2020 年河北省第三届研究生数学建模竞赛

题 目：交通检测器数据质量控制及预测

摘 要

本文针对交通检测器采集原始数据的数据质量问题，运用数据清洗法和多元回归线性拟合法建立了数学模型，分别对三种不同参数（速度，时间占有率，流量）的建模，观察拟合程度，运用 Python 软件编程和 Matlab 编程得出合理的结论，实现对交通数据质量控制和预测，并且最终对模型的结果做出了误差分析。

针对问题一要求根据给定数据（data1）建立识别问题数据的方法和流程，完成对问题数据的修正及缺失数据的填补。我们先设计了识别问题数据的方法，并转化为流程图展示出来。通过遍历数据、检查数据的完整性、方差检验和阈值检验清洗异常数据。

针对问题二要求在时间维度上对交通流三参数使用曲线拟合方法建立多元回归模型。利用交通检测器数据（data2）进行数据清洗并重构数据集，这里我们取每一天固定检测时间的均值，并以其序列号为时间索引，分别形成了二维的关系图。最后使用 Gauss 6 函数建立多元线性回归模型，以获取时间维度和交通流参数的关系。又基于交通流三参数间的关系使用曲面拟合方法建立模型，首先清洗数据并重构数据集，然后初始化参数并设置参数限制，随后使用 Polynomial Features 进行特征构造，将特征转换后进行线性回归构建模型并开始训练，得到交通流参数间的关系。最后将 Data2 中缺失的数据补全。

针对问题三要求对给定数据（data3）建立预测模型并预测出缺失的数据。利用已有 data3 中的数据建立 ARIMA 预测模型确定时间维度与交通流参数的组合表达式，确定 ARIMA 模型后对模型结果进行评价，然后进行拟合预测，最后缺失值补充。

本文主要应用 Python 语言对相关的模型进行编程求解，计算方便、快捷、准确，整篇文章图文并茂。文章最后根据所建立的拟合模型进行了误差分析，结果可靠，使得模型具有现实意义。

关键词

数据清洗；Pandas；曲线拟合；Matlab；ARIMA 预测模型；SPSS。

1 问题重述

1.1 背景知识

1. 城市道路中大量应用交通检测器用于采集断面交通数据[1],采集得到的数据质量存在着诸如缺失、异常等诸多问题。如果直接使用包含质量问题的交通数据展开交通应用。将会给应用结果带来不稳定和不安全隐患。

2.交通流数据的三要素（流量、速度和时间占有率）对于交通的演变规律有着重要影响[2]，通过研究交通量的变化规律以及三个参数之间的关系，对未来的交通的发展趋势和预测起了重要作用，同时也是改善现有交通道路设施的关键前提。

3.交通需求预测在交通规划领域中非常重要,交通需求预测的四阶段法更是城市地区规划的基石[3]，更是交通规划领域符合未来发展的条件，如何利用不同场景所给出的交通数据来进行研究预测和建立模型，这对于实测交通流数据的模型检验结果，总结出实用的数据质量控制和预测经验，为交通管理部门的数据质量改善的实践工作提供指导提供很大的帮助。

1.2 问题概括

问题一：对于给定数据(data 1)，需要建立识别问题数据（包括异常数据、重复记录等）的方法和流程，并完成对问题数据的修正及缺失数据的填补。

问题二：首先需要在时间维度上对交通流参数（流量、速度和时间占有率）进行建模，用以刻画交通系统在时间上的演化过程。其次，需要对这三个参数之间的关系进行建模。对于给定的数据(data 2),将缺失的数据(NA)补全,即用数值代替 NA。

问题三：要求对给定的数据（data 3）,建立模型并对于给定的数据(data 3),将缺失的数据(NA)补全,即用数值代替 NA。

2 问题分析

本题是基于交通流参数（占有率、流量、速度）来建立交通检测器数据质量控制和预测的数学模型，本题提出了采集到的数据会有数据缺失的情况,首先依据数据缺失的处理方法（数据删除、数据插补）对数据进行第一步的处理，对异常数据进行数据清洗后再进行阈值检验，提高数据集的质量。其次，根据时间维度上建模的方法，得到它们在时间尺度的演化过程，再根据三者之间的关系建立三者关系公式的模型，补全数据。然后根据之前的数据清洗与时间建模，关系建模，从而建立预测的数学模型来实现对交通需求的预测。

对于**问题一**，要对所给出的数据利用 python 进行数据清洗，将给定的 data1 进行遍历，检查数据完整性，通过前后数据补全，缺的数据多的话，考虑平滑处理，去除重复记录，对三参数进行方差检验，设置阈值检验，确保数据合法性，以此得到质量提高的数据集。

对于**问题二**，交通系统已经在时间周期上表现出依赖性等显著特征，将 data2 数据清洗，重构数据集，对三个参数分别在时间尺度上利用拟合进行建模，得出模型 1，推导三参数的关系公式，建立三维坐标从而建立模型 2。

对于**问题三**，需要对已知数据建立预测模型，首先处理已知数据得出其相关性，建立预测模型后可对模型进行分析，输入相关时间数据做出预测，最后补全缺失数据。

3 模型假设

1.所有数据均为原始数据，来源真实可靠。
2.数据类型完整,同一字段是同一类型数据。
3.假设路面车辆类型一致，车辆长度和检测器长度固定。
4.除了交通流三参数，其它参数不参与研究。

4 名词解释和符号说明

4.1 名词解释

最小二乘法：是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配[4].

线性拟合：是曲线拟合的一种形式。设 x 和 y 都是被观测的量，且 y 是 x 的函数： $y = f(x;b)$。

曲线拟合：通过 x,y 的观测值来寻求参数 b 的最佳估计值，及寻求最佳的理论曲线 $y = f(x;b)$。当函数 $y = f(x;b)$ 为关于 b 的线性函数时，称这种曲线拟合为线性拟合[5]。

速度：地点速度又称为点速度，指车辆通过道路某一特定地点的瞬时速度。在现有交通调查中，线圈和微波检测器采集得到的速度数据非常接近此定义。

流量：流量是指在单位时间段内通过道路某一指定地点、某一断面或道路.上某一条车道的交通实体数[6]。

时间占有率：占有率 o 即车辆的时间密集度，就是在一定的观测时间 T 内,车辆通过检测器时所占用的时间与观测总时间的比值[7]。

方差检验： 质量监控术语，是用于比较两个或者多个变量数据的样本，来确定它们之间的差别是简单随机的，或者是由于流程之间统计上显著的差别所致。方差分析要求数据为正态分布，且方差相等。

4.2 符号说明

表 1 符号说明

符号	意义
Q	流量
o	占有率
V	速度
E^2	误差平方和
P（X）	拟合函数

5 模型建立与求解

5.1 问题一的分析与求解

5.1.1 问题分析

题目要求的是对于给定数据(data 1)，需要建立识别问题数据（包括异常数据、重复记录等） 的方法和流程，并完成对问题数据的修正及缺失数据的填补。我们将数据遍历，检查数据的完整性，通过方差检验和阈值检验对异常数据进行清洗，方便后面的时间维度上三要素模型的建立，同时也提高数据的质量，保证后续试验结果的正确性。

5.1.2 数据清洗流程

交通检测器识别问题数据的方法：将数据导入处理工具，利用 pandas 来数据清洗，包括去重复值，数据填补，数据删除。具体流程如图 1 所示。

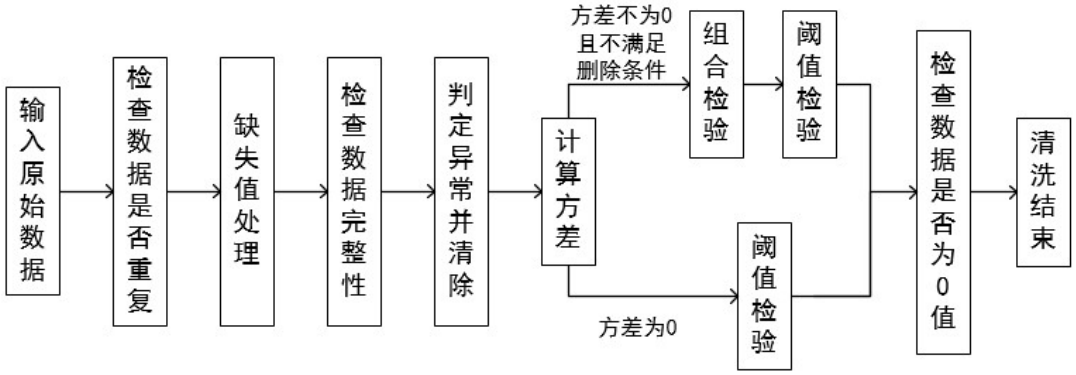

图 1 交通检测器识别问题数据的流程图

交通检测器数据清洗的**步骤**：

步骤一：将原始数据导入处理工具，遍历原始数据后，查看数据是否存在连续重复，若存在就进行去重，若不存在则进行下一步。

步骤二：查看数据是否存在缺失值，若存在缺失值就使用**邻近线性差值法**填充缺失值，若不存在则进行下一步。

步骤三：检查数据是否完整，若不完整就判定其为异常并清除，若完整则进行下一步。

步骤四：计算方差进行方差检验以判断异常数据，若方差为 0 就进行阈值检验，若方差不为 0 就进行判断其是否满足删除条件，若满足就直接删除，不满足则进行组合检验和阈值检验。不满足检验条件的数据直接删除。

