归一化处理对 LSTM 训练结果无影响，因此不对其进行归一化，仍采用原始数值。由于“小区 PDCP 流量”的数量级高达 10^{11}，我们对其取 10^{-9} 后是使得数值大致分布在[0,100]。

6.3 模型建立

我们采用 LSTM 方法预测未来一定时间内三个指标的结果。

1）理论基础

LSTM（Long Short-Term Memory）模型是一种 RNN 的变型，最早由 Juergen Schmidhuber 提出的。经典的 LSTM 模型结构如下：

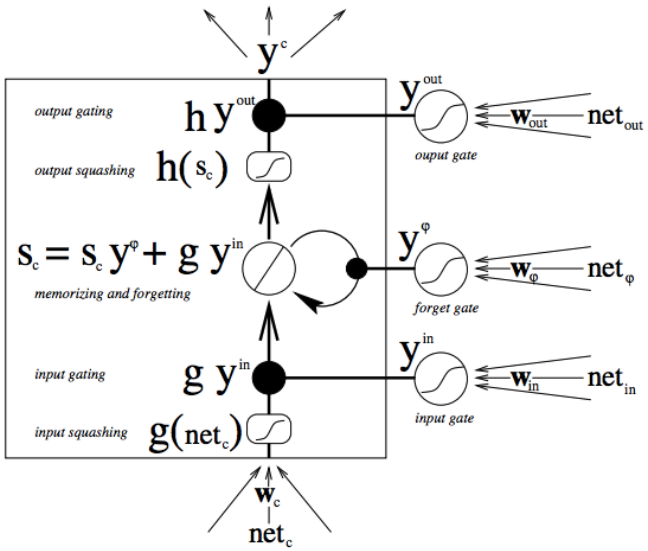


图 8 LSTM 模型结构

LSTM 的特点就是在 RNN 结构以外添加了各层的阀门节点。阀门有 3 类：遗忘阀门、输入阀门和输出阀门。这些阀门可以打开或关闭，用于将判断模型网络的记忆态在该层输出的结果是否达到阈值从而加入到当前该层的计算中。如图中所示，阀门节点利用 sigmoid 函数将网络的记忆态作为输入计算；如果输出结果达到阈值则将该阀门输出与当前层的的计算结果相乘作为下一层的输入；如果没有达到阈值则将该输出结果遗忘掉。每一层包括阀门节点的权重都会在每一次模型反向传播训练过程中更新。更具体的 LSTM 的判断计算过程如下图所示：

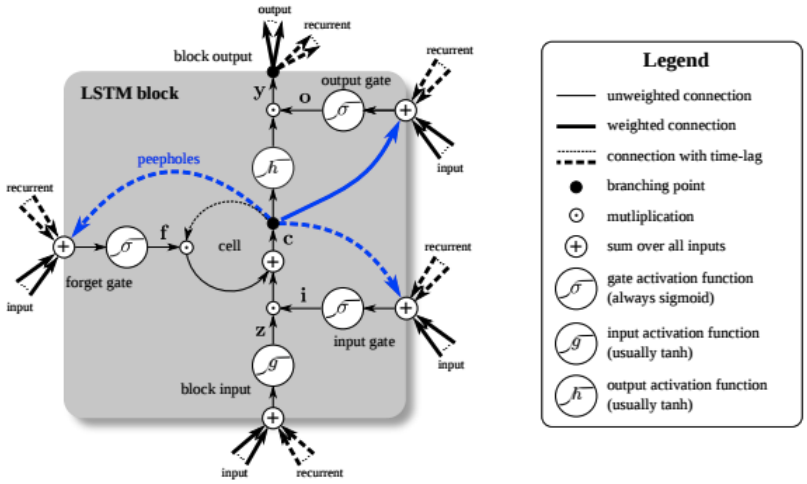


图 9 LSTM 的判断计算过程

LSTM 模型的记忆功能就是由这些阀门节点实现的。当阀门打开的时候，前面模型的训练结果就会关联到当前的模型计算，而当阀门关闭的时候之前的计算结果就不再影响当前的计算。因此，通过调节阀门的开关我们就可以实现早期序列对最终结果的影响。[2]

2）模型求解

在使用 LSTM 预测三类指标数值时，我们选取均方根误差（RMSE）作为测评结

果。计算方式如下：

$$RMSE = \sqrt{\frac{\sum(X_{actual} - X_{predict})^2}{n}}$$

为了提高模型鲁棒性，我们将数据集划分为训练集和验证集。同时，为了更好解决问题三的问题，我们统一设置输出数据时间跨度（stepout）为 72 用于模型训练。

为了得到较优的训练超参数，我们不断调整神经网络层数(layers)、dropout 值、以及输入数据时间跨度（stepin），得到九个数据集上三个指标的最低 RMSE 结果如下：

表 6 验证集中 RMSE 数值

	小区内的平均用户数	小区 PDCP 流量	平均激活用户数
第一类小区	0.51	1.00	0.29
第二类小区	6.13	5.23	0.08
第三类小区	11.13	11.95	0.98

由表 6 可以看出，不同类别小区的训练结果之间存在显著差异。其中，由于第三类小区的数值普遍较大，RMSE 也高于其他类小区。同时，第一类小区的三个指标预测结果较好，可以说明该类别小区的数值分布相似度较高。第二类小区则在平均激活用户数指标上表现较好，在其余两个指标上存在一定提升空间。

表 6 的 RMSE 数值对应的模型参数如下：

表 7 验证集中模型层数和输入数据时间跨度

	小区内的平均用户数	小区 PDCP 流量	平均激活用户数
第一类小区	layers=250 stepin=16	layers=250 stepin=36	layers=100 stepin=24
第二类小区	layers=100 stepin=16	layers=250 stepin=16	layers=50 stepin=36
第三类小区	layers=100 stepin=16	layers=100 stepin=36	layers=250 stepin=36

由表 7 数据可以看出，性能较好的模型神经网络层数普遍较多、步长较长，说明模型可以从有限的输入数据中学习更多信息。同时，不同类别小区间、三个指标间差异较大。

为了找到异常值点，我们设置了阈值作为划分标准。阈值设置方法为：

（1）计算每条数据的 Y_actual 与 Y_predict 的距离；

（2）按照从小到大的顺序排列，找到位于 95%位置的距离作为 threshold；

（3）若 Y_actual 与 Y_predict 的距离大于 threshold，则将其判为异常数据。

以第三类小区的平均激活用户数为例，拟合结果如下图所示：

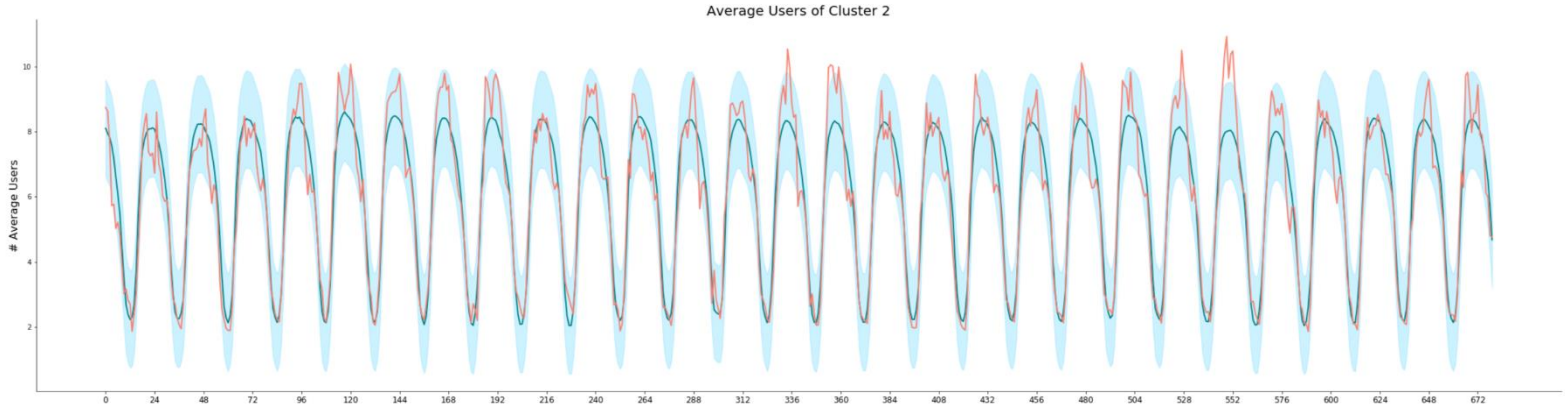


图 10 第三类小区的平均激活用户数拟合结果

如上图所示，深蓝色曲线为预测值，橙色曲线为实际值，浅蓝色背景为正常值范围。可以明显看出，部分点超过正常值范围，因此将其判断为异常数值。

按照上述方法处理后，我们找到的异常数据统计如下：

表 8 问题二模型针对所有数据预测结果

	小区内的平均用户数	小区 PDCP 流量	平均激活用户数
异常值数量	1914	1871	1861

七、问题三模型的建立与求解

6.1 建模思路

问题三沿用问题二中建模思路，利用问题二中训练得到的模型对接下来 72 个时刻的三个指标进行预测。在预测时，我们仍用问题一中的聚类结果将 58 个小区划分为三类，假设每个类别中小区的分布大致相同，得到九类预测结果。

