# 预测并计算快递需求数量对经济发展的重要性

摘 要

网络购物已成为现代消费模式的重要组成部分，快递服务需求也因此飞速增长，为我国经济发展做出了重要贡献。准确地预测快递运输需求数量对于快递公司布局仓库站点、节约存储成本、规划运输线路等具有重要的意义。本文分析了网络购物消费模式对快递需求的影响，利用回归模型建立了快递运输需求预测模型，并对该模型进行了实证分析和验证。

首先，针对问题一，我们需要对数据进行预处理。在预处理阶段，我们会选择综合评价模型的指标。对于每个城市，我们将考虑其增长/减少趋势，并且要处理数据文件中的缺失值和异常值等问题。我们将使用收获量、发货量和相关性等指标来衡量城市的表现。为了提高模型的准确率，我们将采用平均相关性作为一个指标。接下来，我们将根据各个指标来评估城市的表现，并计算出综合评分。我们将使用加权平均法，根据一定的权重对各指标进行加权求和。为了更准确地反映实际情况，我们还会根据实际值进行调整。最后，我们将对城市进行综合打分，并按照评分的次序进行排列。

在问题二中，我们进行了时间序列分析，并建立了一个ARIMA模型。我们使用这个模型对快递运输量的指标进行预测，并取得了令人满意的结果。这一方法不仅加快了模型的求解过程，还对附件数据中的快递总量进行了详尽的分析。我们的预测结果在预测范围内表现良好，为我们提供了更深入的洞察和理解。

在问题三中，我们采用了类似问题二的方法，但是我们针对是否能正常发货建立了一个时间预测模型。这个模型基于在规定时间内建立的时间模型，通过使用0-1标记来表示日期是否存在缺失问题。具体而言，我们将数据转化为0-1形式，如果数据大于0.5，则标记为1，否则标记为0，用以判断是否能正常发货。在确定了是否能正常发货后，我们使用问题二中的ARIMA模型对快递运输数量进行预测。对于那些被确定为能正常发货的数据，我们根据相应的城市对应的模型代码进行修改，从而得到其快递运输数量的预测结果。通过优化方法，能准确地预测快递运输数量，并考虑到是否能正常发货的因素。

在问题四中，我们首先要考虑如何最佳安排物流运输活动，以实现最大效益、最小成本或最快速度等目标。这个问题涉及到货物种类和数量、运输距离、交通状况、天气因素、运输成本、运输时间、配送顺序等多个因素。为了解决这些问题，我们采用了数学模型和运筹学等方法，以求得最优的物流配送方案。在本问题中，我们运用了线性规划方法来求解最优方案。此外，为了进一步优化求解过程，我们还考虑了使用LSTM模型的方法。通过综合线性规划和LSTM模型的结果，我们计算出它们的成绩均值，并分析相关数据。在LSTM模型中，我们使用目标函数和约束条件进行规划建模。我们使用0-1矩阵将所有相邻道路的成本计入考虑，并将决策变量代入成本表达式，进而对所有相邻道路求和以获得目标函数。通过优化方法，能更深入地解决问题四，找到最优的物流配送方案，并综合考虑多种因素来达到更优的结果。

针对问题五，建立数学模型，按季度估计固定需求常数，并验证其准确性。将指定季度、指定“发货-收货”站点城市对的固定需求常数，以及当季度所有“发货-收货”城市对的固定需求常数总和，填入表5。

【关键词】ARIMA模型预测;时间序列;实证分析;LSTM模型预测;线性分析;

# 问题重述

网络购物作为一种重要的消费方式，带动着快递服务需求飞速增长，为我国经济发展做出了重要贡献。准确地预测快递运输需求数量对于快递公司布局仓库站点、节约存储成本、规划运输线路等具有重要的意义。

问题**1**：附件1为该快递公司记录的2018年4月19日—2019年4月17日的站点城市之间(发货城市-收货城市)的快递运输数据，请从收货量、发货量、快递数量增长/减少趋势、相关性等多角度考虑，建立数学模型，对各站点城市的重要程度进行综合排序，并给出重要程度排名前5的站点城市名称，将结果填入表1。

表1 问题1结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 排序 | 1 | 2 | 3 | 4 | 5 |
| 城市名称 | L | G | V | Q | R |

问题**2**：请利用附件1数据，建立数学模型，预测2019年4月18日和2019年4月19日各“发货-收货”站点城市之间快递运输数量，以及当日所有“发货-收货”站点城市之间的总快递运输数量，并在表2中填入指定的站点城市之间的快递运输数量，以及当日所有“发货-收货”站点城市之间的总快递运输数量。

表2 问题2结果

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | “发货-收货”城市之间的快递运输数量 | | 所有“发货-收货”城市之间的总快递运输数量 |
| 2019年4月18日 | M-U | 157 | 847 |
| Q-V | 52 |
| K-L | 70 |
| G-V | 568 |
| 2019年4月19日 | V-G | 464 | 888 |
| A-Q | 128 |
| D-A | 45 |
| L-K | 251 |

问题**3**：附件2为该快递公司记录的2020年4月28日—2023年4月27日的快递运输数量。由于受到突发事件影响，部分城市之间快递线路无法正常运输，导致站点城市之间无法正常发货或收货(无数据表示无法正常收发货，0表示无发货需求)。请利用附件2数据，建立数学模型，预测2023年4月28日和2023年4月29日可正常“发货-收货”的站点城市对(发货城市-收货城市)，并判断表3中指定的站点城市对是否能正常发货，如果能正常发货，给出对应的快递运输数量，并将结果填入表3。

表3 问题3结果

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | “发货-收货”站点城市对 | 是否能正常发货  (填写“是”或“否”) | 快递运输数量 |
| 2023年4月28日 | I-S | 否 | / |
| M-G | 是 | 43 |
| S-Q | 是 | 99 |
| V-A | 是 | 122 |
| Y-L | 是 | 36 |
| 2023年4月29日 | D-R | 否 | / |
| J-K | 是 | 205 |
| Q-O | 否 | / |
| U-O | 否 | / |
| Y-W | 否 | / |

问题**4**：图1给出了所有站点城市间的铁路运输网络，铁路运输成本由以下公式计算：。在本题中，假设实际装货量允许超过额定装货量。所有铁路的固定成本、额定装货量在附件3中给出。在运输快递时，要求每个“发货-收货”站点城市对之间使用的路径数不超过5条，请建立数学模型，给出该快递公司成本最低的运输方案。利用附件2和附件3的数据，计算该公司2023年4月23—27日每日的最低运输成本，填入表4。

备注：为了方便计算，不对快递重量和大小进行区分，假设每件快递的重量为单位1。仅考虑运输成本，不考虑中转等其它成本。

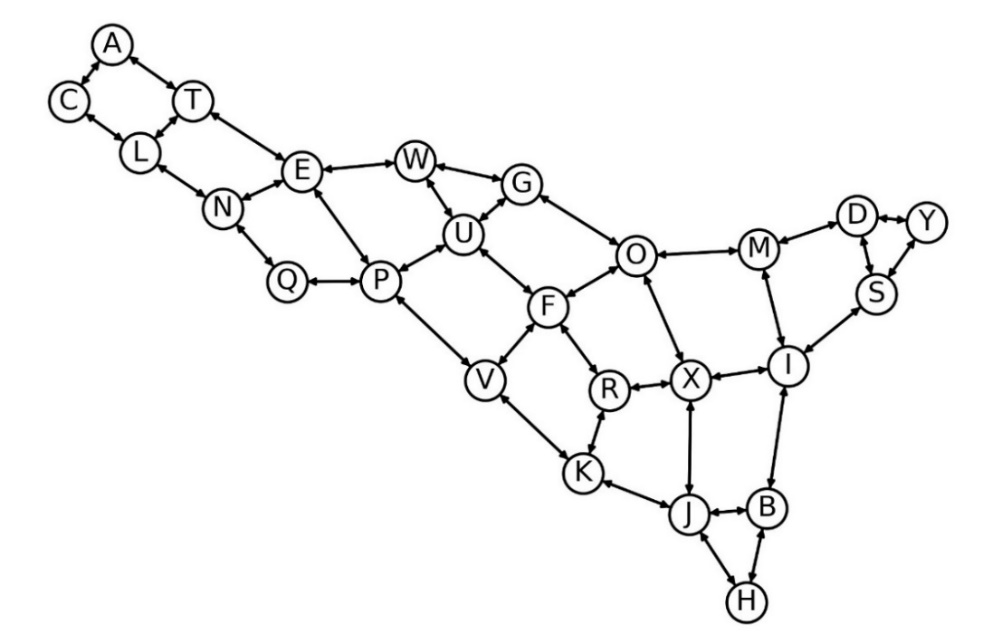


图1 站点城市间铁路运输网络

表4 问题4结果

|  |  |
| --- | --- |
| 日期 | 最低运输成本 |
| 2023年4月23日 | 16432 |
| 2023年4月24日 | 22964 |
| 2023年4月25日 | 18677 |
| 2023年4月26日 | 14567 |
| 2023年4月27日 | 13786 |