步骤五：阈值检验完毕后检查数据是否为 0 值，若是 0 值就判定为异常并删除，若不为 0 值则保留。

步骤六：数据清洗完成，形成新的数据。

在这个问题中我们用 Pandas 实现数据清洗，处理空值。这里涉及到对空值的处理：删除或是替换然后填充（这里删除分为有一个空值删除还是有两个空值删除一行。替换可以用特定值替换或者是用平均值替换）。方差检验和阈值检验部分处理：根据查阅的资料以及离群值检测来处理异常的数据。最终经过此次数据清洗的值，原始值有 8635 个，我们进行数据清洗后的数据有 8522 个，可以看出已经剔除了异常数据。

5.2 问题二的分析与求解

5.2.1 问题分析

题目要求的是先在时间维度上对交通流参数（流量、速度和时间占有率）进行建模，其次对这三个参数之间的关系进行建模，最后补全 data2 缺失的数据。而我们就按照这个顺序进行建模，从已知的数据出发，先进行数据清洗，再重构数据集，对三个参数分别在时间尺度上利用拟合进行建模，得出模型 1，找出三参数的关系公式，建立三维坐标从而建立模型 2。

5.2.2 模型 1:在时间维度上的交通流参数多元回归模型

1.模型分析

题目要求的是先在时间维度上对交通流参数（流量、速度和时间占有率）分别进行建模，其次对这三个参数之间的关系进行建模，最后补全 data2 缺失的数据。而我们就按照这个顺序进行建模，从已知的数据出发，先进行数据清洗，再重构数据集，对三个参数分别在时间尺度上利用拟合进行建模，得出模型 1，找出三参数的关系公式，建立三维坐标从而建立模型 2。

2.模型建立

最小二乘法：最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。最小二乘法还可用于曲线拟合。其他一些优化问题也可通过最小化能量或最大化熵用最小二乘法来表达。

对给定数据点集合 $\{(X_i, y_i)\}(i = 0,1,2...,m)$，在取定的函数类 φ 中，求 $p(X) \in \varphi$，使误差的平方和 E^2 最小，$E^2 = [p(X_i) - y_i]^2$ 的求和。从几何意义上讲，就是寻求与给定点集 $E^2 = [p(X_i) - y_i]^2$ 的距离平方和为最小的曲线 $y = p(x)$。函数 $p(x)$ 称为拟合函数或最小二乘解，求拟合函数 $p(x)$ 的方法称为曲线拟合的最小二乘法。最小二乘法的矩阵形式为: $Ax = b$，其中 A 为 $n*k$ 的矩阵，x 为 $k*1$ 的列向量，b 为 $n*1$ 的列向量。如果 $n > k$ (方程的个数大于未知量的个数)，这个方程系统称为矛盾方程组(Over Determined System)，如果 $n < k$ (方程的个数小于未知量的个数)，这个系统就是 Under Determined System.

先进行问题一中的数据清洗，清除异常数据，步骤如问题一所示。

然后重构数据集，这里我们取每一天固定检测时间的均值，并以其序列号为时间索引，例如：0 号时间索引是已知数据集 data2 中的每一天的 00:00:00 检测到的数据的平均值。重构数据集的时间索引为 1 天，是 0-224 号。由此我们便可得出在时间维度上对交通流参数进行的建模。自变量为时间索引，因变量为这三个参数（速度、车流量和时间占有率），分别形成了二维的关系图。