6.2 模型建立

我们选择 LSTM 模型建模，将输出数据时间跨度 stepout 设置为 72。其余指标与表 7 相同，预测结果详见附件 2。三种指标的预测值分布如下图所示：

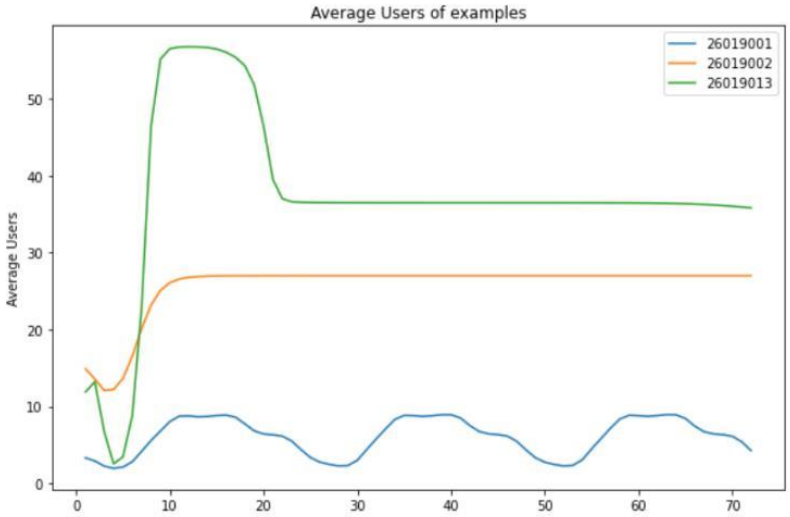


图 11 问题三中小区内的平均用户数的预测值分布

图 11 显示了三种类别的小区内的平均用户数在 72 个时刻的预测值。26019002、26019001、26019013 分别代表第一、第二、第三类小区的平均激活用户数量的预测情况。可以看出，第一类小区（橙色曲线）在 26 日使用人数激增，达到峰值后用户数量不变，持续 27、28 日两天；第二类小区（绿色曲线）在 9 月 26 日较多，而 27、28 日平均用户数量减小，且保持稳定；第三类小区（蓝色曲线）在 26-28 日每天平均用户数量呈现周期性分布。通过查找日历，我们发现 9 月 26 日为周五、26、27 日为周末：

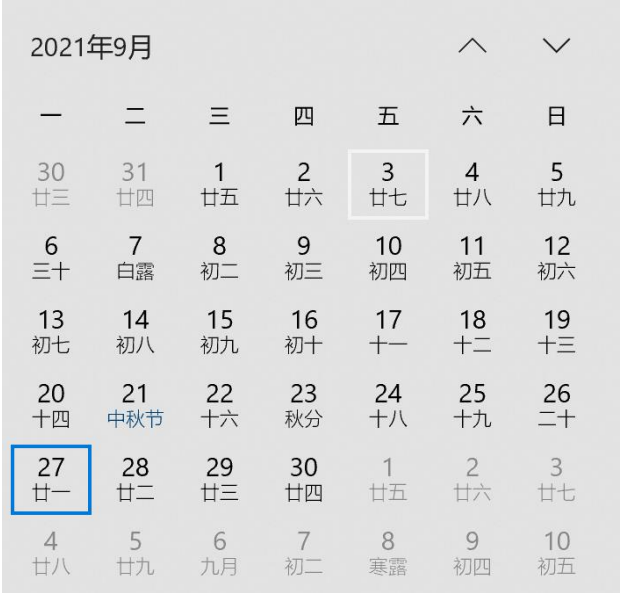


图 12 测试集时间分布

结合小区内平均用户数量，我们做出合理推测：

（1）第一类小区

用户主要为工作日上班人群，上网时间与时间分布相关性较强。在周五晚上休息后会以上网作为休息方式，上网人数增加；周末开始后上网时间和时段呈现均匀分布。

（2）第二类小区

用户主要为儿童或周末上班人群。在周五晚上休息后会以上网作为休息方式，上网人数增加；但由于周末仍要外出，因此平均用户数量减小。

（3）第三类小区

用户主要为老年人。由于其作息周期性较强，受时间变化影响较小，是稳定的时间序列。因此，平均用户数量随时间呈现周期性变化。

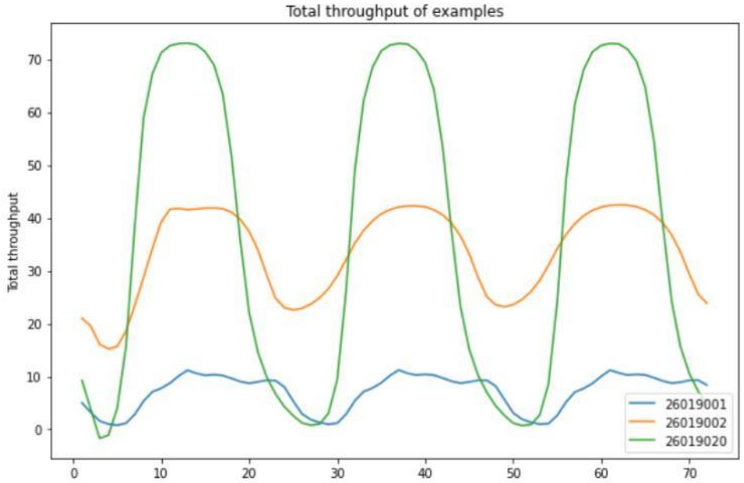


图 13 问题三中小区 PDCP 流量的预测值分布

图 13 显示了三种类别的小区 PDCP 流量在 72 个时刻的预测值。可以看出，三类小区虽然数量存在较大差异，但周期性大致相同。

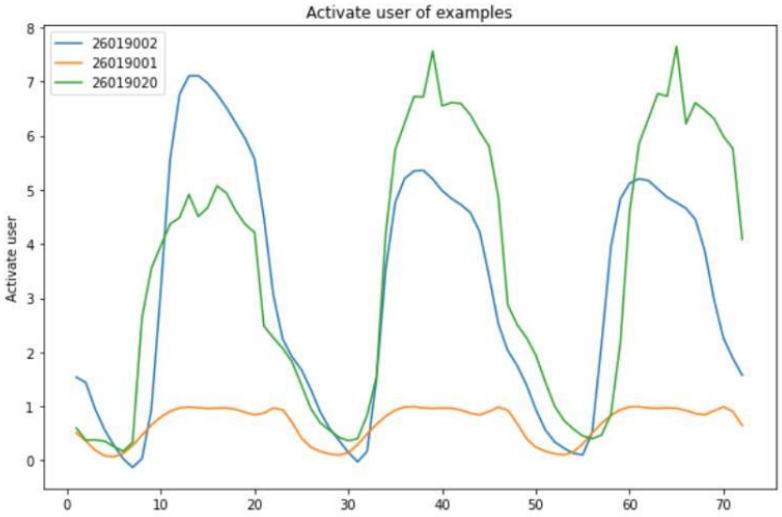


图 14 问题三中平均激活用户数量的预测值分布

图 14 显示了三种类别的小区中平均激活用户数量在 72 个时刻的预测值。26019001、26019002、26019020 分别代表第一、第二、第三类小区的平均激活用户数量的预测情况。可以看出，第一类小区的平均激活用户数量较小、周期性较长；第二类小区的平均激活用户数量较大、波峰与波谷差异较大；第三类小区在 9 月 26 日的平均激活用户数量相比后两天明显偏小，与问题一中出现的趋势大致相同。

九、模型评价

9.1 模型优点

（1）问题一用 Kmeans 聚类方法将小区分为三类，凸显小区之间特征分布差异，针对不同类别小区建模，结果可信度较高。
（2）问题二使用对使用目标编码，将离散特征根据目标变量数值连续化，信息利用率增加。
（3）问题二、三使用 LSTM 模型，损失较小，预测结果可信度较高。

9.2 模型缺点

（1）由于 58 个小区随时间分布不尽相同，更为准确的方法应当是针对 58 个小区或更多分类分别建模分析。
（2）问题二中数据量较少，使用神经网络模型存在过拟合的风险。
（3）问题二、三模型中仍存在其他影响小区内的平均用户数、小区 DCPC 流量、平均激活用户数的因素，应结合相关信息全面考虑。