问题5：通常情况下，快递需求由两部分组成，一部分为固定需求，这部分需求来源于日常必要的网购消费(一般不能简单的认定为快递需求历史数据的最小值，通常小于需求的最小值)；另一部分为非固定需求，这部分需求通常有较大波动，受时间等因素的影响较大。假设在同一季度中，同一“发货-收货”站点城市对的固定需求为一确定常数(以下简称为固定需求常数)；同一“发货-收货”站点城市对的非固定需求服从某概率分布(该分布的均值和标准差分别称为非固定需求均值、非固定需求标准差)。请利用附件2中的数据，不考虑已剔除数据、无发货需求数据、无法正常发货数据，解决以下问题。

(1) 建立数学模型，按季度估计固定需求常数，并验证其准确性。将指定季度、指定“发货-收货”站点城市对的固定需求常数，以及当季度所有“发货-收货”城市对的固定需求常数总和，填入表5。

(2) 给出非固定需求概率分布估计方法，并将指定季度、指定“发货-收货”站点城市对的非固定需求均值、标准差，以及当季度所有“发货-收货”城市对的非固定需求均值总和、非固定需求标准差总和，填入表5。

表5 问题5结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 季度 | 2022年第三季度  (7—9月) | | 2023年第一季度  (1—3月) | |
| “发货-收货”站点城市对 | V-N | V-Q | J-I | O-G |
| 固定需求常数 | 0 | 0 | 4 | 0 |
| 非固定需求均值 | 105.4186 | 86.5072 | 241.0997 | 141.0111 |
| 非固定需求标准差 | 73.4808 | 51.7697 | 120.5498 | 73.8952 |
| 固定需求常数总和 | 0 | | 4 | |
| 非固定需求均值总和 | 191.9258 | | 382.1108 | |
| 非固定需求标准差总和 | 125.2505 | | 194.445 | |

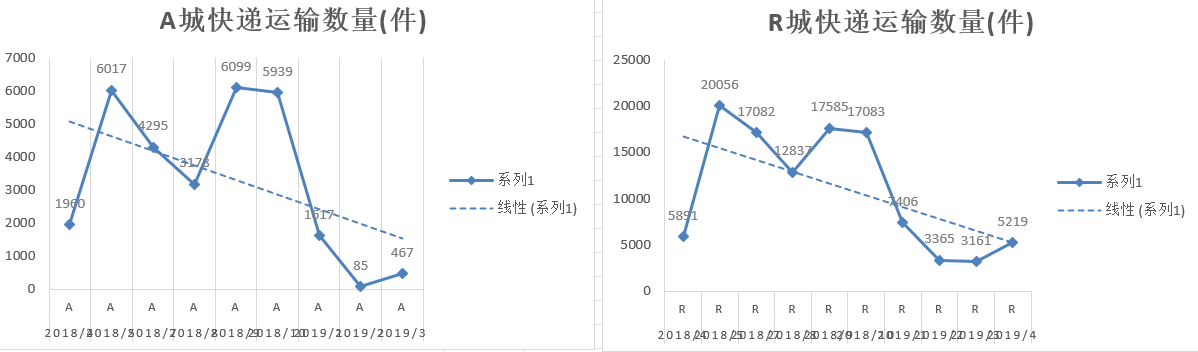
# 符号说明

* Mmax........................................................................................................................最大收货量
* Mmin ........................................................................................................................最小收货量
* MAverage.................................................................................................................平均收货量
* Lmaxj.............................................................................................................日收货最大变化量
* Lmaxi.............................................................................................................日发货最大变化量
* LAverage.......................................................................................................日收获平均变化量
* LAverage.......................................................................................................日发货平均变化量
* At..............................................................................................t天某市到莫地快递运输量矩阵
* Rote<>..................................................................................................................运输路线矩阵
* Sumon<>.........................................................................运输路线对应的运输量的正整数矩阵

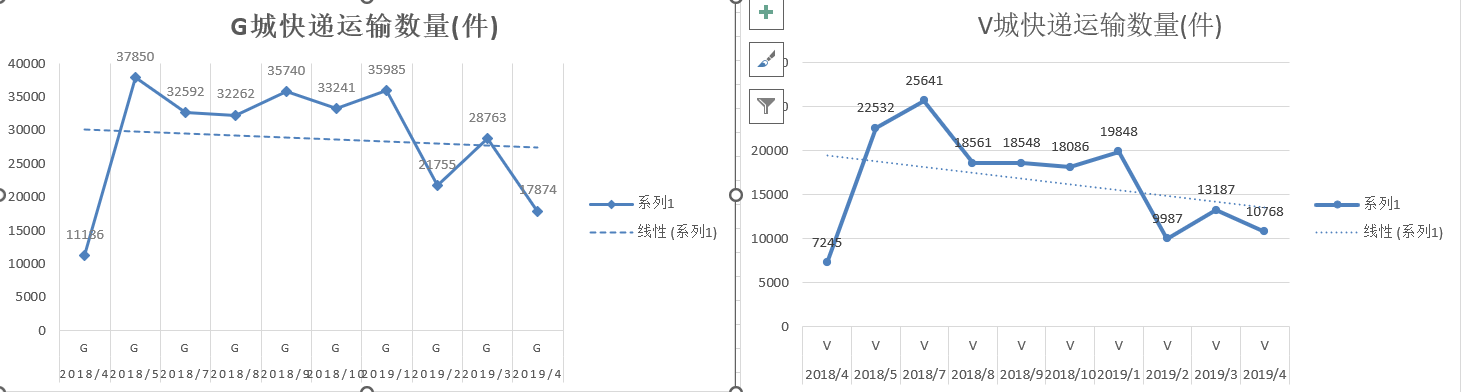
# 模型的建立与求解

3.1问题一分析流程  
3.1.1数据源：副本附件1(Attachment 1)2023-51MCM-Problem B）

3.1.2根据附件一用excel建立每月城市的快递运输数量

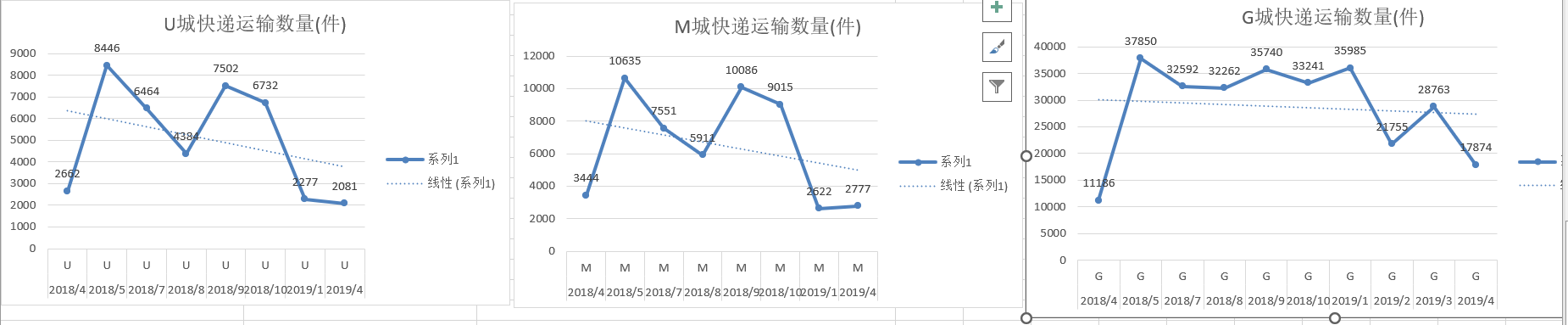


(3.1.2.1)



(3.1.2.2)

(3.1.2.3)



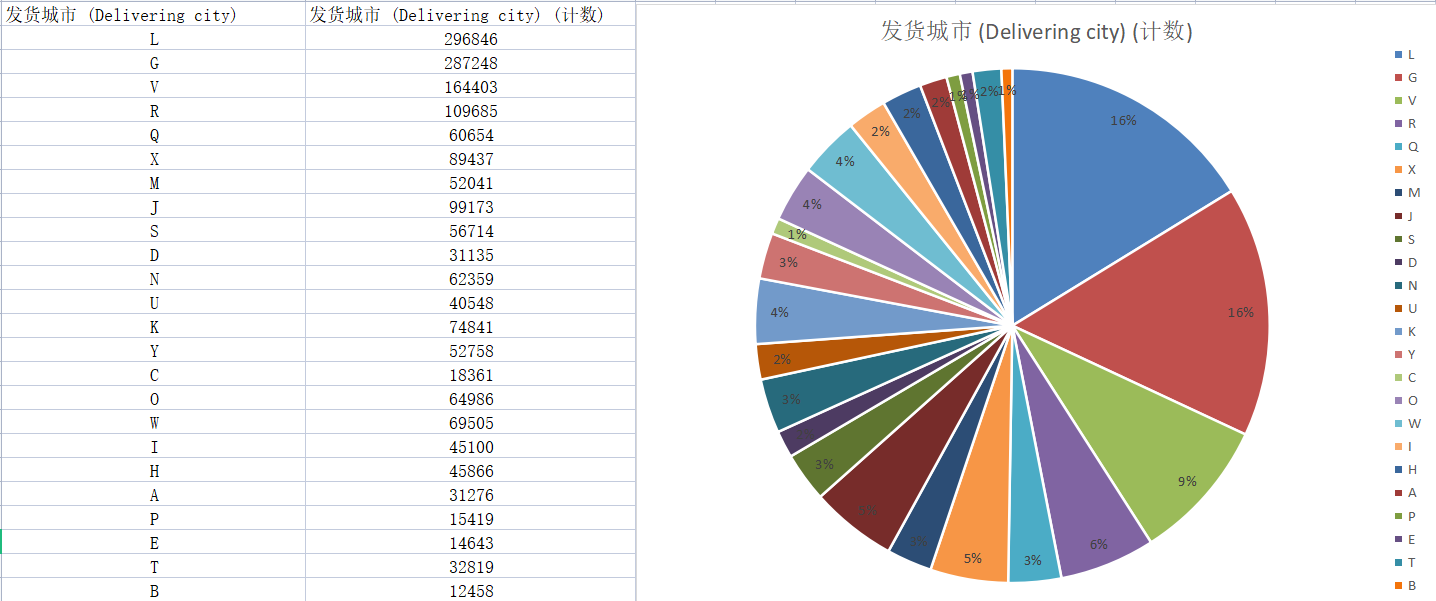
(3.1.1.4)