（1）时间维度和速度的关系图如图 2 所示。

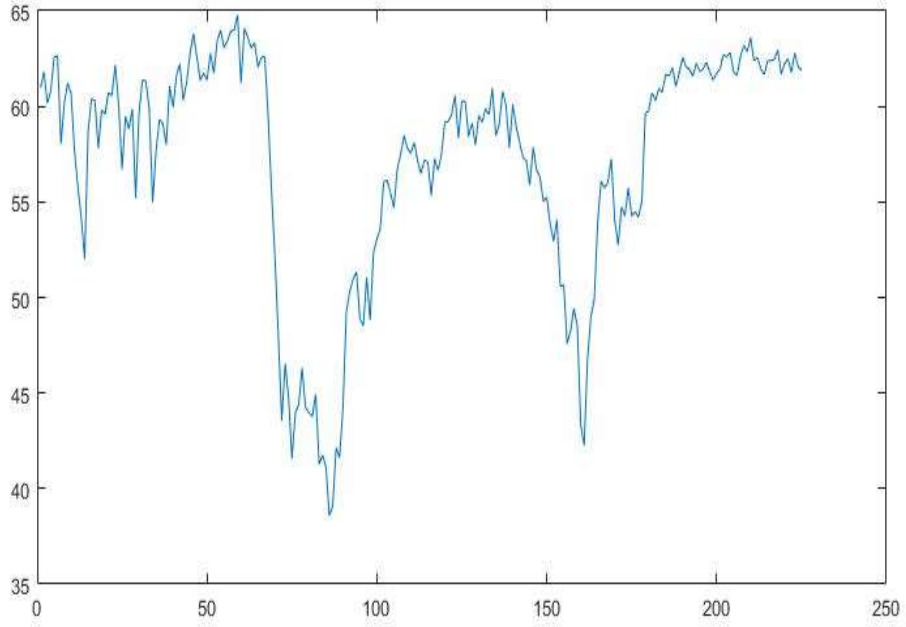

图 2 时间维度和速度的关系图

（2）时间维度和车流量的关系图如图 3 所示。

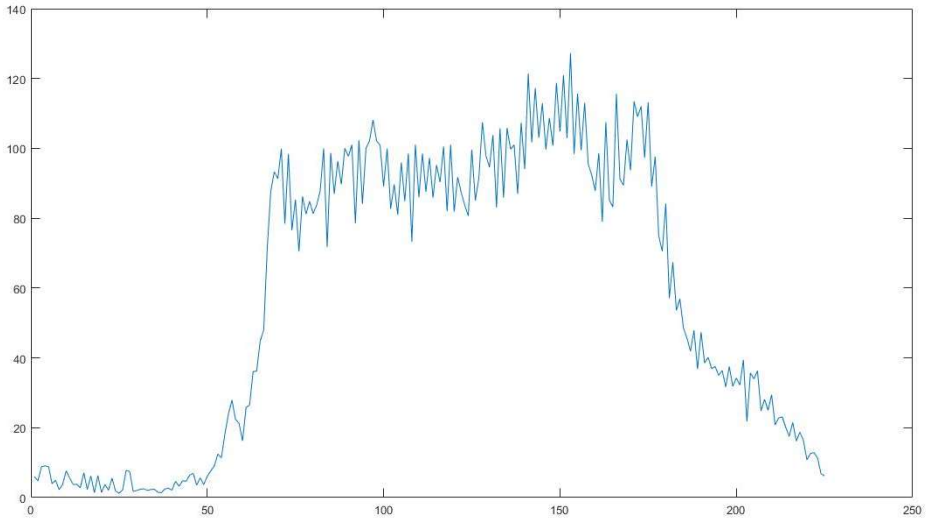

图 3 时间维度和车流量的关系图

（3）时间维度和时间占有率的关系图如图 4 所示。

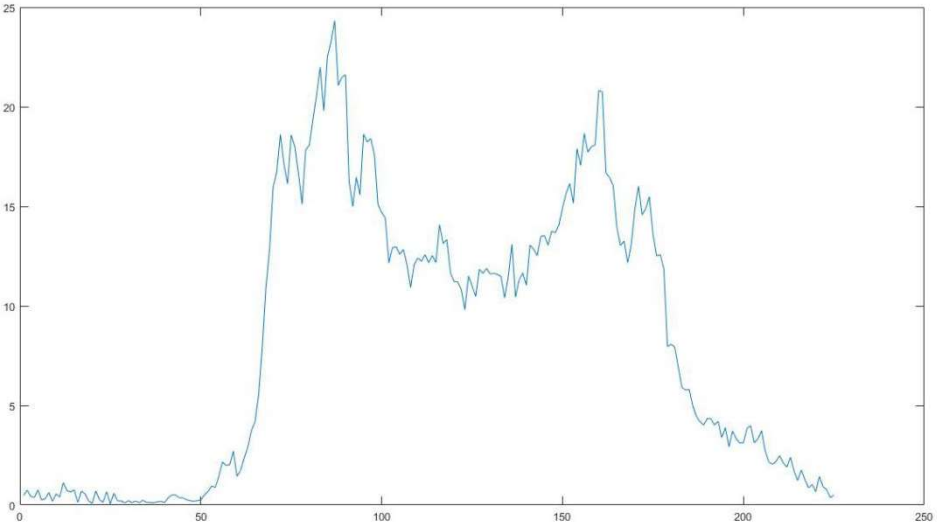

图 4 时间维度和时间占有率的关系图

使用 Gauss 6 函数建立多元线性回归模型，以获取时间维度和交通流参数的关系。具体表达式见表 2。

表 2 交通流三参数的具体表达式

Parameter	Formula
Speed	f(x)=a1*exp(-((x-b1)/c1)^2)+a2*exp(-((x-b2)/c2)^2) +a3*exp(-((x-b3)/c3)^2)+a4*exp(-((x-b4)/c4)^2) +a5*exp(-((x-b5)/c5)^2)+a6*exp(-((x-b6)/c6)^2)
Volume	f(x)=a1*exp(-((x-b1)/c1)^2)+a2*exp(-((x-b2)/c2)^2) +a3*exp(-((x-b3)/c3)^2)+a4*exp(-((x-b4)/c4)^2) +a5*exp(-((x-b5)/c5)^2)+a6*exp(-((x-b6)/c6)^2)
Occupancy	f(x)=a1*exp(-((x-b1)/c1)^2)+a2*exp(-((x-b2)/c2)^2) +a3*exp(-((x-b3)/c3)^2)+a4*exp(-((x-b4)/c4)^2) +a5*exp(-((x-b5)/c5)^2) + a6*exp(-((x-b6)/c6)^2)

具体参数见表 3。

表 3 参数数值表

Speed	a1=13.33	a2=59.19	a3=37.64	a4=59.91	a5=26.85	a6=9.154
	b1=64.76	b2=286,9	b3=139	b4=15.44	b5=208	b6=173.1
	c1=18.38	c2=84.72	c3=35.52	c4=59.91	c5=44.01	c6=18.65
Volume	a1=138.4	a2=-78.35	a3=42.64	a4=-38.87	a5=24.06	a6=-8.894
	b1=204.8	b2=263.1	b3=73.79	b4=53.01	b5=210.1	b6=146.8
	c1=147	c2=93.52	c3=26.29	c4=18.93	c5=18	c6=12.49
Occupancy	a1=3.161	a2=14.47	a3=16.44	a4=5.304	a5=0.971	a6=2.66
	b1=14.87	b2=180.9	b3=91.68	b4=117.4	b5=142.2	b6=61.48
	c1=58.21	c2=71.59	c3=22.9	c4=12.35	c5=5.098	c6=29.12

5.2.3 模型求解

我们通过使用 Gauss 6 函数来建立多元回归模型以获取在时间维度上交通流参数的关系。通过建模获取了时间维度和交通流参数的关系并形成了拟合图。

（1）时间维度和速度的拟合图如图 5 所示。

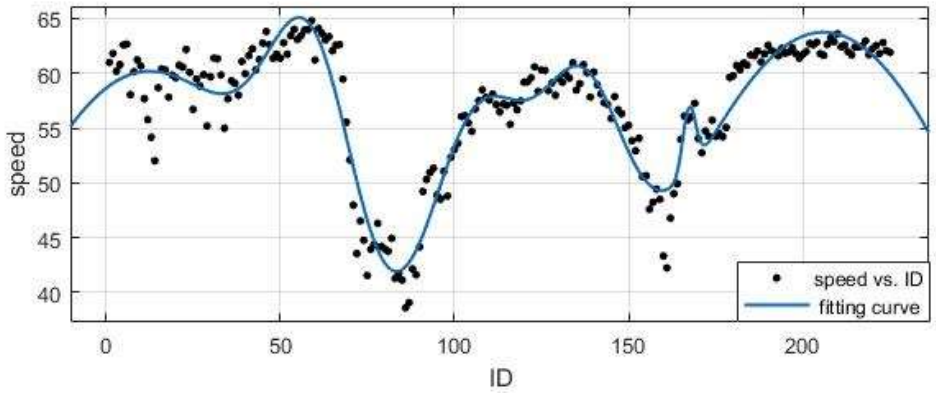

图 5 时间维度和速度的拟合图

（2）时间维度和车流量的拟合图如图 6 所示。

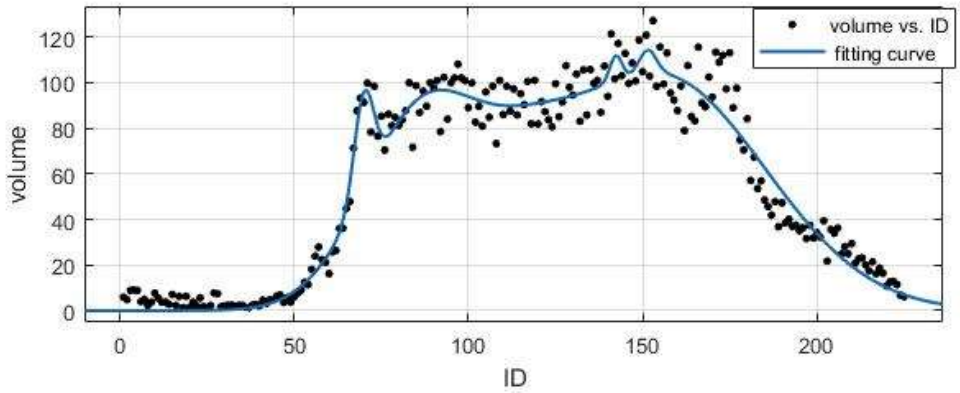

图 6 时间维度和车流量的拟合图

（3）时间维度和时间占有率的拟合图如图 7 所示。

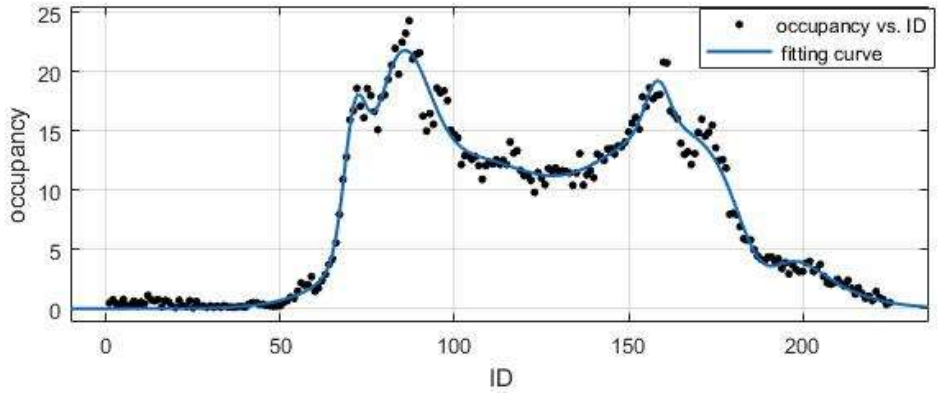

图 7 时间维度和时间占有率的拟合图

从图 5、6、7 中可以看出，RMSE 接近于 0，说明模型选择和拟合较为合适，数据预测就较准确。R-square 是通过数据的变化来表征一个拟合的好坏，其正常取值范围为[0，1]，越接近 1，表明方程的变量对 y 的解释能力越强，这个模型对数据拟合的也较好。由表 3 可看出 R-square 很接近 1，说明拟合效果较为理想。

表 3 相关系数分析表

	Speed	Volume	Occupancy
SSE	928	1.547e+04	198.6
R-square	0.8861	0.9591	0.982
Adjusted R-square	0.8767	0.9557	0.9805
RMSE	2.117	8.645	0.9796