9.3 模型改进方向

（1）由于 58 个小区随时间分布不尽相同，可以针对 58 个小区或更多分类分别建模分析。
（2）针对潜在过拟合问题，可以使用增大数据量、减少参数数量等方法解决。
（3）对于其他影响因素的考量，可以参考其他数据或信息全面考虑。

参考文献

[1] [德]Andreas C.Muller [美]Sarah Guido,《Python 机器学习基础教程》,人民邮电出版社 2018 年

[2] Arkenstone, Python 中利用 LSTM 模型进行时间序列预测分析，https://www.cnblogs.com/arkenstone/p/5794063.html , 2020 年 11 月 1 日

附　录

代码名称	代码解决的问题	代码电子版所在的位置
问题一：小区 Kmeans 聚类	问题一：小区 Kmeans 聚类	A21081 附件 9
问题一：DBSCAN 聚类代码	问题一：DBSCAN 聚类代码	A21081 附件 9
问题二：删除异常数据	问题二：删除异常数据	A21081 附件 9
问题二：数据预处理	问题二：数据预处理	A21081 附件 9
问题二：LSTM-指标一	问题二：LSTM-指标一	A21081 附件 9
问题二：LSTM-指标二	问题二：LSTM-指标二	A21081 附件 9
问题二：LSTM-指标三	问题二：LSTM-指标三	A21081 附件 9
问题一：异常数据汇总	问题一：异常数据汇总	A21081 附件 9
问题二：数据预处理数据	问题二：数据预处理数据	A21081 附件 9

问题一：小区 Kmeans 聚类

```
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import  StandardScaler
from sklearn import preprocessing
from collections import Counter
from sklearn.cluster import KMeans
import mglearn
import matplotlib.pyplot as plt

df = pd.read_csv('dataAstandard.csv')
df.drop('时间', axis=1, inplace=True)
df.head(3)

lst_community=[]
for i in range(58):
    lst_community.append(df[df["小区编号"]==26019001+i])

lst_feature1=[]
for community in lst_community:
    lst_feature1.append(community["小区内的平均用户数
"].reset_index().T)

df_q1=lst_feature1[0]
for community in range(len(lst_feature1)-1):
    df_q1=df_q1.append(lst_feature1[community+1])
df_q1.drop('index', axis=0, inplace=True)
```

```
27.
28. lst_q1_index=[]
29. for i in range(58):
30.     lst_q1_index.append("第{}小区".format(i+1))
31. df_q1.index=lst_q1_index
32.
33. lst_q1_column=[]
34. for i in range(29):
35.     for j in range(24):
36.         lst_q1_column.append("第{}天{}点平均用户数".format(i+1,j))
37. df_q1.columns=lst_q1_column
38.
39. df_q1
40.
41. lst_feature2=[]
42. for community in lst_community:
43.     lst_feature2.append(community["总吞吐量"].reset_index().T)
44.
45. df_q2=lst_feature2[0]
46. for community in range(len(lst_feature2)-1):
47.     df_q2=df_q2.append(lst_feature2[community+1])
48.
49. df_q2.drop('index', axis=0, inplace=True)
50.
51. lst_q2_index=[]
52. for i in range(58):
53.     lst_q2_index.append("第{}小区".format(i+1))
54. df_q2.index=lst_q2_index
55.
56. lst_q2_column=[]
57. for i in range(29):
58.     for j in range(24):
59.         lst_q2_column.append("第{}天{}点总吞吐量".format(i+1,j))
60. df_q2.columns=lst_q2_column
61.
62. df_q2
63.
64. lst_feature3=[]
65. for community in lst_community:
66.     lst_feature3.append(community["平均激活用户数
    "].reset_index().T)#pd Series 变 dataframe
67.
68. df_q3=lst_feature3[0]
69. for community in range(len(lst_feature2)-1):
```

```
    df_q3=df_q3.append(lst_feature3[community+1])

df_q3.drop('index', axis=0, inplace=True)

lst_q3_index=[]
for i in range(58):
    lst_q3_index.append("第{}小区".format(i+1))
df_q3.index=lst_q3_index

lst_q3_column=[]
for i in range(29):
    for j in range(24):
        lst_q3_column.append("第{}天{}点平均激活用户数
".format(i+1,j))
df_q3.columns=lst_q3_column

df_q3

df_q=pd.concat([df_q1,df_q2,df_q3],axis=1)
df_q

kmeans=KMeans(n_clusters=3)

kmeans.fit(df_q)
assignments=kmeans.labels_
print("三种指标下聚类的标签")
print(assignments)
print(Counter(assignments),"\n")

kmeans.fit(df_q1)
assignments1=kmeans.labels_
  print("平均用户数聚类的标签")
  print(assignments1)
  print(Counter(assignments1),"\n")

  kmeans.fit(df_q2)
  assignments2=kmeans.labels_
  print("总吞吐量聚类的标签")
  print(assignments2)
  print(Counter(assignments2),"\n")

  kmeans.fit(df_q3)
  assignments3=kmeans.labels_
  print("平均激活用户数聚类的标签")
```

```
print(assignments3)
print(Counter(assignments3))

df_q1['type'] = assignments1
df_q2['type'] = assignments2
df_q3['type'] = assignments3

mean1=df_q1.groupby(['type']).mean()
mean2=df_q2.groupby(['type']).mean()
mean3=df_q3.groupby(['type']).mean()

scaler=StandardScaler()

scaler.fit(mean1.T)
df_q1=pd.DataFrame(scaler.transform(mean1.T)) # numpy.ndarray 转 dataframe
scaler.fit(mean2.T)
df_q2=pd.DataFrame(scaler.transform(mean2.T))
scaler.fit(mean3.T)
df_q3=pd.DataFrame(scaler.transform(mean3.T))

outputpath = 'E:/Fanny_Python_Files/df_q1.csv'
(df_q1).to_csv(outputpath,sep=',',index=False,header=True)

outputpath = 'E:/Fanny_Python_Files/df_q2.csv'
(df_q2).to_csv(outputpath,sep=',',index=False,header=True)

outputpath = 'E:/Fanny_Python_Files/df_q3.csv'
(df_q3).to_csv(outputpath,sep=',',index=False,header=True)
```

问题一：DBSCAN 聚类

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import warnings; warnings.simplefilter('ignore')
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import  StandardScaler
from sklearn import preprocessing
from collections import Counter
from sklearn.cluster import KMeans
import mglearn
from sklearn import metrics
```

```
14.
15. df = pd.read_csv('dataAstandard.csv')
16.
17. #平均激活用户数
18. cluster_3 = [1 ,0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
    2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,\
19.  1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 1, 2, 2, 0, 1, 1]
20. c1, c2, c3 = [], [], []
21. for i in range(len(cluster_3)):
22.     if cluster_3[i]==0:
23.         c1.append(int("260190"+str(i+1)))
24.     if cluster_3[i]==1:
25.         c2.append(int("260190"+str(i+1)))
26.     if cluster_3[i]==2:
27.         c3.append(int("260190"+str(i+1)))
28.
29. cluster_3_1 = df.loc[df["小区编号"].isin(c1),:]
30. cluster_3_2 = df.loc[df["小区编号"].isin(c2),:]
31. cluster_3_3 = df.loc[df["小区编号"].isin(c3),:]
32.
33. feature3_class1=cluster_3_1[["时段","平均激活用户数"]]
34. feature3_class2=cluster_3_2[["时段","平均激活用户数"]]
35. feature3_class3=cluster_3_3[["时段","平均激活用户数"]]
36.
37. scaler=StandardScaler()
38.
39. scaler.fit(feature3_class1)
40. feature3_class1_sclaed=scaler.transform(feature3_class1)
41.
42. scaler.fit(feature3_class2)
43. feature3_class2_sclaed=scaler.transform(feature3_class2)
44.
45. scaler.fit(feature3_class3)
46. feature3_class3_sclaed=scaler.transform(feature3_class3)
47.
48. # 平均激活用户数 class1
49. plt.rcParams['font.sans-serif'] = ['SimHei']
50. plt.rcParams['axes.unicode_minus'] = False
51.
52. dbscan1=DBSCAN(eps=0.13,min_samples=1)
53. result1=dbscan1.fit_predict(feature3_class1_sclaed)
54. plt.figure(figsize=(10,5))
55. cmap2=plt.get_cmap('Set2')
```

```
plt.scatter(feature3_class1.iloc[:,0],feature3_class1.iloc[:,1],c=result1)
plt.title("平均激活用户数 Type1 ",fontsize=25)
plt.show()

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)

feature3_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<10:
        feature3_class1_lst.append(k)

feature3_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature3_class1_lst:
        feature3_class1_location.append(i)

len(feature3_class1_location)/len(result1)