通过对(3.1.1.1-3.1.1.4)图分析，采用了对数据进行进行线性折线式可视化分析，我们从中选出了这几个城市对1问更加贴切，通过快递运输数量和线性相关选择这8座城市作为参考对象，并从这8组中选择需要的部分进行验证。这些城市的线性均呈现下趋势，我们考虑到这可能是一个季的循环，不同季的特点，均在5月份的时候较多。

3.1.3对附件一中的数据进行筛选，根据数据做饼图分析并比较。在这里我们进一步选择了L,A,R,G,V等城市，并对3.1.2中的信息进行了筛选，结合饼图例可以更好地看出结果。

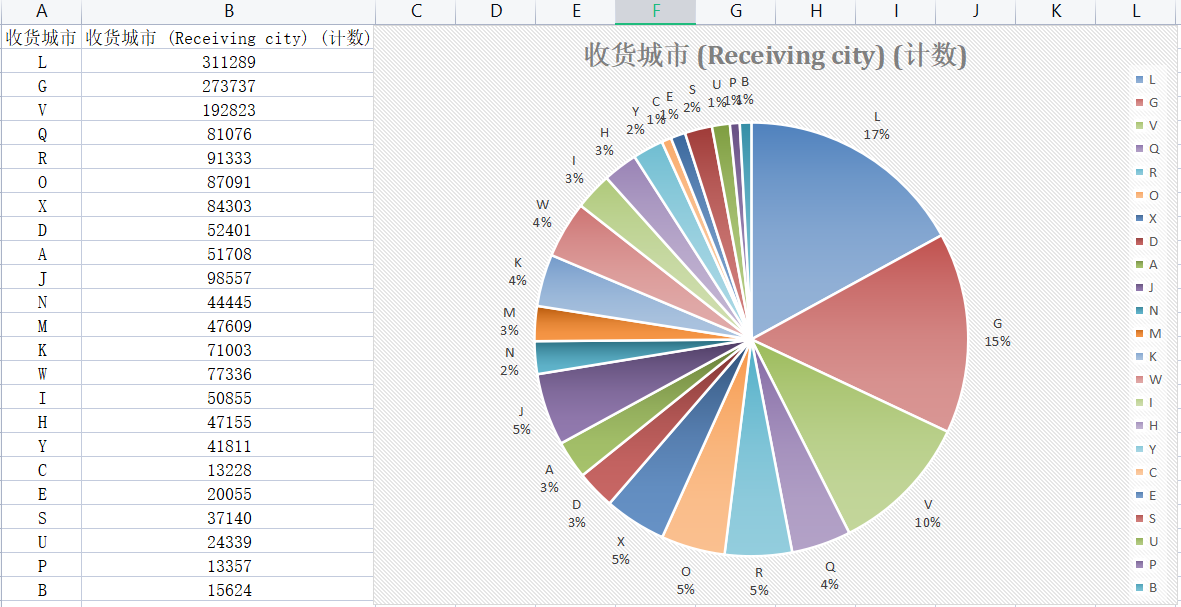
饼图分析

发货城市占比及排名 从图中可以分析得到，发货城市的排名，通过数据筛选，我们可以看到发货城市数量从高到低前五个是L、G、V、 R、Q，通过对发货量的指标考核我们得出了这五座城市。



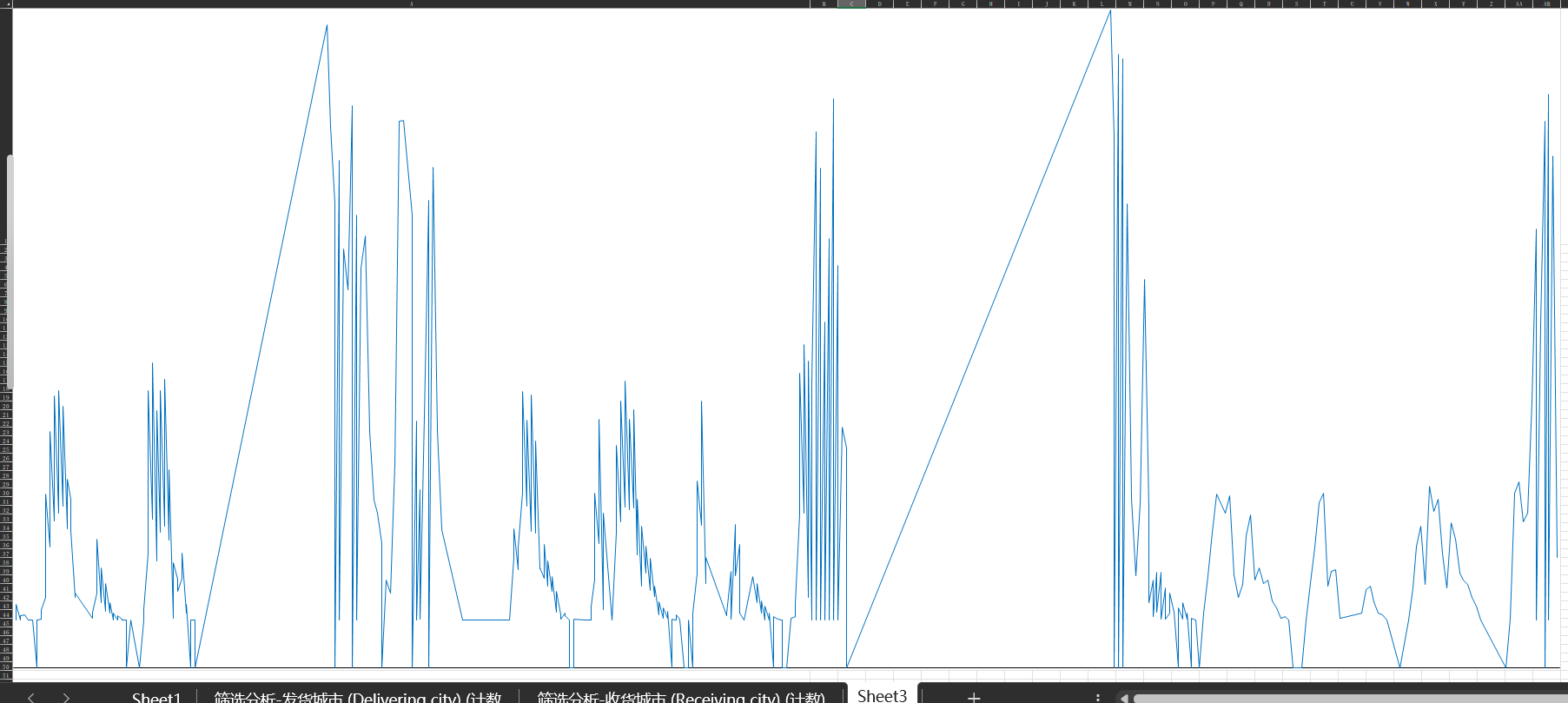
（3.1.3.1）

通过对发货城市的饼图示例直接分析，我们可以发现，在发货城市中存在这些城市L、G、V、R、Q、在我们需要预测的范围中,通过定性分析在发货量的基础上，我们可以很直观的得到一些我们所需要的数据，这些数据作为我们分析预测的重点城市。



（3.1.3.2）

通过对数据的预处理建立发货收货城市的快递运输数量并建立针对该属性的图示(3.1.3.2)可以得到重要程度排名前5的站点城市名称L,G,V,Q,R，这里为考虑比值问题，我们向后多加了两O、X城市作为也作为参考预测。



（3.1.3.3）

通过3.1.3.3图分析，对于发货数量通过WPS工具建立对应的折线分析图分析折线的分布和表现状况，从图中可以看出除去特别的几个日期分布以外，其余的均有一定的周期性表现，这符合我们对快递物流的季节性等因素的预期。