5.2.4 模型 2 交通流三参数间的关系的曲面拟合模型

1.模型分析

由问题分析可知，我们的模型使用曲面拟合的方法进行，即采用三维坐标轴将三个参数量作为变量描绘出坐标系中的点的位置，通过曲面拟合的方法得出拟合曲面表达式，这样就可以得到三参数之间的关系，如图 8 所示。

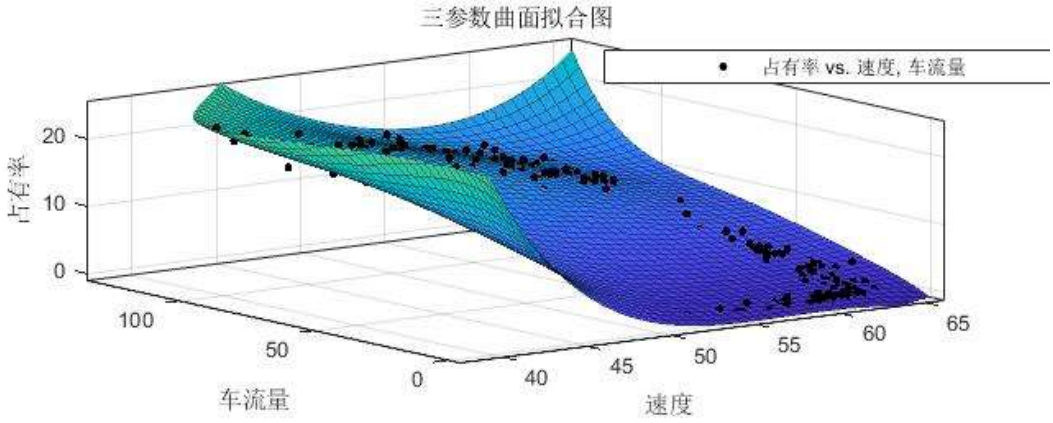

图 8 交通流参数间的关系

更好的，我们采用多元多项式进行拟合，同样由于需要寻求三参数之间的关系，我们使用时间占有率作为标签，将速度和车流量作为变量进行建模分析，采用 Scikit-learn 的 liner-model 和 curve-fit 进行变量的拟合，得到关系表达式，进而得到三者之间的关系。

对应的关系表达式为： $Result = \text{int}\,ercept + coefficients * \text{var}\,iableX.\text{T}$

2.模型建立

建立曲面拟合模型的步骤为：

步骤一：数据集重构；

步骤二：初始化参数；

步骤三：设置参数限制；

步骤四：使用 PolynomialFeatures 进行特征构造；

步骤五：特征转换与线性回归；

步骤六：建立模型，开始训练；

步骤七：得到交通流参数间的关系。

3.模型求解

将预处理的数据带入上述模型中，提取需要用到的列作为变量，以 dataframe 格式存储，在运算时转为 numpy 的数组形式，通过 python 变成，得到对应阶数的特征构造多项式，接着我们对得到的特征进行转换，将转换后的特征数据作为输入值带入模型进行训练。将训练得到的结果再次进行特征转换得到我们期望的数据形式，然后与实际值进行误差对比，如图 9 所示。

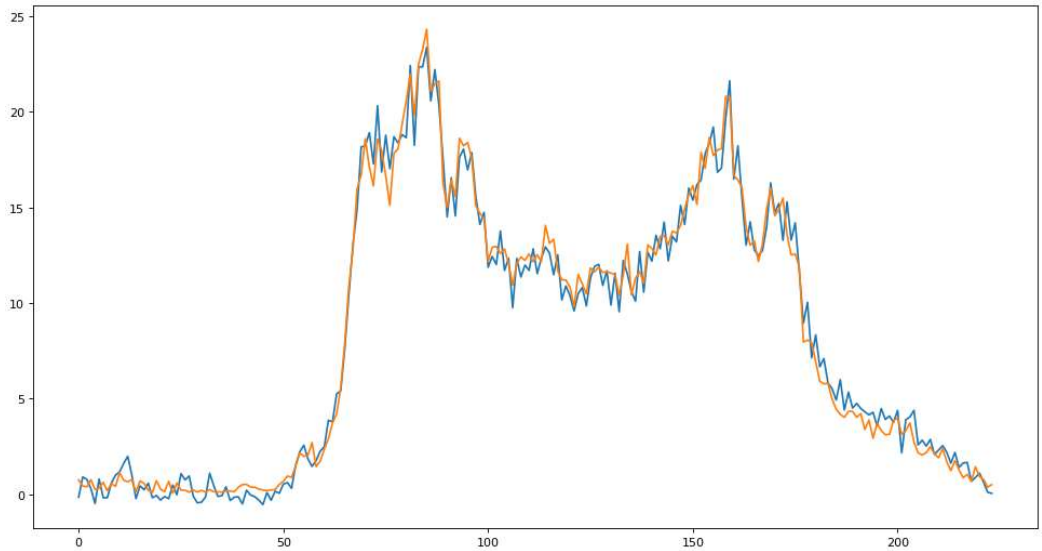

图 9 实际值与拟合值的误差对比图

将拟合值和实际值进行比较并计算误差，如图 10 所示。我们的拟合值和实际值的误差最大约为 2.21，证明了选用该模型进行拟合的正确性及数据处理的可靠性。

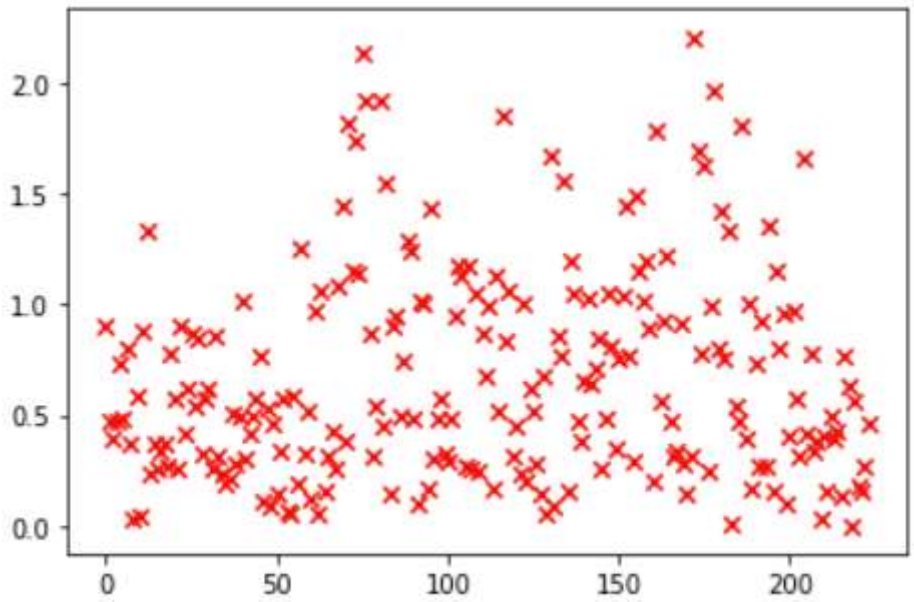

图 10 实际值与拟合值的误差散点图

5.3 问题三的分析与求解

5.3.1 问题分析

需要对已知数据建立预测模型，首先处理已知数据得出其相关性，建立预测模型后可对模型进行分析，输入相关时间数据做出预测，最后补全缺失数据。这里原本想使用问题一问题二的步骤先填充缺失值再进行建模，经过模拟训练后放弃了该想法，选择了先根据已有数据建立预测模型，选用的是 ARIMA 模型。

5.3.2 **模型 3 ：ARIMA 预测模型**

建立 ARIMA 预测模型的步骤为：

步骤一：导入 data3 的数据并进行数据预处理；

步骤二：观察数据是否具有周期性；

步骤三：对数据进行平稳性分析；

步骤四：检查数据的自相关和偏自相关性；

步骤五：确定 ARIMA 模型；

步骤六：模型结果评价；

步骤七：进行拟合预测；

步骤八：缺失值补充。

5.3.3 模型求解

建立 ARIMA 预测模型后得到模型拟合度，见表 4、5、6。可以看到 R 方值为 0.818，与 1 很接近，证明拟合程度很好。

表 4 模型拟合度

拟合统计	平均值	标准误差	最小值	最大值
平稳 R 方	.200	.093	.134	.306
R 方	.818	.056	.755	.860
RMSE	10.342	8.634	4.254	20.223
MAPE	37.378	24.556	9.347	55.093
MaxAPE	1842.085	786.530	935.188	2337.744
MAE	6.662	6.495	2.254	14.121
MaxAE	78.117	30.523	54.807	112.666
正态化 BIC	4.229	1.610	2.902	6.020