# 平均激活用户数 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan2.fit_predict(feature3_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature3_class2.iloc[:,0],feature3_class2.iloc[:,1],c=result2)
plt.title("平均激活用户数 Type2 ",fontsize=25)
plt.show()

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan2.fit_predict(feature3_class2_sclaed)

feature3_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<5:
        feature3_class2_lst.append(k)

feature3_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature3_class2_lst:
```

```
        feature3_class2_location.append(i)

len(feature3_class2_location)/len(result2)

# 平均激活用户数 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature3_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature3_class3.iloc[:,0],feature3_class3.iloc[:,1],c=result3)
plt.title("平均激活用户数 Type3 ",fontsize=25)
plt.show()

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature3_class3_sclaed)

feature3_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature3_class3_lst.append(k)

feature3_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature3_class3_lst:
        feature3_class3_location.append(i)

len(feature3_class3_location)/len(result3)

#平均用户数
cluster_1 = [1 ,0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1, 1, 2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,\
  1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 2, 0, 2, 0, 1, 1]
c1, c2, c3 = [], [], []
for i in range(len(cluster_1)):
    if cluster_1[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_1[i]==1:
        c2.append(int("260190"+str(i+1)))
    if cluster_1[i]==2:
        c3.append(int("260190"+str(i+1)))
```

```
140.
141.   cluster_1_1 = df.loc[df["小区编号"].isin(c1),:]
142.   cluster_1_2 = df.loc[df["小区编号"].isin(c2),:]
143.   cluster_1_3 = df.loc[df["小区编号"].isin(c3),:]
144.
145.   feature1_class1=cluster_1_1[["时段","小区内的平均用户数"]]
146.   feature1_class2=cluster_1_2[["时段","小区内的平均用户数"]]
147.   feature1_class3=cluster_1_3[["时段","小区内的平均用户数"]]
148.
149.   scaler=StandardScaler()
150.   scaler.fit(feature1_class1)
151.   feature1_class1_sclaed=scaler.transform(feature1_class1)
152.
153.   scaler.fit(feature1_class2)
154.   feature1_class2_sclaed=scaler.transform(feature1_class2)
155.
156.   scaler.fit(feature1_class3)
157.   feature1_class3_sclaed=scaler.transform(feature1_class3)
158.
159.   # 平均用户数 class1
160.   plt.rcParams['font.sans-serif'] = ['SimHei']
161.   plt.rcParams['axes.unicode_minus'] = False
162.
163.   dbscan1=DBSCAN(eps=0.14,min_samples=1)
164.   result1=dbscan1.fit_predict(feature1_class1_sclaed)
165.   print(Counter(result1))
166.   plt.figure(figsize=(10,5))
167.   plt.scatter(feature1_class1.iloc[:,0],feature1_class1.iloc[:,1],c=
    result1)
168.   plt.title("平均用户数 Type1 ",fontsize=25)
169.   plt.show()
170.
171.   dbscan1=DBSCAN(eps=0.14,min_samples=1)
172.   result1=dbscan1.fit_predict(feature1_class1_sclaed)
173.
174.   feature1_class1_lst=[]
175.   for k in Counter(result1).keys():
176.       if Counter(result1)[k]<5:
177.           feature1_class1_lst.append(k)
178.
179.   feature1_class1_location=[]
180.   for i in range(len(result1)):
181.       if result1[i] in feature1_class1_lst:
182.           feature1_class1_location.append(i)
```

```
183.
184.   len(feature1_class1_location)/len(result1)
185.
186.   # 平均用户数 class2
187.   plt.rcParams['font.sans-serif'] = ['SimHei']
188.   plt.rcParams['axes.unicode_minus'] = False
189.
190.   dbscan2=DBSCAN(eps=0.12,min_samples=1)
191.   result2=dbscan2.fit_predict(feature1_class2_sclaed)
192.   print(Counter(result2))
193.   plt.figure(figsize=(10,5))
194.   plt.scatter(feature1_class2.iloc[:,0],feature1_class2.iloc[:,1],c=
    result2,s=30)
195.   plt.title("平均用户数 Type2 ",fontsize=25)
196.   plt.show()
197.
198.   dbscan2=DBSCAN(eps=0.12,min_samples=1)
199.   result2=dbscan2.fit_predict(feature1_class2_sclaed)
200.
201.   feature1_class2_lst=[]
202.   for k in Counter(result2).keys():
203.       if Counter(result2)[k]<3:
204.           feature1_class2_lst.append(k)
205.
206.   feature1_class2_location=[]
207.   for i in range(len(result2)):
208.       if result2[i] in feature1_class2_lst:
209.           feature1_class2_location.append(i)
210.
211.   len(feature1_class2_location)/len(result2)
212.
213.   # 平均用户数 class3
214.   plt.rcParams['font.sans-serif'] = ['SimHei']
215.   plt.rcParams['axes.unicode_minus'] = False
216.
217.   dbscan3=DBSCAN(eps=0.13,min_samples=1)
218.   result3=dbscan3.fit_predict(feature1_class3_sclaed)
219.   print(Counter(result3))
220.   plt.figure(figsize=(10,5))
221.   plt.scatter(feature1_class3.iloc[:,0],feature1_class3.iloc[:,1],c=
    result3,s=30)
222.   plt.title("平均用户数 Type3 ",fontsize=25)
223.   plt.show()
224.
```

```
#多少作为断层?
dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)

feature1_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature1_class3_lst.append(k)

feature1_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature1_class3_lst:
        feature1_class3_location.append(i)

len(feature1_class3_location)/len(result3)

#总吞吐量
cluster_2 = [0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0,
0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0,\
            0, 0, 0, 1, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0]
c1, c2, c3 = [], [], []
for i in range(len(cluster_2)):
    if cluster_2[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_2[i]==1:
        c2.append(int("260190"+str(i+1)))
    if cluster_2[i]==2:
        c3.append(int("260190"+str(i+1)))

cluster_2_1 = df.loc[df["小区编号"].isin(c1),:]
cluster_2_2 = df.loc[df["小区编号"].isin(c2),:]
cluster_2_3 = df.loc[df["小区编号"].isin(c3),:]

feature2_class1=cluster_2_1[["时段","总吞吐量"]]
feature2_class2=cluster_2_2[["时段","总吞吐量"]]
feature2_class3=cluster_2_3[["时段","总吞吐量"]]

#总吞吐量
scaler=StandardScaler()

scaler.fit(feature2_class1)
feature2_class1_sclaed=scaler.transform(feature2_class1)

```

```
scaler.fit(feature2_class2)
feature2_class2_sclaed=scaler.transform(feature2_class2)

scaler.fit(feature2_class3)
feature2_class3_sclaed=scaler.transform(feature2_class3)

# 总吞吐量 class1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature2_class1_sclaed)
print(Counter(result1))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class1.iloc[:,0],feature2_class1.iloc[:,1],c=
result1)
plt.title("PDCP 流量 Type1 ",fontsize=25)
plt.show()


dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature2_class1_sclaed)

feature2_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<5:
        feature2_class1_lst.append(k)

feature2_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature2_class1_lst:
        feature2_class1_location.append(i)

len(feature2_class1_location)/len(result1)

#总吞吐量 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

esp_lst=[0.03,0.04,0.05]
min_samples_lst=[0,1]

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan1.fit_predict(feature2_class2_sclaed)
```

```
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class2.iloc[:,0],feature2_class2.iloc[:,1],c=result2)
plt.title("PDCP 流量 Type2 ",fontsize=25)
plt.show()


dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan1.fit_predict(feature2_class2_sclaed)

feature2_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<2:
        feature2_class2_lst.append(k)

feature2_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature2_class2_lst:
        feature2_class2_location.append(i)

len(feature2_class2_location)/len(result2)

# 总吞吐量 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class3.iloc[:,0],feature2_class3.iloc[:,1],c=result3)
plt.title("PDCP 流量 Type3 ",fontsize=25)
plt.show()

#断层选多少？
dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)

feature2_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<3:
        feature2_class3_lst.append(k)
```

```
352.
353.   feature2_class3_location=[]
354.   for i in range(len(result3)):
355.       if result3[i] in feature2_class3_lst:
356.           feature2_class3_location.append(i)
357.   #断层选多少？
358.   dbscan3=DBSCAN(eps=0.14,min_samples=1)
359.   result3=dbscan3.fit_predict(feature2_class3_sclaed)
360.
361.   feature2_class3_lst=[]
362.   for k in Counter(result3).keys():
363.       if Counter(result3)[k]<3:
364.           feature2_class3_lst.append(k)
365.
366.   feature2_class3_location=[]
367.   for i in range(len(result3)):
368.       if result3[i] in feature2_class3_lst:
369.           feature2_class3_location.append(i)
370.
371.   len(feature2_class3_location)/len(result3)
372.
373.   len(feature2_class3_location)/len(result3)
```

问题二：删除异常数据