3.1.4

A城市分析 通过对A城市2018年4月到2019年四月的聚类分析可知，订单量在4-10月份中绿蔟较多，此时订单量大，而在11月 之后则为较少，分析原因可能在于11.11和12.12购物节等原因，用户需求已经得到满足，购物欲望减少导致；

黄蔟也 在4-10月份偏多，具有一定的实际参考价值，此时用户的需求量为中等；

紫色蔟为订单较为少，PCS日订单在50左右 偏多，从这里可以看出每月（剔除特殊月份）均有订单产生，分布较为平均，其中在7-9月份偏多。

同时我们从蔟的分布可以直观的发现三色蔟集中分布在前半段，即4-9月内快递物流数量始终在各个阶段均有且较为密集，在之后出现了大量的减少，以此更加验证了模型剔除的特殊场景的假设。

图表, 散点图

描述已自动生成

（3.1.4.1）

通过建立预测模型可以看出预测值如果大于0，是可以发货的，对于预测值小于0的则不以发货 ，且中间还有月份的剔除。

3.1.5相关评价指标计算

通过一个城市最大收获量表示这个城市在所有日期中，收取快递数量的最大值，针对城市j,设出这样的定义，定义他的最大收获量。

M

即固定城市j,将它的最大收获总量表示出来，求出城市日最大收获量。我们模型加入的参考值中，当一个城市最大收获量越大说明，这个城市在物流物品的增长中，可以接收更多的货物，也说明了这个城市在物流网格中具有更重要的地位。将这一值作为一种参考指标。

M

同理，在这里我们用这个指标作为日最小收取快递量，指标越小说明在城市运输当中一直保持一个比较活跃的状态，说明在城市物流网格中具有重要作用。

MAverag=Average

一个城市的平均收货量表示这个城市在所有日期中，收取快递数量的平均值。对特定的城市j，根据日期求取它的平均值，这个平均值作为它的平均收获量。一个城市的平均收获量越大说明在这个城市物流水平网格中，一直处于接收大量货物的状态，说明这个城市在网格中具有更重要的地位。

物流快递增长趋势相关指标:

Lma=max

在城市中，我们选择城市j，根据相邻两天收货数量的差值进行日收获的变化量，将相邻两天的收货量做差，之后针对所有天数，求取最大值，得到日变化的最大量，该数值越大说明城市的物流收货并不稳定，存在其他因素等干扰，可以认为重要程度并不重要。

Lma=max

同样，为同求取指标值，选择相邻两天售货量做差，针对所有天数求得日发货最大变化量，在这里同样若变化量越大说明在城市物流网格中越不稳定，设为城市重要度并不怎么高。

LAverag=Average

定义相邻两天，快递收货数量的差值为日收获变化量，根据相邻两天差值求得日收获均变化量，这里采用日收获量小作为稳定指标。

L

定义相邻两天，快递发货数量的差值为日发获变化量，根据相邻两天差值求得日收获均变化量，依然采用日收获量小作为稳定指标。

3.1.6模型验证

通过对问题一的考虑，我们采用python建立相关的模型进行对图论假设分析，采用对相关数据的分析，并根据实际情况和考虑的相关模型采用一定的权重比例，在计算当中通过对数据处理和可视化分析，对缺失值和异常值进行预处理，我们对总运输量进行相加的统计分析，根据不同的比重进行计算分析，最终得到验证值的答案符合我们在图论预测中的结果范围。

3.2问题二分析

3.2.1数据源：副本附件2(Attachment 2)2023-51MCM-Problem B

3.2.2考虑到整个物流网络是一个整体，我们对整个网络进行预测。在不考虑中转和复杂地理环境等其它因素的情况下，我们可以针对第t天构造一个矩阵，

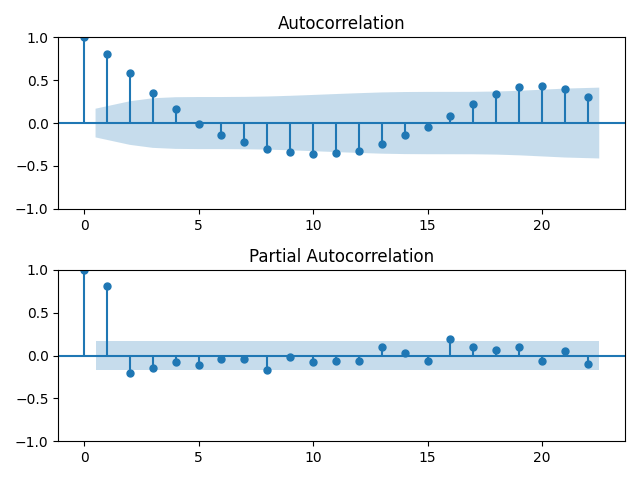
=

在这里通过将该天所有的路径的相关数据放在矩阵里（即每个元素表示第t天从i城市到j城市的快递运输量）。接着，我们将整个矩阵输入到预测模型中进行预测。因为这是一个多输入多输出的预测模型，传统的ARIMA方法可能不适用。相比之下，我们使用了神经网络来进行预测，通过训练达到最终的预测效果。依靠历史数据进行预测。

3.2.3同时我们也采用了相应的ARIMA差分预测模型[1]采用置信结果通过更改相应的自回归系数,差分阶数和移动平均项数的值，以尽量达到我们所需要的预期，并对我们模型的验证给出相应的参考。

先针对对ARIMA模型[2]相关预测获取范围值。

实例Y-M城市对中我们通过ACF和PACF图分析，其中ACF在参考标5之前正相关之后呈现负相关，在15之后得到改变，同时也更加趋近于蓝色区域内；在PACF中呈现了拖尾，在预测中从而确定了每个城市对的p,d,q的值，之后根据值根据实际参考指标进行验证改正p,d,q的值，以此达到更高的预测值。



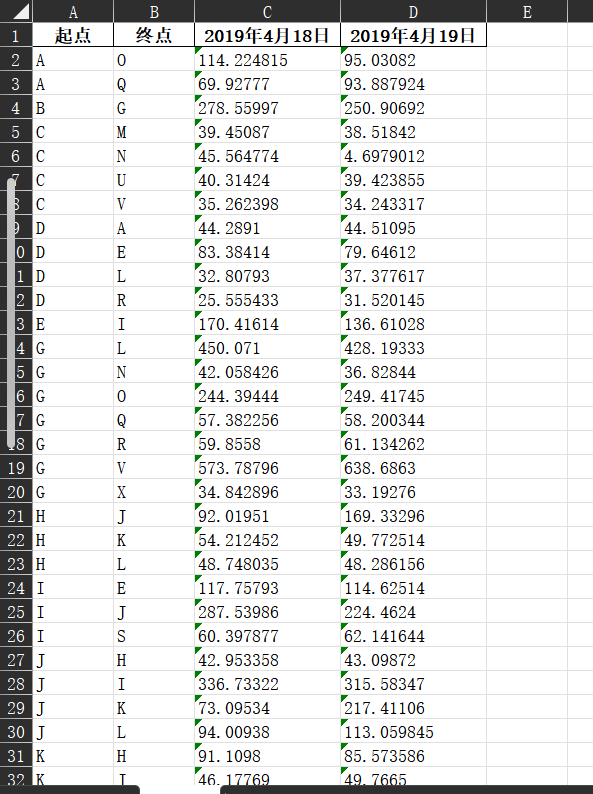
3.2.3.1

3.2.4在LSTM模型训练[3]中，我们采用大量的数据，先根据小数据针对精度问题进行改正，这里使用了MSE均方误差分析作为一个评估指标，之后根据每一天进行训练，预测每一天的货量，将预测结果转换原始尺度通过预测函数进行对应的模型预测，之后通过数据划分为训练集和测试集，对数据进行归一化操作。

在LSTM模型训练中中通过将原始数据按照起点和终点线路以及日期进行分类，并将是否开通和预测值存储在数组d和d1中。创建了一个循环，对每个起点和终点线路进行时间序列预测。

用训练数据集和测试数据集对模型进行预测，并将预测结果逆归一化，恢复为原始尺度。最后一部分开始预测未来一段时间的值。在每次预测中，我们将输入数据重塑为符合LSTM模型输入格式的形状，并用模型进行预测。预测结果存储在predictions数组中。

在LSTM模型中。我们使用Adam优化器进行训练，并使用均方误差作为损失函数，通过create\_dataset函数生成用于训练和测试的数据集。然后，将数据集转换为适合LSTM模型输入的三维张量形式。来预测未来一段时间的货量。