表 5 模型统计

模型统计

模型	预测变量数	模型拟合度统计		杨-博克斯 Q(18)			离群值数
		平稳 R 方	R 方	统计	DF	显著性	
occupancy-模型_1	1	.159	.839	46.529	16	.000	0
speed-模型_2	1	.134	.755	43.897	16	.000	0
volume-模型_3	1	.306	.860	63.798	16	.000	0

表 6 ARIMA 模型参数

					估算	标准误差	t	显著性
occupancy-模型_1	occupancy	不转换	常量		.208	.056	3.739	.000
			AR	延迟 1	.188	.028	6.795	.000
			差异		1			
			MA	延迟 1	.615	.022	27.679	.000
	startdate	不转换	分子	延迟 0	-4.814E-6	1.120E-6	-4.297	.000
speed-模型_2	speed	不转换	常量		-.161	.106	-1.512	.131
			AR	延迟 1	.122	.033	3.713	.000
			差异		1			
			MA	延迟 1	.525	.028	18.574	.000
	startdate	不转换	分子	延迟 0	3.594E-6	2.073E-6	1.734	.083
volume-模型_3	volume	不转换	常量		1.427	.186	7.682	.000
			AR	延迟 1	.009	.020	.447	.655
			差异		1			
			MA	延迟 1	.671	.015	44.432	.000
	startdate	不转换	分子	延迟 0	-3.303E-5	3.747E-6	-8.813	.000

所以 ARIMA 模型表达式为：

$$\begin{cases} \Delta x_t = 0.122\Delta x_{t-1} + q_t - 0.525q_{t-1}, Speed \\ \Delta x_t = 0.009\Delta x_{t-1} + q_t - 0.671q_{t-1}, Volume \\ \Delta x_t = 0.188\Delta x_{t-1} + q_t - 0.615q_{t-1}, \text{Occupancy} \end{cases}$$

残差的 ACF 和 PACF 图，如图 11 所示，可以看到都是平稳的，因此 ARIMA(1,1,1) 是合理的。

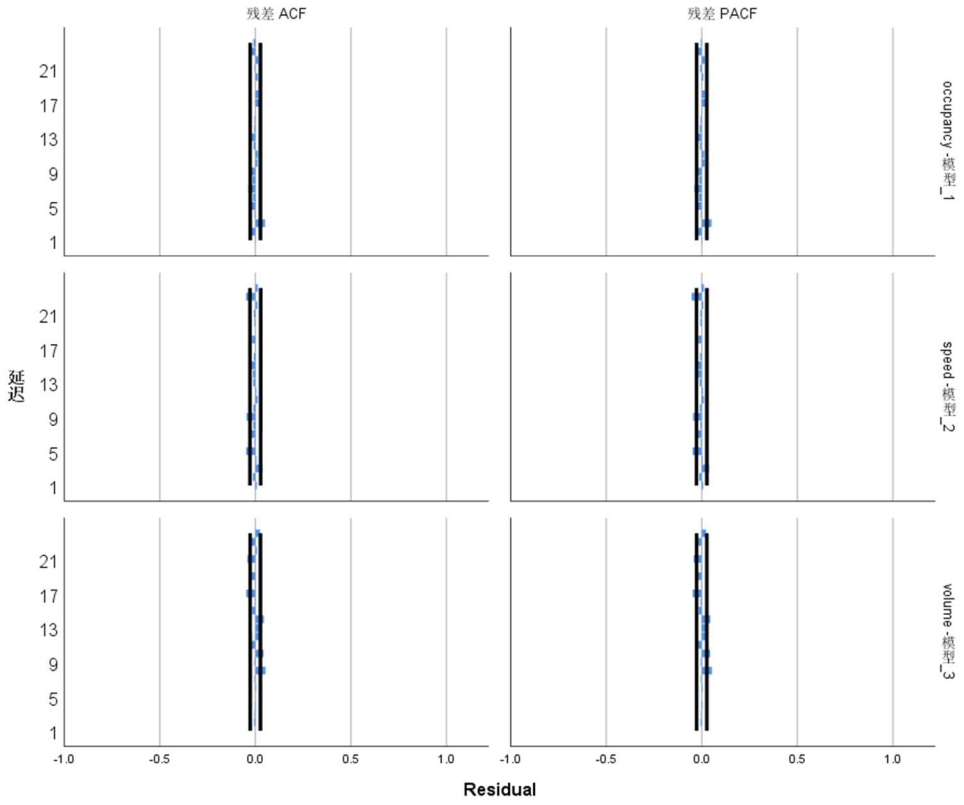

图 11 残差 ACF/PACF 图表

进行拟合预测，如图 12 所示，可以看到拟合效果很好。实际数据和拟合数据重合度较高，具有说服力。将缺失值的时间维度带入表达式中，得到的数据可直接替换数据集（data3）中的 NA 值。