```
1.  import pandas as pd
2.  import numpy as np
3.  import matplotlib.pyplot as plt
4.  import matplotlib as mpl
5.  import seaborn as sns
6.  import warnings; warnings.simplefilter('ignore')
7.  from sklearn.cluster import DBSCAN
8.  from sklearn.preprocessing import  StandardScaler
9.  from sklearn import preprocessing
10. from collections import Counter
11. from sklearn.cluster import KMeans
12. import mglearn
13. from sklearn import metrics
14.
15. df = pd.read_csv('dataAstandard.csv')
16.
17. #平均激活用户数
18. cluster_3 = [1 ,0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
     2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,\
19.  1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 1, 2, 2, 0, 1, 1]
```

```
c1, c2, c3 = [], [], []
for i in range(len(cluster_3)):
    if cluster_3[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_3[i]==1:
        c2.append(int("260190"+str(i+1)))
    if cluster_3[i]==2:
        c3.append(int("260190"+str(i+1)))

cluster_3_1 = df.loc[df["小区编号"].isin(c1),:]
cluster_3_2 = df.loc[df["小区编号"].isin(c2),:]
cluster_3_3 = df.loc[df["小区编号"].isin(c3),:]

feature3_class1=cluster_3_1[["时段","平均激活用户数"]]
feature3_class2=cluster_3_2[["时段","平均激活用户数"]]
feature3_class3=cluster_3_3[["时段","平均激活用户数"]]

scaler=StandardScaler()

scaler.fit(feature3_class1)
feature3_class1_sclaed=scaler.transform(feature3_class1)

scaler.fit(feature3_class2)
feature3_class2_sclaed=scaler.transform(feature3_class2)

scaler.fit(feature3_class3)
feature3_class3_sclaed=scaler.transform(feature3_class3)

# 平均激活用户数 class1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)
plt.figure(figsize=(10,5))
cmap2=plt.get_cmap('Set2')
plt.scatter(feature3_class1.iloc[:,0],feature3_class1.iloc[:,1],c=res
ult1)
plt.title("平均激活用户数 Type1 ",fontsize=25)
plt.show()

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)

```

```
feature3_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<10:
        feature3_class1_lst.append(k)

feature3_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature3_class1_lst:
        feature3_class1_location.append(i)

len(feature3_class1_location)/len(result1)

# 平均激活用户数 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan2.fit_predict(feature3_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature3_class2.iloc[:,0],feature3_class2.iloc[:,1],c=res
ult2)
plt.title("平均激活用户数 Type2 ",fontsize=25)
plt.show()

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan2.fit_predict(feature3_class2_sclaed)

feature3_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<5:
        feature3_class2_lst.append(k)

feature3_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature3_class2_lst:
        feature3_class2_location.append(i)

  len(feature3_class2_location)/len(result2)

  # 平均激活用户数 class3
  plt.rcParams['font.sans-serif'] = ['SimHei']
  plt.rcParams['axes.unicode_minus'] = False

```

```
106.   dbscan3=DBSCAN(eps=0.14,min_samples=1)
107.   result3=dbscan3.fit_predict(feature3_class3_sclaed)
108.   print(Counter(result3))
109.   plt.figure(figsize=(10,5))
110.   plt.scatter(feature3_class3.iloc[:,0],feature3_class3.iloc[:,1],c=
    result3)
111.   plt.title("平均激活用户数 Type3 ",fontsize=25)
112.   plt.show()
113.
114.   dbscan3=DBSCAN(eps=0.14,min_samples=1)
115.   result3=dbscan3.fit_predict(feature3_class3_sclaed)
116.
117.   feature3_class3_lst=[]
118.   for k in Counter(result3).keys():
119.       if Counter(result3)[k]<5:
120.           feature3_class3_lst.append(k)
121.
122.   feature3_class3_location=[]
123.   for i in range(len(result3)):
124.       if result3[i] in feature3_class3_lst:
125.           feature3_class3_location.append(i)
126.
127.   len(feature3_class3_location)/len(result3)
128.
129.   #平均用户数
130.   cluster_1 = [1 ,0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1,
     1, 2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,\
131.    1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 2, 0, 2, 0, 1, 1]
132.   c1, c2, c3 = [], [], []
133.   for i in range(len(cluster_1)):
134.       if cluster_1[i]==0:
135.           c1.append(int("260190"+str(i+1)))
136.       if cluster_1[i]==1:
137.           c2.append(int("260190"+str(i+1)))
138.       if cluster_1[i]==2:
139.           c3.append(int("260190"+str(i+1)))
140.
141.   cluster_1_1 = df.loc[df["小区编号"].isin(c1),:]
142.   cluster_1_2 = df.loc[df["小区编号"].isin(c2),:]
143.   cluster_1_3 = df.loc[df["小区编号"].isin(c3),:]
144.
145.   feature1_class1=cluster_1_1[["时段","小区内的平均用户数"]]
146.   feature1_class2=cluster_1_2[["时段","小区内的平均用户数"]]
147.   feature1_class3=cluster_1_3[["时段","小区内的平均用户数"]]
```

```
148.
149.   scaler=StandardScaler()
150.   scaler.fit(feature1_class1)
151.   feature1_class1_sclaed=scaler.transform(feature1_class1)
152.
153.   scaler.fit(feature1_class2)
154.   feature1_class2_sclaed=scaler.transform(feature1_class2)
155.
156.   scaler.fit(feature1_class3)
157.   feature1_class3_sclaed=scaler.transform(feature1_class3)
158.
159.   # 平均用户数 class1
160.   plt.rcParams['font.sans-serif'] = ['SimHei']
161.   plt.rcParams['axes.unicode_minus'] = False
162.
163.   dbscan1=DBSCAN(eps=0.14,min_samples=1)
164.   result1=dbscan1.fit_predict(feature1_class1_sclaed)
165.   print(Counter(result1))
166.   plt.figure(figsize=(10,5))
167.   plt.scatter(feature1_class1.iloc[:,0],feature1_class1.iloc[:,1],c=
    result1)
168.   plt.title("平均用户数 Type1 ",fontsize=25)
169.   plt.show()
170.
171.   dbscan1=DBSCAN(eps=0.14,min_samples=1)
172.   result1=dbscan1.fit_predict(feature1_class1_sclaed)
173.
174.   feature1_class1_lst=[]
175.   for k in Counter(result1).keys():
176.       if Counter(result1)[k]<5:
177.           feature1_class1_lst.append(k)
178.
179.   feature1_class1_location=[]
180.   for i in range(len(result1)):
181.       if result1[i] in feature1_class1_lst:
182.           feature1_class1_location.append(i)
183.
184.   len(feature1_class1_location)/len(result1)
185.
186.   # 平均用户数 class2
187.   plt.rcParams['font.sans-serif'] = ['SimHei']
188.   plt.rcParams['axes.unicode_minus'] = False
189.
190.   dbscan2=DBSCAN(eps=0.12,min_samples=1)
```

```
result2=dbscan2.fit_predict(feature1_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class2.iloc[:,0],feature1_class2.iloc[:,1],c=result2,s=30)
plt.title("平均用户数 Type2 ",fontsize=25)
plt.show()

dbscan2=DBSCAN(eps=0.12,min_samples=1)
result2=dbscan2.fit_predict(feature1_class2_sclaed)

feature1_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<3:
        feature1_class2_lst.append(k)

feature1_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature1_class2_lst:
        feature1_class2_location.append(i)

len(feature1_class2_location)/len(result2)

# 平均用户数 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class3.iloc[:,0],feature1_class3.iloc[:,1],c=result3,s=30)
plt.title("平均用户数 Type3 ",fontsize=25)
plt.show()

#多少作为断层?
dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)

feature1_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature1_class3_lst.append(k)
```

```
233.
234.  feature1_class3_location=[]
235.  for i in range(len(result3)):
236.      if result3[i] in feature1_class3_lst:
237.          feature1_class3_location.append(i)
238.
239.  len(feature1_class3_location)/len(result3)
240.
241.  #总吞吐量
242.  cluster_2 = [0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0,
     0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
     0,\
243.              0, 0, 0, 1, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0]
244.  c1, c2, c3 = [], [], []
245.  for i in range(len(cluster_2)):
246.      if cluster_2[i]==0:
247.          c1.append(int("260190"+str(i+1)))
248.      if cluster_2[i]==1:
249.          c2.append(int("260190"+str(i+1)))
250.      if cluster_2[i]==2:
251.          c3.append(int("260190"+str(i+1)))
252.
253.  cluster_2_1 = df.loc[df["小区编号"].isin(c1),:]
254.  cluster_2_2 = df.loc[df["小区编号"].isin(c2),:]
255.  cluster_2_3 = df.loc[df["小区编号"].isin(c3),:]
256.
257.  feature2_class1=cluster_2_1[["时段","总吞吐量"]]
258.  feature2_class2=cluster_2_2[["时段","总吞吐量"]]
259.  feature2_class3=cluster_2_3[["时段","总吞吐量"]]
260.
261.  #总吞吐量
262.  scaler=StandardScaler()
263.
264.  scaler.fit(feature2_class1)
265.  feature2_class1_sclaed=scaler.transform(feature2_class1)
266.
267.  scaler.fit(feature2_class2)
268.  feature2_class2_sclaed=scaler.transform(feature2_class2)
269.
270.  scaler.fit(feature2_class3)
271.  feature2_class3_sclaed=scaler.transform(feature2_class3)
272.
273.  # 总吞吐量 class1
274.  plt.rcParams['font.sans-serif'] = ['SimHei']
```