3.2.3.2

表格

描述已自动生成

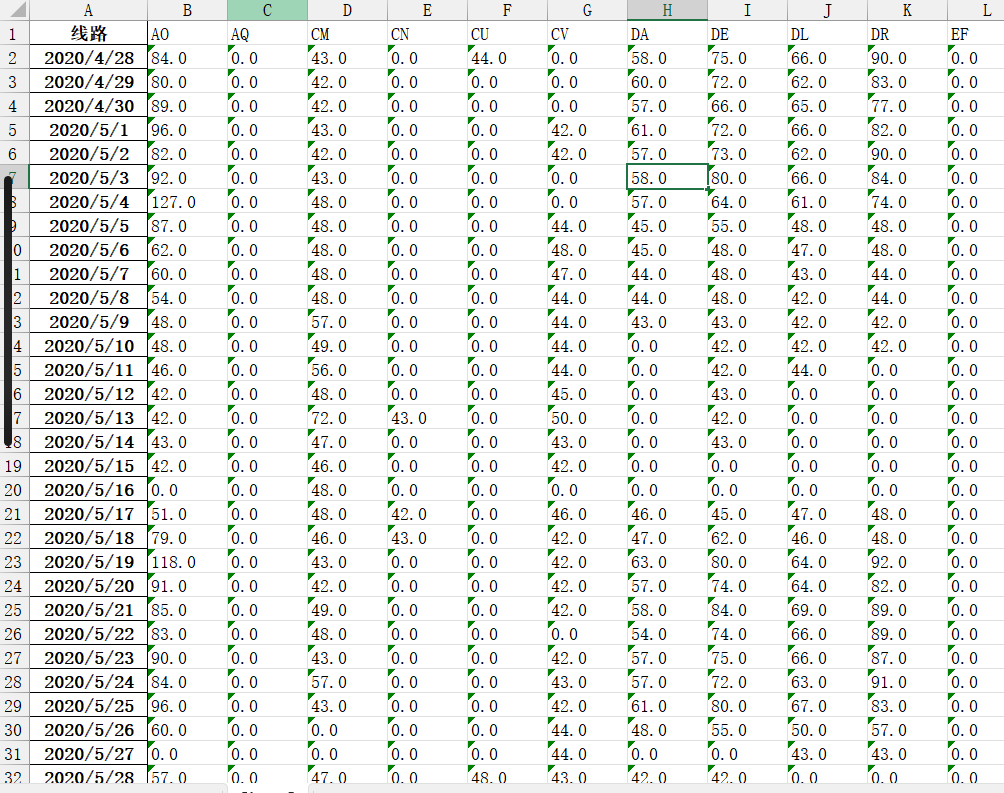
3.2.3.3

从筛选出来的数据3.2.3.2和数据3.2.3.3中可以通过对应的代码和WPS工具筛选获取对应的结果。

3.3问题三分析

3.3.1数据源：副本附件2(Attachment 2)2023-51MCM-Problem B和主要的各个城市对附件

3.3.2这里做出假设将不通的路径设定为负值，比如-100，然后代入到第二问的模型中，一起做输入输出，如果新输出的结果是负值，代表着，这条道路的预测结果不是通路，如果预测出来的新结果为正值，那么就直接代表这条道路预测出来的物流货物数量.



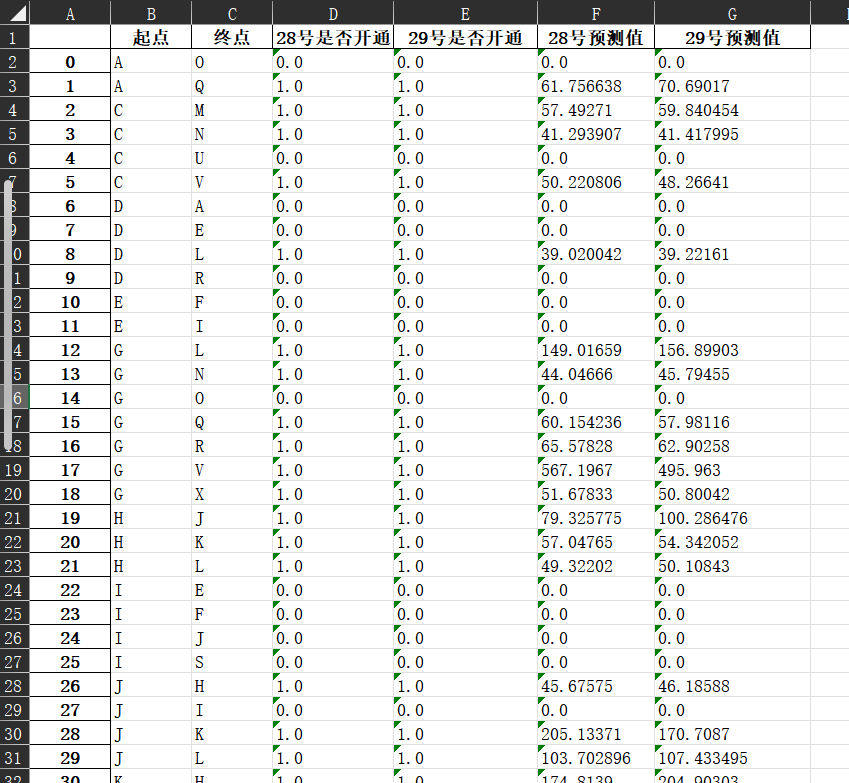
3.3.2.1

这里的时间序列数据通过第二问LSTM预测模型，将相应的获取值和其它地方进行简易修改过后，通过LSTM数据在xlsx表格中展现出来，给出直观的数据。

any ()=

采用第二问的矩阵模型，将第二问的矩阵转化为0-1矩阵，针对第T天，构造矩阵；这样我们把第t天所有的路径相关的数据放到了一个矩阵里面（矩阵中第i行第j列即第t天从i城市到第j城市的快递运输量），我们使用any(xijt),判断对于第t天第i个城市是否向第j个城市运输快递，如果运输，那么any(xijt)设为1，反之不运输我们将其值定位0，如此将运输问题通过0-1模型简单标注出来，给出直观的感受。

其次，我们通过将any（A）代入到模型二中，判断通路状态，以此根据通路状态再次利用二问模型预测出每条路的对应的结果3.3.2.2。



3.3.2.2

3.3.3在ARIMA模型预测中，我们采用第二问的模型，简单修改过后，做出这一思考，将这段时间进行一个时间预测分析，通过0-1值来标记预测值，也就是在时间模型中，针对缺失的值我们对其进行标0操作，针对xlsx表中中有的值我们进行标1操作，同时去做每个城市的0-1值，求得相应的预测模型值。

针对预测第i天的0-1值，我们采用了通过对0-1值加入条件，即当预测的值大于0.5的时候，我们将其标记为1，我们认为能发货，反之，当预测值小于0.5的时候，我们直接记为0，认为在该状态下不能发货。通过模型的预测，我们可以确定出哪些天是否能发货，这样再采用二问中的ARIMA模型预测，来预测对应的发货值。

针对ARIMA的示例给出们的相关预测状态结果。

（3.3.3.1）

3.4问题四分析

3.4.1数据源：附件3(Attachment 3)2023-51MCM-Problem B

3.4.2在数据中给定了所有铁路的固定成本、额定装货量等参数，作为参考在目标函数中，因此实际上运输过程在相邻的城市之间进行，这里我们采用三问中的矩阵。

Any

此时，根据两个城市是否相邻,判断 any’(Xij)是否等于1，如果相邻则判断等于1，否则记为0。

接着，设定决策变量，对于某一天，某条快递线路，决策变量就是当天在具体某些城市上面进行运输.比如,A 到E,可以走 ATE,也可以走 ACLNE.最多可以把 A到E 的所有快递拆成五个部分，每个部分走不同的路径，在每条相邻道路的成本计算时，所有经过这条道路的路径都要算进去。

这里设定决策变量为其中一条路线，以A到E为例，通过0-1矩阵:

Rote（<1> AE）=

以及相应的运输量对应的正整数矩阵

Sumon（<1> AE）=

这里我们运输路径选择每次一个距离，去到相邻的城市进行，即

Any(<k>) <= any

通过约束条件缩小运算所需的变量。

对于5条路线，对于中间的路过城市，我们可以推断知道，它接收的货物量必须等于发出的货物量，即

也就是矩阵Sumon<AE>的第j列所有数字的和(j城市接收量)，必须等于第j行所有数字的和(j城市的发出量)

同时，道路问题还需考虑在这个城市只进出一次，所以有

对于起点城市A来说，必须只有一条路径出发，无到达，即

对于终点城市，必须有一条路径到达，没有出发路径。

这样我们可以完成一条路径的决策变量，同时还需要以同样的形式对其它路径进行定义。

对于第k条路线，以下为0-1矩阵

Rote

以及运输量正整数矩阵

Sumon

矩阵里边的K取值为12345。

对于与运输总量有关的约束条件AE运输的快递总量为Sumon(AE)

所以对于发出的所有货物总量与收到的所有货物总量均应该是Sumon(AE),在路途中，对于所有途径城市，设置接收的货物量必须等于发出的货物量，所以总量在中间不会有损失。

对于出发点A

=Sumon(AE)