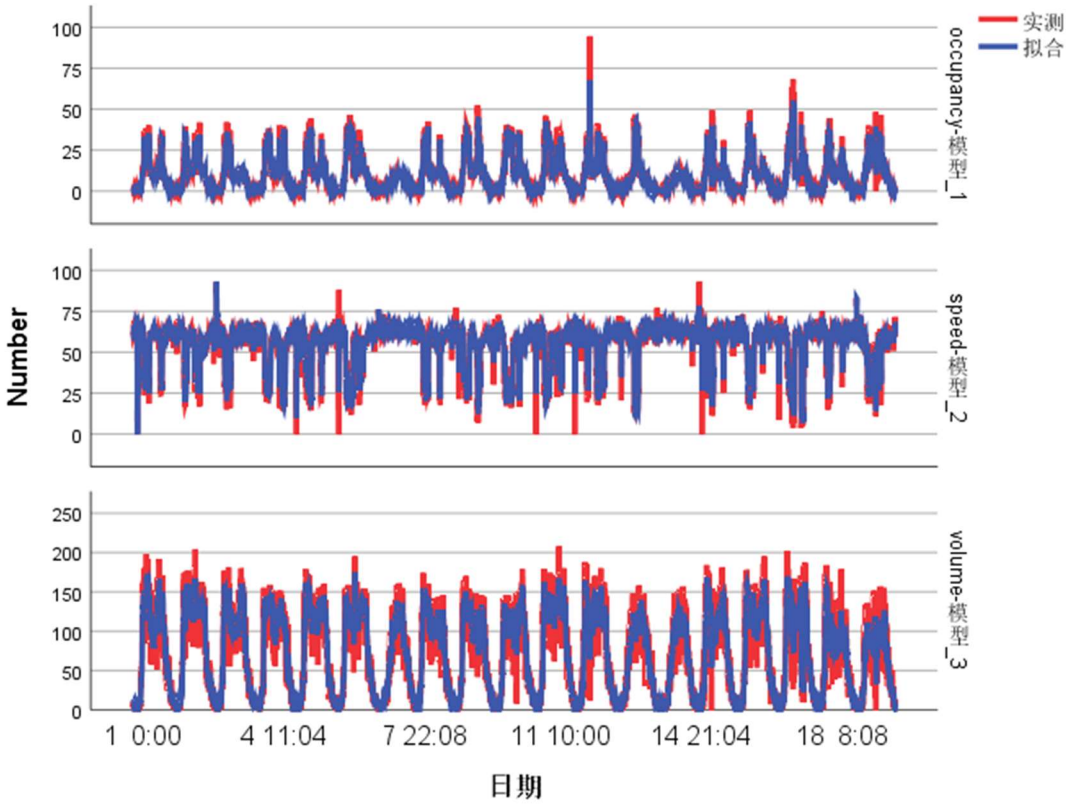

图 12 序列图表

最后将缺失值的时间维度代入表达式中，得到的数据可直接替换数据集（data3）中的 NA 值。部分数据见表 7。

表 7 问题三填充后的部分数据

Start time	speed	volume	occupancy	speed_处理后	volume_处理后	occupancy_处理后
2011-03-01T00:00:00	NA	NA	NA	60.87	6	0.96
2011-03-01T00:05:00	NA	NA	NA	60.98	6	0.98
2011-03-01T00:10:00	NA	NA	NA	61.67	5	0.74
2011-03-01T00:15:00	NA	NA	NA	64.6	4	0.16
2011-03-01T00:20:00	NA	NA	NA	67	7	0.27
2011-03-01T00:25:00	NA	NA	NA	66.5	2	0.1
2011-03-01T00:30:00	NA	NA	NA	60.94	6	0.41
2011-03-01T00:35:00	NA	NA	NA	53.09	2	0.11
2011-03-01T00:40:00	NA	NA	NA	59.36	4	0.41
2011-03-01T00:45:00	NA	NA	NA	60.84	4	0.4
2011-03-01T00:50:00	NA	NA	NA	61.93	2	0.13
2011-03-01T00:55:00	NA	NA	NA	66.4	3	0.16
2011-03-01T01:00:00	NA	NA	NA	63.71	4	0.45
2011-03-01T01:05:00	NA	NA	NA	67.8	2	0.01
2011-03-01T01:10:00	NA	NA	NA	63.5	0	0
2011-03-01T01:15:00	NA	NA	NA	60.67	0	0.03
2011-03-01T01:20:00	NA	NA	NA	57.56	1	0.04
2011-03-01T01:25:00	NA	NA	NA	61.75	0	0
2011-03-01T01:30:00	NA	NA	NA	62.33	1	0.06
...	...	...	...	...	...	...
2011-03-01T08:15:00	NA	NA	NA	31.27	82	30.77
2011-03-01T08:20:00	NA	NA	NA	27.36	110	33.17
2011-03-01T08:25:00	NA	NA	NA	28.44	131	32.78
2011-03-01T08:30:00	NA	NA	NA	37.76	140	25.72
2011-03-01T08:35:00	NA	NA	NA	35.57	136	26.84
2011-03-01T08:40:00	NA	NA	NA	42.37	144	23.05
2011-03-01T08:45:00	NA	NA	NA	46.73	148	19.98
2011-03-01T08:50:00	NA	NA	NA	49.05	158	21.4
2011-03-01T08:55:00	NA	NA	NA	34.36	132	30.41
2011-03-01T09:00:00	NA	NA	NA	32.86	136	30.43
2011-03-01T09:05:00	NA	NA	NA	34.16	91	26.74
2011-03-01T09:10:00	NA	NA	NA	32.83	129	28.7
2011-03-01T09:15:00	NA	NA	NA	19.39	85	37.99
2011-03-01T09:20:00	NA	NA	NA	38.67	124	23.82
2011-03-01T09:25:00	NA	NA	NA	41.8	140	22.58
2011-03-01T09:30:00	NA	NA	NA	47.97	135	19.4

2011-03-01T09:35:00	NA	NA	NA	49.9	142	22.65

6 模型评价与推广

6.1 模型优点

1.该模型深入研究了交通检测器数据的各项指标，充分利用数据，具有说服力。

2.该模型清晰明了地刻画了交通系统在时间上的演化过程，创新性地通过数据对交通流量进行了预测。

3.该模型利用了不同时间点检测出了交通流参数的数据的除了交通流参数在时间维度上的关系模型以及交通流参数之间的关系，十分巧妙。

6.2 模型缺点

交通流参数之间存在着联系，但不仅限于线性回归关系，从而会影响模型的正确性。

6.3 模型推广

本文所用的模型具有很强的通用性，具有多元限制条件的问题都可以类比使用，比如文中是交通检测数据参数之间的关系，在研究其他方面的数据参数之间的关系时也可套用本模型。不仅仅局限于交通领域，也可扩展到生活其他领域，适应我们的日常需求。

参考文献

[1]孙亚, 朱鲤.ITS 检测器交通流数据质量控制系统研究[J]. 测控技术,2008, 027(007):35-37,40.

[2]李琦, 姜桂艳. 城市快速路车辆检测器数据质量评价与控制方法[J]. 交通运输工程学报,2013(02):124-130.

[3]谷远利, 马韵楠, 李巨伟. 道路交通流数据质量控制与评价[C]// 2013 年中国城市交通规划年会暨第 27 次学术研讨会.

[4]李仲来.最小二乘法介绍[J].数学通报, 1992 (2) : 42-45.

[5] 郑德如.回归分析和相关分析[M].上海:上海人民出版社，1983: 2-96.

[6]胡佩锋;交通流量短时预测方法研究[D] ;北京交通大学;2007 年.

[7]贺国光，李宇，马寿峰.基于数学模型的短时交通流预测算法探讨[J].系统工程理论与实践，2000.

附录

附录 1

数据清理部分代码（全部见附件）

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import read_csv
f = open('data 1.csv',encoding='UTF-8')
names = ['starttime','speed','volume','occupancy']
# filename=input("文件名： ")
# f=open(filename,encoding='UTF-8')
# names=['作业日期',' η CO',' η H2','TF(℃)','TC(℃)','mass','送风流量']
data=read_csv(f,names=names,header=1)
print(data)
data.starttime.drop_duplicates(keep=False,inplace=True)
#   把重复的值全部删除
print(data[data.duplicated(keep=False)])
# 检验。现在没有重复值
# 缺省值处理
#speed 缺省值填补
count = 1
for count in range(data.shape[0]):
#         print(tup[0],'-->',tup[1::],type(tup[1:]))
      speedtmp = data.loc[count,["speed"]].values
      if pd.isnull(speedtmp) :
            if count == 1 :  #先给第一个缺省值赋值
                  tmpcount = 0
                  while pd.isnull(data.loc[count+tmpcount,["speed"]].values) :
                        tmpcount+=1
                  data.loc[count,["speed"]] = data.loc[count+tmpcount,["speed"]]
                  print("speed 第一行为空")
            elif count >2 and count <= data.shape[0]: #或者使用  len(data)
                  tmpcount = 0
                  while pd.isnull(data.loc[count+tmpcount,["speed"]].values) :
                        tmpcount+=1
                  data.loc[count,["speed"]]                    =                    (data.loc[count-
1,["speed"]].values.astype(float)+data.loc[count+tmpcount,["speed"]].values.astype(float))/2
                  print(count ,"speed 修改成功")
            elif count == data.shape[0]+1:
                  data.loc[count,["speed"]] = data.loc[count-1,["speed"]]
                  print("speed 最后一行为空")
```

```
        else :
            Pass
#volume 缺省值填补
count = 1
for count in range(data.shape[0]):
#         print(tup[0],'-->',tup[1::],type(tup[1:]))
        volumetmp = data.loc[count,["volume"]].values
        if pd.isnull(volumetmp) :
            if count == 1 :   #先给第一个缺省值赋值
                tmpcount = 0
                while pd.isnull(data.loc[count+tmpcount,["volume"]].values) :
                    tmpcount+=1
                data.loc[count,["volume"]] = data.loc[count+tmpcount,["volume"]]
                print("volume 第一行为空")
            elif count >2 and count <= data.shape[0]: #或者使用  len(data)
                tmpcount = 0
                while pd.isnull(data.loc[count+tmpcount,["volume"]].values) :
                    tmpcount+=1
                data.loc[count,["volume"]] = (data.loc[count-
1,["volume"]].values.astype(float)+data.loc[count+tmpcount,["volume"]].values.astype(float))/2
                print(count ,"volume 修改成功")
            elif count == data.shape[0]+1:
                data.loc[count,["volume"]] = data.loc[count-1,["volume"]]
                print("volume 最后一行为空")
        else :
            pass
#occupancy 缺省值填补
count = 1
for count in range(data.shape[0]):
#         print(tup[0],'-->',tup[1::],type(tup[1:]))
        occupancytmp = data.loc[count,["occupancy"]].values
        if pd.isnull(occupancytmp) :
            if count == 1 :   #先给第一个缺省值赋值
                tmpcount = 0
                while pd.isnull(data.loc[count+tmpcount,["occupancy"]].values) :
                    tmpcount+=1
                data.loc[count,["occupancy"]] = data.loc[count+tmpcount,["occupancy"]]
                print("occupancy 第一行为空")
            elif count >2 and count <= data.shape[0]: #或者使用  len(data)
                tmpcount = 0
                while pd.isnull(data.loc[count+tmpcount,["occupancy"]].values) :
                    tmpcount+=1
                data.loc[count,["occupancy"]] = (data.loc[count-
```

```
1,["occupancy"]].values.astype(float)+data.loc[count+tmpcount,["occupancy"]].values.astype(float))/2
                print(count ,"occupancy 修改成功")
            elif count == data.shape[0]+1:
                data.loc[count,["occupancy"]] = data.loc[count-1,["occupancy"]]
                print("occupancy 最后一行为空")
        else :
            pass
print (data)
#显示 data 的相关信息
data.dtypes
data.info()
#  异常值处理
#先进行方差检验，若为零则进行阈值检验，若不为零则进行组合检验，然后进行方差检验
# #以最大值的百分之九十五为界限
# data.max() #最值、累加
# data['speed'].max()
# threshold_speed = data['speed'].max() * 0.95
# data.loc[data['speed'] >= threshold_speed]
from pandas import Series
s= pd.Series()
print(s,s.dtype)
for outliercount in range(2 , data.shape[0]):
#         print(outliercount)
#             tmp = data.loc[outliercount,["speed"]].values.astype(float) - data.loc[outliercount - 1,["speed"]].values.astype(float)
#         print(tmp,tmp.dtype)
        if ( (data.loc[outliercount,["speed"]].values.astype(float) - data.loc[outliercount-1,["speed"]].values.astype(float))**2 ==0 and
            (data.loc[outliercount,["volume"]].values.astype(float) - data.loc[outliercount-1,["volume"]].values.astype(float))**2 ==0 and
            (data.loc[outliercount,["occupancy"]].values.astype(float) - data.loc[outliercount-1,["occupancy"]].values.astype(float))**2 ==0) :
            #阈值检测设置速度：0-140；流量：0-225；占用率：0-100；
            if (0<= data.loc[outliercount,["speed"]].values <= 140) & (0<= data.loc[outliercount,["volume"]].values <= 225) & (0<= data.loc[outliercount,["occupancy"]].values <= 100) :#阈值检验
                pass
#                   print("阈值检测通过") #阈值检测通过
            else :     s= s.append(Series([outliercount]))#添加该行标记到 series，之后删除该行数据
                print("添加行", outliercount)
        elif ((data.loc[outliercount,["speed"]].values ==0 and data.loc[outliercount,["volume"]].values ==0 and data.loc[outliercount,["occupancy"]].values !=0) or
```

```
            (data.loc[outliercount,["speed"]].values ==0 and data.loc[outliercount,["volume"]].values !=0 and data.loc[outliercount,["occupancy"]].values ==0) or
            (data.loc[outliercount,["speed"]].values !=0 and data.loc[outliercount,["volume"]].values ==0 and data.loc[outliercount,["occupancy"]].values ==0) or
            (data.loc[outliercount,["speed"]].values !=0 and data.loc[outliercount,["volume"]].values !=0 and data.loc[outliercount,["occupancy"]].values ==0) or
            (data.loc[outliercount,["speed"]].values ==0 and data.loc[outliercount,["volume"]].values !=0 and data.loc[outliercount,["occupancy"]].values !=0)) :
        s= s.append(Series([outliercount])) #加该行标记到 series，之后删除该行数据
        print("添加行", outliercount)
    else :
        #组合检验+阈值检验
        #组合检验（注意量纲）：（速度*1000/60）*（占用率*5）/（流量*5）=结果，结果：0-2038
        Combination_testing = (data.loc[outliercount,["speed"]].values*1000/60)*(data.loc[outliercount,["occupancy"]].values*5)/(data.loc[outliercount,["volume"]].values*5)
        if (0<= Combination_testing <=2038) :
            pass
#               print("组合检验通过") #组合检验通过
        else :      s= s.append(Series([outliercount])) #添加该行标记到 series，之后删除该行数据
        if (0<= data.loc[outliercount,["speed"]].values <= 140) & (0<= data.loc[outliercount,["volume"]].values <= 225) & (0<= data.loc[outliercount,["occupancy"]].values <= 100) :#阈值检验
            pass
#               print("阈值检测通过") #阈值检测通过
        else :      s= s.append(Series([outliercount])) #添加该行标记到 series，之后删除该行数据
            print("添加行", outliercount)
s = s.astype('int')
s = s.sort_values(ascending=False)
print(s)
#删除对应列数据
print(data.shape[0])
data.drop(s,inplace=True) #根据 S 的编号删除
print(data.shape[0])
data.to_csv('data 1_change.csv')        #写入 csv 文件
```

附录 2

曲线拟合部分代码（全部见附件）

```
import pandas as pd
```

```
#显示所有列
pd.set_option('display.max_columns', None)
#显示所有行
pd.set_option('display.max_rows', None)
#设置 value 的显示长度为 100，默认为 50
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import read_csv
f = open('data2_2014.csv',encoding='UTF-8')
names = ['Index_Num','starttime','resolution','speed','volume','occupancy','速度','车流量','占有率','
年','月','日','日期','时间','速度均值','车流量均值','占有率均值','ID','filter']
data=read_csv(f,names=names,header=1)
print(data)
#单变量曲线拟合
#拟合高斯分布
from scipy.optimize import curve_fit
import math
#自定义函数  e 指数形式
def func(x, a,u, sig):
    return    a*(np.exp(-(x - u) ** 2 /(2* sig **2))/(math.sqrt(2*math.pi)*sig))*(431+(4750/x))
#定义 x、y 散点坐标
x = np.array(data['ID'])
x=np.array(x)
# x = np.array(range(20))
print('x is :\n',x)
num = np.array(data['占有率均值'])
y = np.array(num)
print('y is :\n',y)
popt, pcov = curve_fit(func, x, y,p0=[3.1,4.2,3.3])
#获取 popt 里面是拟合系数
a = popt[0]
u = popt[1]
sig = popt[2]
yvals = func(x,a,u,sig) #拟合 y 值
print(u'系数 a:', a)
print(u'系数 u:', u)
print(u'系数 sig:', sig)
#绘图
plot1 = plt.plot(x, y, 's',label='original values')
plot2 = plt.plot(x, yvals, 'r',label='polyfit values')
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.legend(loc=4) #指定 legend 的位置右下角
plt.title('curve_fit')
plt.show()
# 高斯拟合
#单个高斯模型，如果曲线有多个波峰，可以分段拟合
def func(x, a,u, sig):
    return a*np.exp(-(x - u) ** 2 / (2 * sig ** 2)) / (sig * math.sqrt(2 * math.pi))
#混合高斯模型，多个高斯函数相加
def func3(x, a1, a2, a3, a4, a5 ,m1, m2, m3, m4, m5, s1, s2, s3, s4, s5):
    return a1 * np.exp(-((x - m1) / s1) ** 2) + a2 * np.exp(-((x - m2) / s2) ** 2) + a3 * np.exp(-((x - m3) / s3) ** 2) + a4 *np.exp(-((x - m4) / s4) ** 2) + a5 *np.exp(-((x - m5) / s5) ** 2)
#正弦函数拟合
#def fmax(x,a,b,c):
#     return a*np.sin(x*np.pi/6+b)+c
#fita,fitb=optimize.curve_fit(fmax,x,ymax,[1,1,1])
#非线性最小二乘法拟合
#def func(x, a, b,c):
#     return a*np.sqrt(x)*(b*np.square(x)+c)
# 用 3 次多项式拟合，可推广到 n 次多项式，数学上可以证明，任意函数都可以表示为多项式形式
# f1 = np.polyfit(x, y, 3)
# p1 = np.poly1d(f1)
# yvals = p1(x)  #拟合 y 值
#也可使用 yvals=np.polyval(f1, x)
#拟合，并对参数进行限制，bounds 里面代表参数上下限，p0 是初始范围，默认是[1,1,1]
x=np.arange(1,data.shape[0]+1,1)
num = np.array(data['占有率均值']) #<-自己的 y 值
numhunt = np.array(data['占有率均值']) #<-自己的 y 值
y = np.array(num)
yhunt = np.array(numhunt)
# param_bounds=([-np.inf,0],[np.inf,1])#设定 B 和 n 的下界和上界。其中 B 为负无穷到正无穷，n 为 0 到 1
popt, pcov = curve_fit(func3, x, y,maxfev=500000)
popthunt, pcovhunt = curve_fit(func, x, yhunt,p0=[2,2,2])
ahunt = popthunt[0]
uhunt = popthunt[1]
sighunt = popthunt[2]
a1 = popt[0]
u1 = popt[1]
sig1 = popt[2]
a2 = popt[3]
u2 = popt[4]
```

```
sig2 = popt[5]
a3 = popt[6]
u3 = popt[7]
sig3 = popt[8]
a4 = popt[9]
u4 = popt[10]
sig4 = popt[11]
a5 = popt[12]
u5 = popt[13]
sig5 = popt[14]
yvals = func3(x,a1,u1,sig1,a2,u2,sig2,a3,u3,sig3,a4,u4,sig4,a5,u5,sig5) #拟合 y 值
yhuntvals = func(x,ahunt,uhunt,sighunt) #拟合 y 值
print(u'系数 ahunt:', ahunt)
print(u'系数 uhunt:', uhunt)
print(u'系数 sighunt:', sighunt)
#绘图
plot1 = plt.plot(x, y, 's',label='insect original values')
plot2 = plt.plot(x, yvals, 'r',label='insect polyfit values')
plot3 = plt.plot(x, yhunt, 's',label='predator original values')
plot4 = plt.plot(x, yhuntvals, 'g',label='predator polyfit values')
plt.xlabel('date')
plt.ylabel('Nightly catches log10(N+1)')
plt.legend(loc=4) #指定 legend 的位置右下角
plt.title('insect/predator')
plt.show()
#  多项式拟合
#单个高斯模型，如果曲线有多个波峰，可以分段拟合
def func(x, a,u, sig):
    return a*np.exp(-(x - u) ** 2 / (2 * sig ** 2)) / (sig * math.sqrt(2 * math.pi))
#混合高斯模型，多个高斯函数相加
def func3(x, a1, a2, a3, a4, a5 ,m1, m2, m3, m4, m5, s1, s2, s3, s4, s5):
    return a1 * np.exp(-((x - m1) / s1) ** 2) + a2 * np.exp(-((x - m2) / s2) ** 2) + a3 * np.exp(-((x - m3) / s3) ** 2) + a4 *np.exp(-((x - m4) / s4) ** 2) + a5 *np.exp(-((x - m5) / s5) ** 2)
#正弦函数拟合
#def fmax(x,a,b,c):
#     return a*np.sin(x*np.pi/6+b)+c
#fita,fitb=optimize.curve_fit(fmax,x,ymax,[1,1,1])
#非线性最小二乘法拟合
#def func(x, a, b,c):
#     return a*np.sqrt(x)*(b*np.square(x)+c)
# 用 3 次多项式拟合，可推广到 n 次多项式，数学上可以证明，任意函数都可以表示为多项式形式
```

```
# f1 = np.polyfit(x, y, 3)
# p1 = np.poly1d(f1)
# yvals = p1(x)  #拟合 y 值
#也可使用 yvals=np.polyval(f1, x)
#拟合，并对参数进行限制，bounds 里面代表参数上下限，p0 是初始范围，默认是[1,1,1]
x=np.arange(1,data.shape[0]+1,1)
num = np.array(data['占有率均值']) #<-自己的 y 值
numhunt = np.array(data['占有率均值']) #<-自己的 y 值
y = np.array(num)
f1 = np.polyfit(x, y, 13)
p1 = np.poly1d(f1)
yvals = p1(x)
yhunt = np.array(numhunt)
# param_bounds=([-np.inf,0],[np.inf,1])#设定 B 和 n 的下界和上界。其中 B 为负无穷到正无穷，n 为 0 到 1
popt, pcov = curve_fit(func3, x, y,maxfev=500000)
popthunt, pcovhunt = curve_fit(func, x, yhunt,p0=[2,2,2])
ahunt = popthunt[0]
uhunt = popthunt[1]
sighunt = popthunt[2]
yhuntvals = func(x,ahunt,uhunt,sighunt) #拟合 y 值
print(u'系数 ahunt:', ahunt)
print(u'系数 uhunt:', uhunt)
print(u'系数 sighunt:', sighunt)
#绘图
plot1 = plt.plot(x, y, 's',label='insect original values')
plot2 = plt.plot(x, yvals, 'r',label='insect polyfit values')
plot3 = plt.plot(x, yhunt, 's',label='predator original values')
plot4 = plt.plot(x, yhuntvals, 'g',label='predator polyfit values')
plt.xlabel('date')
plt.ylabel('Nightly catches log10(N+1)')
plt.legend(loc=4) #指定 legend 的位置右下角
plt.title('insect/predator')
plt.show()
#  多元多项式拟合
import pandas as pd
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
x = np.array(data[['速度均值','车流量均值']])
y = np.array(data['占有率均值'])
def stdError_func(y_test, y):
```

```
    return np.sqrt(np.mean((y_test - y) ** 2))
def R2_1_func(y_test, y):
    return 1 - ((y_test - y) ** 2).sum() / ((y.mean() - y) ** 2).sum()
def R2_2_func(y_test, y):
    y_mean = np.array(y)
    y_mean[:] = y.mean()
    return 1 - stdError_func(y_test, y) / stdError_func(y_mean, y)
poly_reg =PolynomialFeatures(degree=2) #三次多项式
X_ploy =poly_reg.fit_transform(x)
lin_reg_2=linear_model.LinearRegression()
lin_reg_2.fit(X_ploy,y)
predict_y =   lin_reg_2.predict(X_ploy)
strError = stdError_func(predict_y, y)
R2_1 = R2_1_func(predict_y, y)
R2_2 = R2_2_func(predict_y, y)
score = lin_reg_2.score(X_ploy, y) ##sklearn 中自带的模型评估，与 R2_1 逻辑相同
print("coefficients", lin_reg_2.coef_)
print("intercept", lin_reg_2.intercept_)
print('degree={}: strError={:.2f}, R2_1={:.2f},   R2_2={:.2f}, clf.score={:.2f}'.format(
      3, strError,R2_1,R2_2,score))
#clf.score 它提供了一个缺省的评估法则来解决问题，简要的说，它用你训练好的模型在测试集上进行评分（0~1）1 分代表最好
#算一下得到的方程表达式跟实际的差值
intercept = lin_reg_2.intercept_
coefficients =lin_reg_2.coef_
yres =[]
for i in range(data.shape[0]):
      variable_x = [1,data.loc[i,["速度均值"]].values,data.loc[i,["车流量均值"]].values,(data.loc[i,["速度均值"]].values)**2,(data.loc[i,["速度均值"]].values*data.loc[i,["车流量均值"]].values),(data.loc[i,["车流量均值"]].values)**2]
      tmp_varX = np.array(variable_x,dtype=np.float32).T
#       print(tmp_varX, tmp_varX.dtype,tmp_varX.shape)
      yres.append((intercept + np.dot(coefficients , tmp_varX)).astype(float))
#做一下和源数据之间的差值
yres = np.array(yres,dtype=np.float32)
print(yres, yres.dtype,yres.shape)
dval= np.abs(yres - data['占有率均值'].values)
print("*******************************************")
print(dval)
#画出散点图
x = range(0,dval.shape[0],1)
print(len(x),len(dval))
plt.scatter(x, dval, marker = 'x',color = 'red', s = 40 ,label = 'First')
```

```
plt.show()
print(max(dval))
#开启一个窗口，num 设置子图数量，figsize 设置窗口大小，dpi 设置分辨率
fig = plt.figure(num=1, figsize=(15, 8),dpi=80)
#直接用 plt.plot 画图，第一个参数是表示横轴的序列，第二个参数是表示纵轴的序列
plt.plot(x,yres)
plt.plot(x,data['占有率均值'])
#显示绘图结果
plt.show()
# 三参数各自表达式画图及预测和误差分析
# 速度-时间（流量-时间、占有率-时间类似）
import matplotlib.pyplot as plt
import numpy as np
import math
# x = range(1,dval.shape[0]+1,1)
# x= np.array(x)
Xlabel = range(1,dval.shape[0]+1+200,1)
Xlabel = np.array(Xlabel)
a1 = 13.33
b1 = 64.76
c1 = 18.38
a2 = 59.19
b2 = 286.9
c2 = 84.72
a3 = 37.64
b3 = 139
c3 = 35.25
a4 = 59.91
b4 = 15.44
c4 = 103.8
a5 = 26.85
b5 = 208
c5 = 44.01
a6 = 9.154
b6 = 173.1
c6 = 18.65
f1=[]
for x in range(1,dval.shape[0]+1+200,1):
    f1.append(a1*math.exp(-((x-b1)/c1)**2)+a2*math.exp(-((x-b2)/c2)**2)+a3*math.exp(-((x-b3)/c3)**2) + a4*math.exp(-((x-b4)/c4)**2) + a5*math.exp(-((x-b5)/c5)**2) + a6*math.exp(-((x-b6)/c6)**2))
print(f1)
#开启一个窗口，num 设置子图数量，figsize 设置窗口大小，dpi 设置分辨率
```

```
fig = plt.figure(num=1, figsize=(15, 8),dpi=80)
#直接用 plt.plot 画图，第一个参数是表示横轴的序列，第二个参数是表示纵轴的序列
plt.plot(Xlabel,f1)
# plt.plot(Xlabel,data["速度均值"].values)
#显示绘图结果
plt.show()
#画出散点图
plt.scatter(x, dval, marker = 'x',color = 'red', s = 40 ,label = 'First')
plt.show()
print(max(dval))
```

附录 3

问题一：预处理后补全缺省值后数据集（见“data1_处理后”）

问题二：预处理后补全缺省值后数据集（见“data2_处理后”）

问题三：预处理后补全缺省值后数据集（见“data3_处理后”）