```
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature2_class1_sclaed)
print(Counter(result1))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class1.iloc[:,0],feature2_class1.iloc[:,1],c=result1)
plt.title("PDCP 流量 Type1 ",fontsize=25)
plt.show()


dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature2_class1_sclaed)

feature2_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<5:
        feature2_class1_lst.append(k)

feature2_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature2_class1_lst:
        feature2_class1_location.append(i)

len(feature2_class1_location)/len(result1)

#总吞吐量 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

esp_lst=[0.03,0.04,0.05]
min_samples_lst=[0,1]

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan1.fit_predict(feature2_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class2.iloc[:,0],feature2_class2.iloc[:,1],c=result2)
plt.title("PDCP 流量 Type2 ",fontsize=25)
plt.show()


```

```
dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan1.fit_predict(feature2_class2_sclaed)

feature2_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<2:
        feature2_class2_lst.append(k)

feature2_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature2_class2_lst:
        feature2_class2_location.append(i)

len(feature2_class2_location)/len(result2)

# 总吞吐量 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class3.iloc[:,0],feature2_class3.iloc[:,1],c=
result3)
plt.title("PDCP 流量 Type3 ",fontsize=25)
plt.show()

#断层选多少？
dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)

feature2_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<3:
        feature2_class3_lst.append(k)

feature2_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature2_class3_lst:
        feature2_class3_location.append(i)
#断层选多少？
dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)
```

```
360.
361.   feature2_class3_lst=[]
362.   for k in Counter(result3).keys():
363.       if Counter(result3)[k]<3:
364.           feature2_class3_lst.append(k)
365.
366.   feature2_class3_location=[]
367.   for i in range(len(result3)):
368.       if result3[i] in feature2_class3_lst:
369.           feature2_class3_location.append(i)
370.
371.   len(feature2_class3_location)/len(result3)
372.
373.   len(feature2_class3_location)/len(result3)
```

问题二：按小区分类

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  import matplotlib as mpl
4.  import seaborn as sns
5.  import warnings; warnings.simplefilter('ignore')
6.  from sklearn.cluster import DBSCAN
7.  from sklearn.preprocessing import  StandardScaler
8.  from sklearn import preprocessing
9.  from collections import Counter
10. from sklearn.cluster import KMeans
11. import mglearn
12. from sklearn import metrics
13.
14. df = pd.read_csv('dataAstandard.csv')
15.
16. #平均激活用户数
17. cluster_3 = [1 ,0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
        2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,\
18.  1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 1, 2, 2, 0, 1, 1]
19. c1, c2, c3 = [], [], []
20. for i in range(len(cluster_3)):
21.     if cluster_3[i]==0:
22.         c1.append(int("260190"+str(i+1)))
23.     if cluster_3[i]==1:
24.         c2.append(int("260190"+str(i+1)))
25.     if cluster_3[i]==2:
26.         c3.append(int("260190"+str(i+1)))
27.
```

```
cluster_3_1 = df.loc[df["小区编号"].isin(c1),:]
cluster_3_2 = df.loc[df["小区编号"].isin(c2),:]
cluster_3_3 = df.loc[df["小区编号"].isin(c3),:]

feature3_class1=cluster_3_1[["时段","平均激活用户数"]]
feature3_class2=cluster_3_2[["时段","平均激活用户数"]]
feature3_class3=cluster_3_3[["时段","平均激活用户数"]]

scaler=StandardScaler()

scaler.fit(feature3_class1)
feature3_class1_sclaed=scaler.transform(feature3_class1)

scaler.fit(feature3_class2)
feature3_class2_sclaed=scaler.transform(feature3_class2)

scaler.fit(feature3_class3)
feature3_class3_sclaed=scaler.transform(feature3_class3)

# 平均激活用户数 class1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)
plt.figure(figsize=(10,5))
cmap2=plt.get_cmap('Set2')
plt.scatter(feature3_class1.iloc[:,0],feature3_class1.iloc[:,1],c=res
ult1)
plt.title("平均激活用户数 Type1 ",fontsize=25)
plt.show()

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)

feature3_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<10:
        feature3_class1_lst.append(k)

feature3_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature3_class1_lst:
        feature3_class1_location.append(i)
```

```
71.
72. len(feature3_class1_location)/len(result1)
73.
74. # 平均激活用户数 class2
75. plt.rcParams['font.sans-serif'] = ['SimHei']
76. plt.rcParams['axes.unicode_minus'] = False
77.
78. dbscan2=DBSCAN(eps=0.14,min_samples=1)
79. result2=dbscan2.fit_predict(feature3_class2_sclaed)
80. print(Counter(result2))
81. plt.figure(figsize=(10,5))
82. plt.scatter(feature3_class2.iloc[:,0],feature3_class2.iloc[:,1],c=res
    ult2)
83. plt.title("平均激活用户数 Type2 ",fontsize=25)
84. plt.show()
85.
86. dbscan2=DBSCAN(eps=0.14,min_samples=1)
87. result2=dbscan2.fit_predict(feature3_class2_sclaed)
88.
89. feature3_class2_lst=[]
90. for k in Counter(result2).keys():
91.     if Counter(result2)[k]<5:
92.         feature3_class2_lst.append(k)
93.
94. feature3_class2_location=[]
95. for i in range(len(result2)):
96.     if result2[i] in feature3_class2_lst:
97.         feature3_class2_location.append(i)
98.
99. len(feature3_class2_location)/len(result2)
100.
101.   # 平均激活用户数 class3
102.   plt.rcParams['font.sans-serif'] = ['SimHei']
103.   plt.rcParams['axes.unicode_minus'] = False
104.
105.   dbscan3=DBSCAN(eps=0.14,min_samples=1)
106.   result3=dbscan3.fit_predict(feature3_class3_sclaed)
107.   print(Counter(result3))
108.   plt.figure(figsize=(10,5))
109.   plt.scatter(feature3_class3.iloc[:,0],feature3_class3.iloc[:,1],c=
    result3)
110.   plt.title("平均激活用户数 Type3 ",fontsize=25)
111.   plt.show()
112.
```

```
dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature3_class3_sclaed)

feature3_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature3_class3_lst.append(k)

feature3_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature3_class3_lst:
        feature3_class3_location.append(i)

len(feature3_class3_location)/len(result3)

#平均用户数
cluster_1 = [1 ,0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1,
1, 2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,\
 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 2, 0, 2, 0, 1, 1]
c1, c2, c3 = [], [], []
for i in range(len(cluster_1)):
    if cluster_1[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_1[i]==1:
        c2.append(int("260190"+str(i+1)))
    if cluster_1[i]==2:
        c3.append(int("260190"+str(i+1)))

cluster_1_1 = df.loc[df["小区编号"].isin(c1),:]
cluster_1_2 = df.loc[df["小区编号"].isin(c2),:]
cluster_1_3 = df.loc[df["小区编号"].isin(c3),:]

feature1_class1=cluster_1_1[["时段","小区内的平均用户数"]]
feature1_class2=cluster_1_2[["时段","小区内的平均用户数"]]
feature1_class3=cluster_1_3[["时段","小区内的平均用户数"]]

scaler=StandardScaler()
scaler.fit(feature1_class1)
feature1_class1_sclaed=scaler.transform(feature1_class1)

scaler.fit(feature1_class2)
feature1_class2_sclaed=scaler.transform(feature1_class2)

scaler.fit(feature1_class3)
```

```
feature1_class3_sclaed=scaler.transform(feature1_class3)

# 平均用户数 class1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature1_class1_sclaed)
print(Counter(result1))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class1.iloc[:,0],feature1_class1.iloc[:,1],c=result1)
plt.title("平均用户数 Type1 ",fontsize=25)
plt.show()

dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature1_class1_sclaed)

feature1_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<5:
        feature1_class1_lst.append(k)

feature1_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature1_class1_lst:
        feature1_class1_location.append(i)

len(feature1_class1_location)/len(result1)

# 平均用户数 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan2=DBSCAN(eps=0.12,min_samples=1)
result2=dbscan2.fit_predict(feature1_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class2.iloc[:,0],feature1_class2.iloc[:,1],c=result2,s=30)
plt.title("平均用户数 Type2 ",fontsize=25)
plt.show()

dbscan2=DBSCAN(eps=0.12,min_samples=1)
```

```
result2=dbscan2.fit_predict(feature1_class2_sclaed)

feature1_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<3:
        feature1_class2_lst.append(k)

feature1_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature1_class2_lst:
        feature1_class2_location.append(i)

len(feature1_class2_location)/len(result2)

# 平均用户数 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class3.iloc[:,0],feature1_class3.iloc[:,1],c=result3,s=30)
plt.title("平均用户数 Type3 ",fontsize=25)
plt.show()

#多少作为断层?
dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)

feature1_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature1_class3_lst.append(k)

feature1_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature1_class3_lst:
        feature1_class3_location.append(i)

len(feature1_class3_location)/len(result3)

#总吞吐量
```

```
  cluster_2 = [0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,\
              0, 0, 0, 1, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0]
  c1, c2, c3 = [], [], []
  for i in range(len(cluster_2)):
      if cluster_2[i]==0:
          c1.append(int("260190"+str(i+1)))
      if cluster_2[i]==1:
          c2.append(int("260190"+str(i+1)))
      if cluster_2[i]==2:
          c3.append(int("260190"+str(i+1)))

  cluster_2_1 = df.loc[df["小区编号"].isin(c1),:]
  cluster_2_2 = df.loc[df["小区编号"].isin(c2),:]
  cluster_2_3 = df.loc[df["小区编号"].isin(c3),:]

  feature2_class1=cluster_2_1[["时段","总吞吐量"]]
  feature2_class2=cluster_2_2[["时段","总吞吐量"]]
  feature2_class3=cluster_2_3[["时段","总吞吐量"]]

  #总吞吐量
  scaler=StandardScaler()

  scaler.fit(feature2_class1)
  feature2_class1_sclaed=scaler.transform(feature2_class1)

  scaler.fit(feature2_class2)
  feature2_class2_sclaed=scaler.transform(feature2_class2)

  scaler.fit(feature2_class3)
  feature2_class3_sclaed=scaler.transform(feature2_class3)

  # 总吞吐量 class1
  plt.rcParams['font.sans-serif'] = ['SimHei']
  plt.rcParams['axes.unicode_minus'] = False

  dbscan1=DBSCAN(eps=0.14,min_samples=1)
  result1=dbscan1.fit_predict(feature2_class1_sclaed)
  print(Counter(result1))
  plt.figure(figsize=(10,5))
  plt.scatter(feature2_class1.iloc[:,0],feature2_class1.iloc[:,1],c=result1)
  plt.title("PDCP 流量 Type1 ",fontsize=25)
```

```
282.  plt.show()
283.
284.
285.  dbscan1=DBSCAN(eps=0.14,min_samples=1)
286.  result1=dbscan1.fit_predict(feature2_class1_sclaed)
287.
288.  feature2_class1_lst=[]
289.  for k in Counter(result1).keys():
290.      if Counter(result1)[k]<5:
291.          feature2_class1_lst.append(k)
292.
293.  feature2_class1_location=[]
294.  for i in range(len(result1)):
295.      if result1[i] in feature2_class1_lst:
296.          feature2_class1_location.append(i)
297.
298.  len(feature2_class1_location)/len(result1)
299.
300.  #总吞吐量 class2
301.  plt.rcParams['font.sans-serif'] = ['SimHei']
302.  plt.rcParams['axes.unicode_minus'] = False
303.
304.  esp_lst=[0.03,0.04,0.05]
305.  min_samples_lst=[0,1]
306.
307.  dbscan2=DBSCAN(eps=0.14,min_samples=1)
308.  result2=dbscan1.fit_predict(feature2_class2_sclaed)
309.  print(Counter(result2))
310.  plt.figure(figsize=(10,5))
311.  plt.scatter(feature2_class2.iloc[:,0],feature2_class2.iloc[:,1],c=
    result2)
312.  plt.title("PDCP 流量 Type2 ",fontsize=25)
313.  plt.show()
314.
315.
316.  dbscan2=DBSCAN(eps=0.14,min_samples=1)
317.  result2=dbscan1.fit_predict(feature2_class2_sclaed)
318.
319.  feature2_class2_lst=[]
320.  for k in Counter(result2).keys():
321.      if Counter(result2)[k]<2:
322.          feature2_class2_lst.append(k)
323.
324.  feature2_class2_location=[]
```

```
for i in range(len(result2)):
    if result2[i] in feature2_class2_lst:
        feature2_class2_location.append(i)

len(feature2_class2_location)/len(result2)

# 总吞吐量 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature2_class3.iloc[:,0],feature2_class3.iloc[:,1],c=result3)
plt.title("PDCP 流量 Type3 ",fontsize=25)
plt.show()

#断层选多少？
dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)

feature2_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<3:
        feature2_class3_lst.append(k)

feature2_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature2_class3_lst:
        feature2_class3_location.append(i)
#断层选多少？
dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature2_class3_sclaed)

feature2_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<3:
        feature2_class3_lst.append(k)

feature2_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature2_class3_lst:
```

```
            feature2_class3_location.append(i)

    len(feature2_class3_location)/len(result3)

    len(feature2_class3_location)/len(result3)
```