对于接收点E

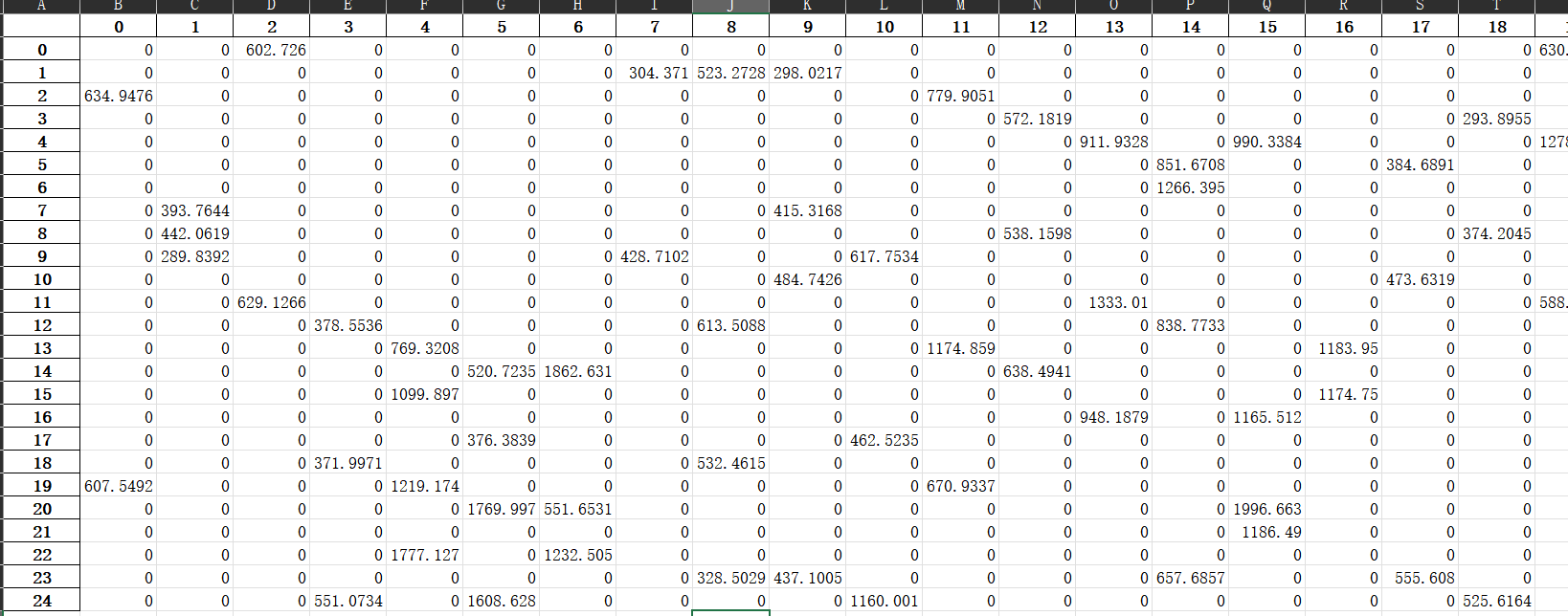
=Sumon(AE)

接着，每条相邻线路的总成本均采用题给出的公式计算，其中固定成本和额定装货量作为参数，实际装货量是所有路径的yij

那么,第i个城市到第j个城市的实际装货量为

代入到成本表达式中，之后再把所有相邻道路求和即可得出目标函数。

以下是由lingo计算出的相关数据参考



3.4.2.1

3.5问题五分析

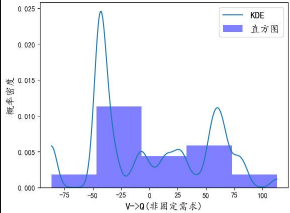
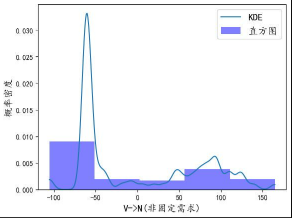
3.5.1分析题为概率论问题，求分布的均值和方差。首先对数据进行处理，对对应季度对应站点的数据按进行整理和收集，并进行直方图统计，使用相邻按付应天数进行做差处理，用于去除时间波动的影响，在对无时间波动影响的数据计算均值记为固定需求。将数据进行平均处理，减去固定需求。得到非固定需求的分布函数，根据分布函数求得均值和方差。

3.5.2根据题意我们可以知道这些问题都是对特定的时间段进行分析，所以我们只分析了2022年第三季度（7-9月）和2023年第一季度（1-3月）的数据并按照季度划分，计算每个季度和每个“发货，收货”站点城市对的需求量平均值。为了找出需求的固定常数。

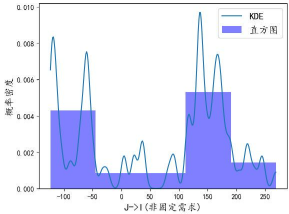
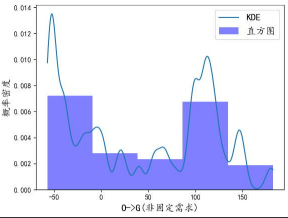
在这里我们采用统计学中的3sigma原则，我们认为正态分布数据中绝大多数的样本都分布在平均值的附近，并将正常的物流量值定义为大于平均值减去2倍标准差。因此，我们就要从每个城市对的平均需求值中减去需求值的2倍标准差以得到固定需求常数。然后，我们可以获得excel表格来呈现这些固定需求部分的数据。



这样我们通过将源数据减去固定部分的数据来获得非固定需求数据，并采用非参数估计方法进行估计。最后，我们可以生成相应的图表以便于进一步分析和预测物流的需求。



3.5.2.1 3.5.2.2



3.5.2.3 3.5.2.4

在这里，我们首先采用了非参数估计方法，这种方法没有一个参数或者显式的表达式，但是我们可以得到每个点的概率分布。因此，我们可以使用对应的Excel文件来进行计算。最后，我们按照以下步骤计算了指定季度和“发货-收货”站点城市对的非固定需求均值和标准差。首先，我们将数据按季度进行分组。然后，对于每个季度，我们计算每个“发货-收货”站点城市对的固定需求常数，这可以通过平均需求值减去整个季度的最小需求值来实现。最后，对于每个“发货-收货”站点城市对，我们从原始需求中减去对应的固定需求常数，得到非固定需求数据。对于每个“发货-收货”站点城市对的非固定需求数据，我们计算了均值和标准差。

非固定需求均值(u)= (非固定需求) /n

非固定需求标准差(o)=根号下[(非固定需求-u)/(n-1)]其中n是每个城市对的需求样本数量。

最终结果是我们得到了所有路径的表格，分别是1季度和3季度。

表格

中度可信度描述已自动生成 表格, Excel

描述已自动生成

3.5.2.5 3.5.2.6

从数据结果3.5.2.5和数据结果3.5.2.6可易得到相对应的答案，据此我们可以通过模型推断出。

# 模型的优缺点

模型优点：对于问题二，我们将整个物流网络作为一个整体，依靠本身的数据进行预测，并构造数学矩阵。我们采用了LSTM和ARIMA模型来进行预测，充分利用它们各自的优势。首先，LSTM模型能够通过历史数据进行训练，捕捉时间序列中的长期依赖关系，以提高预测精度和准确性。我们使用MSE均方误差作为衡量标准，对每天的货量进行预测，并将预测结果转换为原始尺度，通过训练集和测试集的数据划分和归一化操作进行模型训练。其次，ARIMA模型提供了对趋势和季节性因素的清晰解释，通过调整自回归系数、差分阶数和移动平均项数的值来优化预测效果，并提供置信区间的结果作为参考，以提高预测精度。我们采用Adam优化算法来加速LSTM模型的训练速度，并将数据集转换为适合LSTM模型输入的三维张量形式，以预测未来一段时间的货量。综合使用LSTM和ARIMA模型，我们能够进行全面、科学、严谨的问题分析，并在特定场景下具有一定的推广应用。

模型缺点：然而，LSTM模型相对于其他模型需要更多的参数量，这意味着需要更多的计算资源和更长的训练时间。在训练数据较小或模型复杂度较高时，神经网络模型容易出现过拟合现象，可能导致在新数据上表现不佳。此外，该模型没有考虑到物流中转和复杂的地理环境，限制了其在整个物流网络中的适用性。

模型改进：为了改进模型，我们可以采取以下措施。首先，减少计算资源和训练时间的需求，可以尝试使用更高效的模型优化算法，例如批量归一化和替代优化器，以加快模型训练速度。其次，为了考虑到物流中转和复杂的地理环境，可以引入更多的特征，如交通拥堵情况和物流节点之间的关系，以更准确地反映实际物流运作情况。最后，为了综合利用LSTM模型和ARIMA模型的优势，可以采用ARIMA-LSTM混合模型。该模型首先使用ARIMA模型进行初步预测和分析，然而将ARIMA的预测结果作为输入，通过LSTM模型进行进一步的精细预测。这种混合模型可以充分利用ARIMA模型对趋势和季节性因素的解释能力，同时结合LSTM模型对长期依赖关系的建模能力，以提高预测的准确性和稳定性。

# 参考文献

[1] 姜启源，《数学模型》，北京：高等教育出版社，2018.5。225-230

[2] 随风，时间序列分析ARIMA模型，[时间序列分析（2） ARIMA 模型 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/60648709)，2023-06-01。

[3]coloah, Understanding LSTM Networks, [Understanding LSTM Networks -- colah's blog](http://colah.github.io/posts/2015-08-Understanding-LSTMs/),2023-06-01

# 附录

附录1 部分相关代码

蔟类分析

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

# 读取Excel文件并保存为CSV文件

df = pd.read\_excel('A.xlsx')

df.to\_csv('dataA.csv', index=False)