问题二：LSTM-指标一、二、三

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import warnings; warnings.simplefilter('ignore')
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import  StandardScaler
from sklearn import preprocessing
from collections import Counter
from sklearn.cluster import KMeans
import mglearn
from sklearn import metrics

df = pd.read_csv('dataAstandard.csv')

#平均激活用户数
cluster_3 = [1 ,0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
    2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,\
 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 1, 2, 2, 0, 1, 1]
c1, c2, c3 = [], [], []
for i in range(len(cluster_3)):
    if cluster_3[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_3[i]==1:
        c2.append(int("260190"+str(i+1)))
    if cluster_3[i]==2:
        c3.append(int("260190"+str(i+1)))

cluster_3_1 = df.loc[df["小区编号"].isin(c1),:]
cluster_3_2 = df.loc[df["小区编号"].isin(c2),:]
cluster_3_3 = df.loc[df["小区编号"].isin(c3),:]

feature3_class1=cluster_3_1[["时段","平均激活用户数"]]
feature3_class2=cluster_3_2[["时段","平均激活用户数"]]
feature3_class3=cluster_3_3[["时段","平均激活用户数"]]

```

```
scaler=StandardScaler()

scaler.fit(feature3_class1)
feature3_class1_sclaed=scaler.transform(feature3_class1)

scaler.fit(feature3_class2)
feature3_class2_sclaed=scaler.transform(feature3_class2)

scaler.fit(feature3_class3)
feature3_class3_sclaed=scaler.transform(feature3_class3)

# 平均激活用户数 class1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)
plt.figure(figsize=(10,5))
cmap2=plt.get_cmap('Set2')
plt.scatter(feature3_class1.iloc[:,0],feature3_class1.iloc[:,1],c=res
ult1)
plt.title("平均激活用户数 Type1 ",fontsize=25)
plt.show()

dbscan1=DBSCAN(eps=0.13,min_samples=1)
result1=dbscan1.fit_predict(feature3_class1_sclaed)

feature3_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<10:
        feature3_class1_lst.append(k)

feature3_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature3_class1_lst:
        feature3_class1_location.append(i)

len(feature3_class1_location)/len(result1)

# 平均激活用户数 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan2=DBSCAN(eps=0.14,min_samples=1)
```

```
result2=dbscan2.fit_predict(feature3_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature3_class2.iloc[:,0],feature3_class2.iloc[:,1],c=result2)
plt.title("平均激活用户数 Type2 ",fontsize=25)
plt.show()

dbscan2=DBSCAN(eps=0.14,min_samples=1)
result2=dbscan2.fit_predict(feature3_class2_sclaed)

feature3_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<5:
        feature3_class2_lst.append(k)

feature3_class2_location=[]
for i in range(len(result2)):
    if result2[i] in feature3_class2_lst:
        feature3_class2_location.append(i)

len(feature3_class2_location)/len(result2)

# 平均激活用户数 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature3_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature3_class3.iloc[:,0],feature3_class3.iloc[:,1],c=result3)
plt.title("平均激活用户数 Type3 ",fontsize=25)
plt.show()

dbscan3=DBSCAN(eps=0.14,min_samples=1)
result3=dbscan3.fit_predict(feature3_class3_sclaed)

feature3_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature3_class3_lst.append(k)

```

```
feature3_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature3_class3_lst:
        feature3_class3_location.append(i)

len(feature3_class3_location)/len(result3)

#平均用户数
cluster_1 = [1 ,0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1,
1, 2, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,\
 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 2, 0, 2, 0, 1, 1]
c1, c2, c3 = [], [], []
for i in range(len(cluster_1)):
    if cluster_1[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_1[i]==1:
        c2.append(int("260190"+str(i+1)))
    if cluster_1[i]==2:
        c3.append(int("260190"+str(i+1)))

cluster_1_1 = df.loc[df["小区编号"].isin(c1),:]
cluster_1_2 = df.loc[df["小区编号"].isin(c2),:]
cluster_1_3 = df.loc[df["小区编号"].isin(c3),:]

feature1_class1=cluster_1_1[["时段","小区内的平均用户数"]]
feature1_class2=cluster_1_2[["时段","小区内的平均用户数"]]
feature1_class3=cluster_1_3[["时段","小区内的平均用户数"]]

scaler=StandardScaler()
scaler.fit(feature1_class1)
feature1_class1_sclaed=scaler.transform(feature1_class1)

scaler.fit(feature1_class2)
feature1_class2_sclaed=scaler.transform(feature1_class2)

scaler.fit(feature1_class3)
feature1_class3_sclaed=scaler.transform(feature1_class3)

# 平均用户数 class1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature1_class1_sclaed)
```

```
print(Counter(result1))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class1.iloc[:,0],feature1_class1.iloc[:,1],c=result1)
plt.title("平均用户数 Type1 ",fontsize=25)
plt.show()

dbscan1=DBSCAN(eps=0.14,min_samples=1)
result1=dbscan1.fit_predict(feature1_class1_sclaed)

feature1_class1_lst=[]
for k in Counter(result1).keys():
    if Counter(result1)[k]<5:
        feature1_class1_lst.append(k)

feature1_class1_location=[]
for i in range(len(result1)):
    if result1[i] in feature1_class1_lst:
        feature1_class1_location.append(i)

len(feature1_class1_location)/len(result1)

# 平均用户数 class2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan2=DBSCAN(eps=0.12,min_samples=1)
result2=dbscan2.fit_predict(feature1_class2_sclaed)
print(Counter(result2))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class2.iloc[:,0],feature1_class2.iloc[:,1],c=result2,s=30)
plt.title("平均用户数 Type2 ",fontsize=25)
plt.show()

dbscan2=DBSCAN(eps=0.12,min_samples=1)
result2=dbscan2.fit_predict(feature1_class2_sclaed)

feature1_class2_lst=[]
for k in Counter(result2).keys():
    if Counter(result2)[k]<3:
        feature1_class2_lst.append(k)

feature1_class2_location=[]
```

```
for i in range(len(result2)):
    if result2[i] in feature1_class2_lst:
        feature1_class2_location.append(i)

len(feature1_class2_location)/len(result2)

# 平均用户数 class3
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)
print(Counter(result3))
plt.figure(figsize=(10,5))
plt.scatter(feature1_class3.iloc[:,0],feature1_class3.iloc[:,1],c=result3,s=30)
plt.title("平均用户数 Type3 ",fontsize=25)
plt.show()

dbscan3=DBSCAN(eps=0.13,min_samples=1)
result3=dbscan3.fit_predict(feature1_class3_sclaed)

feature1_class3_lst=[]
for k in Counter(result3).keys():
    if Counter(result3)[k]<5:
        feature1_class3_lst.append(k)

feature1_class3_location=[]
for i in range(len(result3)):
    if result3[i] in feature1_class3_lst:
        feature1_class3_location.append(i)

len(feature1_class3_location)/len(result3)

#总吞吐量
cluster_2 = [0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0]
c1, c2, c3 = [], [], []
for i in range(len(cluster_2)):
    if cluster_2[i]==0:
        c1.append(int("260190"+str(i+1)))
    if cluster_2[i]==1:
        c2.append(int("260190"+str(i+1)))
```

```
        if cluster_2[i]==2:
            c3.append(int("260190"+str(i+1)))

    cluster_2_1 = df.loc[df["小区编号"].isin(c1),:]
    cluster_2_2 = df.loc[df["小区编号"].isin(c2),:]
    cluster_2_3 = df.loc[df["小区编号"].isin(c3),:]

    feature2_class1=cluster_2_1[["时段","总吞吐量"]]
    feature2_class2=cluster_2_2[["时段","总吞吐量"]]
    feature2_class3=cluster_2_3[["时段","总吞吐量"]]

    #总吞吐量
    scaler=StandardScaler()

    scaler.fit(feature2_class1)
    feature2_class1_sclaed=scaler.transform(feature2_class1)

    scaler.fit(feature2_class2)
    feature2_class2_sclaed=scaler.transform(feature2_class2)

    scaler.fit(feature2_class3)
    feature2_class3_sclaed=scaler.transform(feature2_class3)

    # 总吞吐量 class1
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    dbscan1=DBSCAN(eps=0.14,min_samples=1)
    result1=dbscan1.fit_predict(feature2_class1_sclaed)
    print(Counter(result1))
    plt.figure(figsize=(10,5))
    plt.scatter(feature2_class1.iloc[:,0],feature2_class1.iloc[:,1],c=
result1)
    plt.title("PDCP 流量 Type1 ",fontsize=25)
    plt.show()


    dbscan1=DBSCAN(eps=0.14,min_samples=1)
    result1=dbscan1.fit_predict(feature2_class1_sclaed)

    feature2_class1_lst=[]
    for k in Counter(result1).keys():
        if Counter(result1)[k]<5:
            feature2_class1_lst.append(k)
```

```
291.
292.   feature2_class1_location=[]
293.   for i in range(len(result1)):
294.       if result1[i] in feature2_class1_lst:
295.           feature2_class1_location.append(i)
296.
297.   len(feature2_class1_location)/len(result1)
298.
299.   #总吞吐量 class2
300.   plt.rcParams['font.sans-serif'] = ['SimHei']
301.   plt.rcParams['axes.unicode_minus'] = False
302.
303.   esp_lst=[0.03,0.04,0.05]
304.   min_samples_lst=[0,1]
305.
306.   dbscan2=DBSCAN(eps=0.14,min_samples=1)
307.   result2=dbscan1.fit_predict(feature2_class2_sclaed)
308.   print(Counter(result2))
309.   plt.figure(figsize=(10,5))
310.   plt.scatter(feature2_class2.iloc[:,0],feature2_class2.iloc[:,1],c=
    result2)
311.   plt.title("PDCP 流量 Type2 ",fontsize=25)
312.   plt.show()
313.
314.
315.   dbscan2=DBSCAN(eps=0.14,min_samples=1)
316.   result2=dbscan1.fit_predict(feature2_class2_sclaed)
317.
318.   feature2_class2_lst=[]
319.   for k in Counter(result2).keys():
320.       if Counter(result2)[k]<2:
321.           feature2_class2_lst.append(k)
322.
323.   feature2_class2_location=[]
324.   for i in range(len(result2)):
325.       if result2[i] in feature2_class2_lst:
326.           feature2_class2_location.append(i)
327.
328.   len(feature2_class2_location)/len(result2)
329.
330.   # 总吞吐量 class3
331.   plt.rcParams['font.sans-serif'] = ['SimHei']
332.   plt.rcParams['axes.unicode_minus'] = False
333.
```

```
334.  dbscan3=DBSCAN(eps=0.14,min_samples=1)
335.  result3=dbscan3.fit_predict(feature2_class3_sclaed)
336.  print(Counter(result3))
337.  plt.figure(figsize=(10,5))
338.  plt.scatter(feature2_class3.iloc[:,0],feature2_class3.iloc[:,1],c=
    result3)
339.  plt.title("PDCP 流量 Type3 ",fontsize=25)
340.  plt.show()
341.  dbscan3=DBSCAN(eps=0.14,min_samples=1)
342.  result3=dbscan3.fit_predict(feature2_class3_sclaed)
343.
344.  feature2_class3_lst=[]
345.  for k in Counter(result3).keys():
346.      if Counter(result3)[k]<3:
347.          feature2_class3_lst.append(k)
348.
349.  feature2_class3_location=[]
350.  for i in range(len(result3)):
351.      if result3[i] in feature2_class3_lst:
352.          feature2_class3_location.append(i)
353.  dbscan3=DBSCAN(eps=0.14,min_samples=1)
354.  result3=dbscan3.fit_predict(feature2_class3_sclaed)
355.
356.  feature2_class3_lst=[]
357.  for k in Counter(result3).keys():
358.      if Counter(result3)[k]<3:
359.          feature2_class3_lst.append(k)
360.
361.  feature2_class3_location=[]
362.  for i in range(len(result3)):
363.      if result3[i] in feature2_class3_lst:
364.          feature2_class3_location.append(i)
365.
366.  len(feature2_class3_location)/len(result3)
367.
368.  len(feature2_class3_location)/len(result3)
```