# 读取数据集并提取需要聚类的第三列和第四列数据

data = pd.read\_csv('dataA.csv')

X = data.iloc[:, 2:4].values

# 将日期转换为时间戳，并将时间戳除以一个月的秒数得到月份表示

for i in range(len(X)):

X[i, 1] = int((pd.to\_datetime(X[i, 1]) - pd.to\_datetime('1970-01-01')).total\_seconds() / (30 \* 24 \* 3600))

# 对数据进行聚类分析

kmeans = KMeans(n\_clusters=3)

kmeans.fit(X)

labels = kmeans.labels\_

# 可视化聚类结果

plt.scatter(X[:, 1], X[:, 0], c=labels)

plt.xlabel('Month(588=18-4|originate)')

plt.ylabel('PCS')

plt.title('A-KMeans Clustering')

plt.show()

# 1

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_excel("附件1(Attachment 1)2023-51MCM-Problem B.xlsx")

missing\_values = data.isnull().sum()

print("缺失值数量：\n", missing\_values)

data = data.dropna()

print("数据类型：\n", data.dtypes)

data['日期(年/月/日) (Date Y/M/D)'] = pd.to\_datetime(data['日期(年/月/日) (Date Y/M/D)'])

plt.boxplot(data['快递运输数量(件) (Express delivery quantity (PCS))'])

plt.show()

Q1 = data['快递运输数量(件) (Express delivery quantity (PCS))'].quantile(0.25)

Q3 = data['快递运输数量(件) (Express delivery quantity (PCS))'].quantile(0.75)

IQR = Q3 - Q1

data = data[~((data['快递运输数量(件) (Express delivery quantity (PCS))'] < (Q1 - 1.5 \* IQR)) | (data['快递运输数量(件) (Express delivery quantity (PCS))'] > (Q3 + 1.5 \* IQR)))]

city\_send = data.groupby('发货城市 (Delivering city)').sum(numeric\_only=True).reset\_index()

city\_receive = data.groupby('收货城市 (Receiving city)').sum(numeric\_only=True).reset\_index()

city\_stat = city\_send.merge(city\_receive, left\_on='发货城市 (Delivering city)', right\_on='收货城市 (Receiving city)',

how='outer').fillna(0)

city\_stat['总运输量'] = city\_stat['快递运输数量(件) (Express delivery quantity (PCS))\_x'] + city\_stat[

'快递运输数量(件) (Express delivery quantity (PCS))\_y']

city\_stat = city\_stat.sort\_values(by='总运输量', ascending=False)

top\_cities = city\_stat['发货城市 (Delivering city)'].head(5).tolist()

print("重要程度排名前5的站点城市名称：", top\_cities)

# 2ARIMA模型预测

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA

from statsmodels.tsa.stattools import adfuller

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

import itertools

import warnings

# 读取CSV数据

df = pd.read\_excel('2.xlsx')

df.to\_csv('data2.csv', index=False)

data = pd.read\_csv('data2.csv')

# 检查缺失值

missing\_values = data.isnull().sum()

print("缺失值数量：\n", missing\_values)

# 如果有缺失值，可以选择删除缺失值所在的行

data = data.dropna()

# 关闭警告提示

warnings.filterwarnings("ignore")

# 检查数据类型

print("数据类型：\n", data.dtypes)

# 数据预处理：以日期为索引，发货-收货城市对为列名，快递运输数量为值

data\_pivot = data.pivot\_table(index='Date Y/M/D', columns='Delivering city-Receiving city', values='Express delivery quantity (PCS)')

data\_pivot.index = pd.to\_datetime(data\_pivot.index)

# 选择一个城市对进行分析

city\_pair = 'V-G'

ts = data\_pivot[city\_pair].dropna()

# 判断时间序列是否平稳，使用ADF检验

def test\_stationarity(timeseries):

adftest = adfuller(timeseries, autolag='AIC')

return adftest[1] # 返回p-value

# 若时间序列非平稳，进行差分操作

d = 0

while test\_stationarity(ts) > 0.05:

ts = ts.diff().dropna()

d += 1

# 计算ACF和PACF，以确定p和q

fig, ax = plt.subplots(2, 1)

plot\_acf(ts, ax=ax[0])

plot\_pacf(ts, ax=ax[1])

plt.show()

# 确定参数搜索范围

p\_range = range(0, 3) # 可以根据PACF图调整

q\_range = range(0, 3) # 可以根据ACF图调整

pdq = list(itertools.product(p\_range, [d], q\_range))

# 网格搜索寻找最佳参数

best\_aic = np.inf

best\_param = None

for param in pdq:

try:

model = ARIMA(ts, order=param)

results = model.fit()

if results.aic < best\_aic:

best\_aic = results.aic

best\_param = param

except:

continue

print(f'Best parameters (p, d, q): {best\_param}')

# 使用最佳参数建立ARIMA模型并进行预测

best\_model = ARIMA(ts, order=best\_param)

results = best\_model.fit()

# 预测2019年4月18日和2019年4月19日的快递运输数量

forecast = results.get\_forecast(steps=2) # 预测两天的数据

# 获取预测结果和置信区间

forecast\_mean = forecast.predicted\_mean

forecast\_conf\_int = forecast.conf\_int()

# 输出预测结果和置信区间

print("预测运输数量:")

print(forecast\_mean)

print("置信区间:")

print(forecast\_conf\_int)

# 计算总运输数量

total\_quantity = forecast\_mean.sum()

# 打印总运输数量

print("总运输数量:", total\_quantity)

# 绘制预测结果

plt.figure(figsize=(10, 6))

plt.plot(ts.index, ts.values, label='Actual')

plt.plot(forecast\_mean.index, forecast\_mean.values, color='red', linestyle='--', label='Forecast')

plt.fill\_between(forecast\_conf\_int.index, forecast\_conf\_int.iloc[:, 0], forecast\_conf\_int.iloc[:, 1], color='gray', alpha=0.3, label='Confidence Interval')

plt.title(f"ARIMA {best\_param} - {city\_pair}")

plt.xlabel('Date')

plt.ylabel('Express delivery quantity (PCS)')

plt.legend()

plt.show()

# 3LSTM

# In[1]:  
  
import numpy as np  
import pandas as pd  
from sklearn.preprocessing import MinMaxScaler  
from keras.models import Sequential  
from keras.layers import Dense, LSTM  
import datetime  
  
# In[7]:  
  
data = pd.read\_excel(r"附件2(Attachment 2)2023-51MCM-Problem B.xlsx").values  
  
# In[14]:  
  
lines = np.unique(np.array([i[0]+i[1]for i in data[:,[1,2]]]))  
lines = np.array([[i[0],i[1]] for i in lines])  
date = np.unique(data[:,0])  
  
# In[15]:  
  
d = np.zeros(shape=(lines.shape[0],date.shape[0]))  
d1 = np.zeros(shape=(lines.shape[0],date.shape[0]))  
for i in range(d.shape[0]):  
 data1 = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])]  
 for j in range(d.shape[1]):  
 data2 = data1[data1[:,0]==date[j]]  
 if data2.shape[0]==0:  
 d[i,j]=0  
 d1[i,j]=0  
 else:  
 d[i,j]=1  
 d1[i,j]=data2[0,-1]  
  
  
# In[20]:  
  
  
all\_pre1 = []  
def create\_dataset(data, look\_back=1):  
 X, Y = [], []  
 for i in range(len(data) - look\_back):  
 X.append(data[i:(i + look\_back), 0])  
 Y.append(data[i + look\_back, 0])  
 return np.array(X), np.array(Y)  
for i in range(lines.shape[0]):  
 y = d[i]  
 # 将数据划分为训练集和测试集  
 train\_size = int(len(y) \* 0.8)  
 train, test = y[:train\_size], y[train\_size:]  
  
 # 数据归一化  
 scaler = MinMaxScaler()  
 train = scaler.fit\_transform(train.reshape(-1, 1))  
 test = scaler.transform(test.reshape(-1, 1))  
  
 # 创建数据生成器  
  
 look\_back = 7  
 X\_train, y\_train = create\_dataset(train, look\_back)  
 X\_test, y\_test = create\_dataset(test, look\_back)  
  
 # 重塑数据以适应LSTM模型的输入格式  
 X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))  
 X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))  
  
 # 创建LSTM模型  
 model = Sequential()  
 model.add(LSTM(50, input\_shape=(look\_back, 1)))  
 model.add(Dense(1))  
 model.compile(loss='mean\_squared\_error', optimizer='adam')  
  
 # 训练模型  
 model.fit(X\_train, y\_train, epochs=20, batch\_size=1, verbose=1)  
  
 # 预测  
 train\_predict = model.predict(X\_train)  
 test\_predict = model.predict(X\_test)  
  
 # 将预测结果转换回原始尺度  
 train\_predict = scaler.inverse\_transform(train\_predict)  
 y\_train = scaler.inverse\_transform(y\_train.reshape(-1, 1))  
 test\_predict = scaler.inverse\_transform(test\_predict)  
 y\_test = scaler.inverse\_transform(y\_test.reshape(-1, 1))  
  
 # 计算预测准确性，例如使用均方误差（MSE）作为评估指标  
 train\_mse = np.mean((train\_predict - y\_train) \*\* 2)  
 test\_mse = np.mean((test\_predict - y\_test) \*\* 2)  
  
 # 计算预测日期与最后一个训练日期之间的天数  
 last\_train\_date = datetime.date(2019, 4, 17)  
 start\_pred\_date = datetime.date(2019, 4, 18)  
 end\_pred\_date = datetime.date(2019, 4, 20)  
 days\_to\_predict = (end\_pred\_date-start\_pred\_date).days  
  
 # 使用训练数据的最后一部分来开始预测  
 input\_data = train[-look\_back:]  
  
 predictions = []  
  
 # 预测每一天的货量  
 for i in range(days\_to\_predict):  
 input\_data\_reshaped = input\_data.reshape(1, look\_back, 1)  
 pred = model.predict(input\_data\_reshaped)  
 predictions.append(pred[0, 0])  
  
 # 更新输入数据，用预测值替换最早的值  
 input\_data = np.roll(input\_data, -1)  
 input\_data[-1] = pred  
  
 # 将预测值转换回原始尺度  
 predictions = scaler.inverse\_transform(np.array(predictions).reshape(-1, 1))  
  
 tdays = []  
 # 打印预测结果  
 for i, pred in enumerate(predictions, start=1):  
 pred\_date = start\_pred\_date + datetime.timedelta(days=i - 1)  
 tdays += [pred[0]]  
 all\_pre1+=[tdays]  
all\_pre1 = np.array(all\_pre1)  
  
# In[21]:  
  
all\_pre = []  
def create\_dataset(data, look\_back=1):  
 X, Y = [], []  
 for i in range(len(data) - look\_back):  
 X.append(data[i:(i + look\_back), 0])  
 Y.append(data[i + look\_back, 0])  
 return np.array(X), np.array(Y)  
for i in range(lines.shape[0]):  
 y = d1[i]  
 y[y==0]=y.mean()  
 # 将数据划分为训练集和测试集  
 train\_size = int(len(y) \* 0.8)  
 train, test = y[:train\_size], y[train\_size:]  
  
 # 数据归一化  
 scaler = MinMaxScaler()  
 train = scaler.fit\_transform(train.reshape(-1, 1))  
 test = scaler.transform(test.reshape(-1, 1))  
  
 # 创建数据生成器  
  
 look\_back = 7  
 X\_train, y\_train = create\_dataset(train, look\_back)  
 X\_test, y\_test = create\_dataset(test, look\_back)  
  
 # 重塑数据以适应LSTM模型的输入格式  
 X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))  
 X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))  
  
 # 创建LSTM模型  
 model = Sequential()  
 model.add(LSTM(50, input\_shape=(look\_back, 1)))  
 model.add(Dense(1))  
 model.compile(loss='mean\_squared\_error', optimizer='adam')  
  
 # 训练模型  
 model.fit(X\_train, y\_train, epochs=20, batch\_size=1, verbose=1)  
  
 # 预测  
 train\_predict = model.predict(X\_train)  
 test\_predict = model.predict(X\_test)  
  
 # 将预测结果转换回原始尺度  
 train\_predict = scaler.inverse\_transform(train\_predict)  
 y\_train = scaler.inverse\_transform(y\_train.reshape(-1, 1))  
 test\_predict = scaler.inverse\_transform(test\_predict)  
 y\_test = scaler.inverse\_transform(y\_test.reshape(-1, 1))  
  
 # 计算预测准确性，例如使用均方误差（MSE）作为评估指标  
 train\_mse = np.mean((train\_predict - y\_train) \*\* 2)  
 test\_mse = np.mean((test\_predict - y\_test) \*\* 2)  
  
 # 计算预测日期与最后一个训练日期之间的天数  
 last\_train\_date = datetime.date(2019, 4, 17)  
 start\_pred\_date = datetime.date(2019, 4, 18)  
 end\_pred\_date = datetime.date(2019, 4, 20)  
 days\_to\_predict = (end\_pred\_date-start\_pred\_date).days  
  
 # 使用训练数据的最后一部分来开始预测  
 input\_data = train[-look\_back:]  
  
 predictions = []  
  
 # 预测每一天的货量  
 for i in range(days\_to\_predict):  
 input\_data\_reshaped = input\_data.reshape(1, look\_back, 1)  
 pred = model.predict(input\_data\_reshaped)  
 predictions.append(pred[0, 0])  
  
 # 更新输入数据，用预测值替换最早的值  
 input\_data = np.roll(input\_data, -1)  
 input\_data[-1] = pred  
  
 # 将预测值转换回原始尺度  
 predictions = scaler.inverse\_transform(np.array(predictions).reshape(-1, 1))  
  
 tdays = []  
 # 打印预测结果  
 for i, pred in enumerate(predictions, start=1):  
 pred\_date = start\_pred\_date + datetime.timedelta(days=i - 1)  
 tdays += [pred[0]]  
 all\_pre+=[tdays]  
all\_pre = np.array(all\_pre)  
  
# In[23]:  
  
all\_pre1[all\_pre1>0.5]=1  
  
# In[25]:  
  
all\_pre1[all\_pre1<=0.5]=0  
  
# In[30]:  
  
p2 = all\_pre\*all\_pre1  
  
# In[36]:  
  
#pd.DataFrame(np.c\_["1",lines,all\_pre1,p2],columns=["起点","终点","28号是否开通","29号是否开通","28号预测值","29号预测值"]).to\_excel(r"C:\Users\MATH\_MGD\Desktop\第三问结果.xlsx")  
  
# In[ ]:

lines = np.unique([i[0]+i[1]for i in data[:,[1,2]]])  
lines = np.array([[i[0],i[1]] for i in lines])  
  
line\_mean = []  
line\_min = []  
xu = []  
f\_mean = []  
f\_std = []  
mt = np.zeros(shape=(lines.shape[0],date.shape[0]))  
for i in range(lines.shape[0]):  
 d1 = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,-1]  
 rq = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,0]  
 line\_mean+=[d1.mean()]  
 line\_min +=[d1.min()]  
 gd =d1.mean()-2\*d1.std()  
 gd=0 if gd<0 else gd  
 xu +=[gd]  
 d2 = d1-(gd)  
 f\_mean += [d2.mean()]  
 f\_std += [d2.std()]  
 for j in range(rq.shape[0]):  
 mt[i,date==rq[j]] = d2[j]  
xu = np.array(xu)  
print("23年1季度固定需求",np.c\_["1",lines,np.round(xu)])  
print("23年1季度非固定需求均值标准差\n",np.c\_["1",lines,np.round(f\_mean,4),np.round(f\_std,4)])  
  
# In[76]:  
  
pd.DataFrame(np.c\_["1",lines,np.round(xu)],columns=["起点","终点","固定需求"]).to\_excel(r"C:\Users\MATH\_MGD\Desktop\第五问\_固定需求(23年第1季度).xlsx")  
  
# In[77]:  
  
pd.DataFrame(np.c\_["1",lines,np.round(f\_mean,4),np.round(f\_std,4)],columns=["起点","终点","均值","标准差"]).to\_excel(r"C:\Users\MATH\_MGD\Desktop\第五问\_非固定需求(均值与标准差)(23年1季度).xlsx")  
  
# In[78]:  
  
pd.DataFrame(np.c\_["1",lines,mt],columns=np.r\_["0",["起点","终点"],date]).to\_excel(r"C:\Users\MATH\_MGD\Desktop\第五问\_非固定需求(23年1季度).xlsx")  
  
# In[79]:  
  
for i in range(2,4):  
 d1 = data[np.logical\_and(data[:,1]==sc[i,0],data[:,2]==sc[i,1])][:,-1]  
 d2 = d1-(d1.mean()-d1.min())  
 # 示例数据（请用实际非固定需求数据替换）  
 sample\_data = d2.reshape(-1, 1)  
  
 # KDE模型实例化  
 kde = KernelDensity(kernel='gaussian', bandwidth=5).fit(sample\_data)  
  
 # 指定评估点（根据实际需求调整范围和间隔）  
 evaluation\_points = np.linspace(d2.min(), d2.max(), num=300).reshape(-1, 1)  
  
 # 评估KDE模型  
 log\_density = kde.score\_samples(evaluation\_points)  
 density = np.exp(log\_density)  
  
 # 绘制KDE结果和直方图  
 fig, ax = plt.subplots()  
 ax.plot(evaluation\_points, density, label='KDE')  
 ax.hist(sample\_data, bins=5, density=True, alpha=0.5, color='blue', label='直方图')  
 ax.set\_xlabel(f'{sc[i,0]}->{sc[i,1]}(非固定需求)',fontsize=14)  
 ax.set\_ylabel('概率密度',fontsize=14)  
 ax.legend(fontsize=14)  
 plt.show()  
 pd.DataFrame(np.c\_["1",evaluation\_points,density[:,None]],columns=["非固定需求量","概率密度"]).to\_excel(r"C:\Users\MATH\_MGD\Desktop\{}\_{}(非固定需求).xlsx".format(sc[i,0],sc[i,1]))  
# In[ ]:  
# In[ ]:  
  
# In[ ]: