队伍编号	202897
题号	D

基于多种模型对新零售问题进行求解

摘 要

随着电商时代的到来，商品的零售问题成为了一项热点话题。如何对新零售问题进行建模从而了解什么因素在影响销售额，以及预测后续状况，显得非常有必要。基于此，本团队采用多种模型和方法对这个新零售问题进行了研究。在问题一的因素探索中，本团队选择了多元线性回归方法对因素之间的影响关系进行了建模分析与研究。而问题二的类预测问题，我们可以认为它是一个时间序列分析问题。对于这类问题，我们又使用了灰色预测法。问题三的分析预测更加精准，那么我们选用了另一模型：ARIMA 模型进行序列建模。三种方法在此题中的运用将显示出它们的效果差别。当然，后续我们还可以换用其他相关或者类似模型进行建模与优化。此次建模我们的程序使用 python 编写。问题四的建议信将在附录中展示。

关键词：

线性回归，灰色预测，ARIMA，python

目录

（这里开始论文正文）

1

一. 问题重述

随着互联网时代的到来，以电商行业为代表的新零售行业蓬勃发展。对于电商行业而言，顾客的需求变的多样，带来的是商品种类和特征的变多，如何用数学模型精准分析与预测商品的需求成为了一个重要问题。本题依靠产品单款单色（skc），要求分析商品的销售情况与各因素的关系，并依此关系对之后的商品需求进行精准预测。我们需要建立模型分析以下问题

1.通过已有数据，建立模型，分析十一，双十一，元旦和双十二这四个节日中各因素对销售量的关系。因素包含销售特征，库存量和节假日折扣等。（（这段加到问题分析），销售特征包括标价和类别等。那么我们也就是讨论不同日期下的库存，标价，类别和节假日折扣对销售情况的影响。）

2.通过给出的前 21 个月的数据，建立目标小类销量与时间的模型，并对 2019 年 10、11、12 三个月的目标小类的销量进行预测，最后给出预测值与实际值的偏差（MAPE）。

3.结合上述模型，考虑小类中 skc 与小类的差异，建立目标小类中 skc 的预测模型，并对目标小类中 skc 的最后 12 周的数据进行预测，并给出预测值与实际值的偏差(MAPE)。

4.通过前述分析结果，结合实际向企业提出建议。

二. 问题分析

由于实际销售中消费者对产品的需求多样，在此分析问题时将产品差异限于产品销售特征，库存信息与节假日折扣。同时我们结合实际商品销售特征，将销售特征分为由款式，颜色等特征区分的 skc(单款单色)和产品价格特征，即产品原有售价。由于不同 skc 难以分类与定性，我们将每种 skc 看作单独一类

对于问题一，通过初步分析数据，我们将不同节假日中同种 skc 变换的特征，即产品原有标价，库存，折扣作为影响因子序列，将产品销售量作为影响结果，对其进行拟合，选用不同的拟合方式，使拟合较为准确，同时我们注意到有些 skc 有部分数据缺失，对此我们选用个案剔除法与均值替换法

问题二则是一个典型的时间序列分析预测问题，我们需要通过已有 21 个月的目标小类销售量与时间的关系，预测 2019 年 10 月之后的 3 个月的销售量。我们需要通过数据分析与整理统计出目标小类，此问题由于只有十个类的数据，所需分析的数据较小。通过灰色预测分析即可以，被广泛应用于时间序列的分析与计算。这里我们便选用了灰色预测。

问题三：我们对数据进行提取后使用第二问的灰色预测进行尝试，但发现效果不好。而且数据中含有太多 0 项，故我们换用了另一方法，ARIMA 模型。并且对数据进行了清洗，将不易预测的数据（容易造成误差较大的数据）进行了排除。随后编写相关程序对问题进行求解。

三. 符号规定

y	销量
x_1, x_2, x_3	分别代指标价，库存，折扣
a, b,c,d	线性回归方程的系数与常数项
MSE	均方误差
x(0,n)/ $x^{(0)}(n)$	原始数据
X(1,n)/ $x^{(1)}(n)$	求和以后的数据
Y(n)	X(1,n)与 x(1,n+1)的算术平均值
$d(x(1,n))$	求微分，结果等价于 $x(0,n)$

四. 模型假设

1. 问题一的有关因素可以用线性模型描述
2. 问题三若有含 0 项较多的数据允许被清洗
3. 问题二三中时间序列不受其他因素影响，相对稳定发展
4. 由于实际销售中消费者对产品的需求多样，在此分析问题时将产品差异限于产品销售特征，库存信息与节假日折扣。由于不同skc难以分类与定性，我们将每种skc看作单独一类

五. 模型建立

1. 问题一

第一问同时分析多元变量对某元素的影响，属于多对一的分析问题。由于时间限制，这里我们没有选用神经网络进行分析。简单起见，我们仅使用了线性回归模型（更确切的说，是多元线性回归）。

我们针对每一个类，分析三个因素（库存，标价，折扣）对这个 x 销量影响。那么，这个函数的一般形式为：

$$1.\quad y = ax_1 + bx_2 + cx_3 + d \text{ 。}$$

简单起见，这里我们假设 d=0。

我们求解多元回归通常有两种方法，最小二乘法和梯度下降法。相比于最小二乘法，梯度下降更具备通用性。但在使用梯度下降法时，我们要注意一个问题，就是学习率不要设置太大，否则会导致震荡从而无法收敛。而且要小心，梯度下降容易导致陷入

局部最小值。

衡量模型的好坏可以用 MSE 衡量。MSE 的表达形式如下：

$$2.\ MSE=\frac{1}{n}\sum_{i=0}^{n}(y_{实际}-y_{预测})^2$$

求解 MSE 的最小值就是回归问题的核心。对于函数而言，便是求得该函数对所有参数（变量）的偏导，每次更新这些参数，直到到达最低点为止，注意这些参数必须在每一轮一起更新，而不是一个一个更新。【1】

需要注意的一个麻烦问题是：所有的八个类中，有一个类只有一个对象，且存在缺失现象，给我们的分析带来了一定难度。我们使用了回归函数法，对剩余的值进行了补齐，得到了关系模型。

2. 问题二

第二问预测最后三个月的销售情况属于时间序列建模分析问题。对于此问题，我团队决定选用灰色预测算法进行分析求解。灰色系统理论是基于关联空间和光滑离散函数等概念定义灰导数与灰微分方程。进而利用离散数据列建立微分方程的动态模型。【2】在这个问题中，我们是这样利用灰色预测模型的：

首先，对于任意一类的数据，我们把它记作原始数据列 $x(0,n)$，灰色预测的第一步就是对原始数列求和，获得 $x(1,n)$ 。

$$3.\ x(1,n)=\sum_{i=1}^{n}x(0,i)$$

我们把这一序列记为 B 序列。另外，我们规定 $y(n)=\frac{x(1,n)+x(1,n+1)}{2}$，这样我们便得到序列 Y。那么，我们的微分方程就可以列出了，这就是我们要求解的核心模型：

$$4.\ d(x(1,n))+ay(n)=b\ ,$$

其中一阶求和列的微分，利用招差法，不难看出，就是原数列。而这一形式，也与问题一中的这个线性回归模型又有了关联。求解这个微分方程，它的形式不难发现，为：

$$5.\quad \hat{x}^{(1)}(t)=(x^{(1)}(0)-\frac{b}{a})e^{-at}+\frac{b}{a}$$ 【3】。

利用这一方程，可以求解预测的 $x(1,n+1)$，再由 $x(0,n+1)=x(1,n+1)-x(1,n)$ 求解得来。

3. 问题三

这一问题较之于第二问则更进一步。一个是时间方面，精确到了周；一个是目标方面，从类到了 skc；一个是数量方面，数据从十条变成了成千上万条。

开始我们尝试着使用灰色预测但是效果并不太理想。一是误差太大，另一方面，数据中含 0 项实在太多。所以我们果断选择了另一个处理时间序列的方法：ARIMA 模型。并设计相关程序对数据进行批量处理。

ARIMA 模型相比于前面两种方法则显得更加难以理解。在获得时间序列（也就是这里题目所给的数据）以后，要想使用 ARIMA 模型，需要将数据进行 N 阶差分化为平稳时间序列。平稳时间序列粗略地讲，一个时间序列，如果均值没有系统的变化（无趋势）、方差没有系统变化，且严格消除了周期性变化，就称之是平稳的。[4] 那么，使用 ARIMA 的第一个步骤就是进行差分。这里我们在开始的一版代码中比较了一阶和二阶差分的结果效果都还算不错。

然后就是利用 ACF 或 PACF 确定 ARIMA 的阶数 p,q。最后就可以模拟与运作了。ACF 的相关公式形式为

$$ACF(k)=\frac{\operatorname{cov}(y(t),y(t-k))}{\operatorname{var}(y(t))}$$ [5]，

6.

其中 cov,var 分别代表协方差和方差，y(t)表示序列。而 PACF 则在 ACF 的基础上进行修改，使其表现出两个变量之间的严格相关性。

当然，实际编写 python 程序时有相应的库和工具包可以引用。

4. 问题四

问题四的推荐信将在附录中给出

六. 模型求解

1. 问题一

我们首先对问题一中所用数据进行整合，得出如下表格

5

标签	定性自变量	定量自变	十一				双十一				双十二				元旦			
目标skc	类别	标价	十一库存	十一件数	十一销售额	十一单价	双十一库	双十一件	双十一销	双十一单价	双十二库	双十二件	双十二销	双十二单价	元旦库存	元旦件数	元旦销售	元旦单价
100572118316		98.75	630	23	2060	89.5652174	482	10	843.75	84.375	367	7	501.25	71.60714286	365	16	1413.75	88.359375
1.02572E+11		98.75	410	14	1218.75	87.0535714	250	8	695	86.875	178	3	206.25	68.75	218	4	375	93.75
1.02573E+11		73.75	472	13	855	65.7692308	373	12	810	67.5	336	4	227.5	56.875	325	5	337.5	67.5
1.02574E+11		173.75	417	5	721.25	144.25	319	5	767.5	153.5	266	2	242.5	121.25	220	2	347.5	173.75
1.96574E+11		123.75	362	10	1067.5	106.75	265	4	411.25	110.3125	226	3	258.75	86.25	213			
2.96572E+11			407	14	931.25	66.5178571	276	13	891.25	68.55769231	309	15	762.5	50.83333333	364	16	1052.5	65.78125
2.96573E+11		98.75	171	11	965	87.7272727	297	1	98.75	98.75	322				311	7	601.25	85.89285714
3.02573E+11	27050401	73.75	652	13	871.25	67.0192308	561	10	646.25	64.625	413	8	432.5	54.0625	430	16	1038.75	64.921875
3.90572E+11		223.75	329	6	1125	187.5	231	2	381.25	190.625	258	6	960	160	243	6	1176.25	196.0416667
3.91572E+11			467	17	1113.75	65.5147059	445	11	716.25	65.11363636	296	13	726.25	55.86538462	351	16	995	62.1875
3.96572E+11		123.75	200	6	651.25	108.541667	128	1	105	105	91	5	456.25	91.25	69	2	191.25	95.625
3.96574E+11		173.75	395	12	1816.25	151.354167	348	1	147.5	147.5	332	2	257.5	128.75	297	1	132.5	132.5
3.96574E+11		198.75	609	20	3433.75	171.6875	413	4	696.25	174.0625	419	3	438.75	146.25	445	5	875	175
4.02574E+11		161.25	275	13	1876.25	144.326923	255	5	708.75	141.75	254	1	15	15	209	8	1133.75	141.71875
4.08573E+11			852	25	2157.5	86.3	684	1	85	85	621	7	601.25	85.89285714	595	6	532.5	88.75
4.33572E+11			366	17	1456.25	85.6617647	228	7	617.5	88.21428571	353	16	1022.5	63.90625	289	15	1263.75	84.25
4.90572E+11			363	19	1987.5	104.605263	228	3	321.25	107.0833333	273	7	566.25	80.89285714	270	8	900	112.5
4.96572E+11		248.75	311	7	1477.5	211.071429	282	2	445	222.5	229	1	173.75	17.75	202	4	900	225
4.96572E+11		123.75	315	12	1290	107.5	221	3	256.25	85.41666667	226	3	296.25	98.75	270	3	328.75	109.5833333
4.96572E+11	27060804	123.75	337	11	1166.25	106.022727	244	1	115.86	115.86	218	3	281.25	93.75	192	1	108.75	108.75
4.96573E+11		123.75	266	13	1450	111.538462	206	6	693.75	115.625	263	6	630	105	300	7	762.5	108.9285714
4.99572E+11		98.75	567	10	892.5	89.25	413	1	83.79	83.79	426	5	393.75	78.75	436	2	147.5	73.75
5.02574E+11		123.75	215	7	751.25	107.321429	122	1	113.75	113.75	166	3	283.75	94.58333333	200	7	793.75	113.3928571
5.96572E+11		98.75	114	6	518.75	86.4583333	187	6	515	85.83333333	218	8	616.25	77.03125	163	9	777.5	86.38888889
5.96572E+11	27071209	98.75	281	15	1313.75	87.5833333	203	4	378.75	94.6875	226	9	618.75	68.75	265	7	560	80
5.96574E+11		123.75	303	12	1322.5	110.208333	321	3	321.25	107.0833333	321	7	603.75	86.25	322	10	1132.5	113.25
6.02573E+11		198.75	300	6	1046.25	174.375	278	2	353.75	176.875	235	1	138.75	138.75	204	1	160	160
6.08573E+11		73.75	751	21	1390	66.1904762	585	4	257.5	64.375	578	12	691.25	57.60416667	608	13	868.75	66.82692308

图一：问题一整合后的数据

这些数据存在一定程度的缺失现象，所以我们对数据又进行了进一步处理。能够补齐的补齐。

随后我们编写了相关的 Python 程序，进行求解结果如图：

```
[Running] python -u "c:\Users\malia\Documents\Tencent Files\2793055528\FileRecv\问题一.py"
linear fit [-0.0126955   0.02726416 -3.1486769 ]
[15.62991022  9.5516786  11.59166862  8.62871406  7.86601052  3.05702655
 16.55257315  5.61916806  3.49480429  8.15764352 13.65190494  5.1200418
 11.42929049  5.18371981  8.96639976  6.12445425  5.31203119  6.84377346
 13.96931592  2.99125677  1.44167145  6.80638345  8.34575619  4.52443994
  7.88680537  2.64277891  7.50400306  4.09502335  3.63648642  7.52537734
  9.48326636  3.2964248   0.08304472  6.03037169  8.06872441  2.0221693
  8.36642745  4.53047852  7.6577203   3.79227098  4.23619691  6.8155173
 10.41038632  3.39465157 -0.40544985  5.14408333  9.2330614   3.26967859]

[Done] exited with code=0 in 0.505 seconds

[Running] python -u "c:\Users\malia\Documents\Tencent Files\2793055528\FileRecv\问题一.py"
linear fit [ 0.01122731  0.01576259 -6.18935215]
[6.75744657 5.54185442 5.81474493 4.98679835 9.45065657 3.95666293
 6.58469715 2.95567253 4.84083449 4.23010591 6.68100049 2.8122669
 0.65472745 3.69818589 3.32517764 4.59716221 6.57011614 2.54720034
 5.38589412 4.93673473 3.66557319 5.37686464 6.41426341 4.02388631]

[Done] exited with code=0 in 0.517 seconds

[Running] python -u "c:\Users\malia\Documents\Tencent Files\2793055528\FileRecv\问题一.py"
linear fit [-0.04435933  0.04582866  4.34867183]
[ 1.38527428  8.98911887  8.87248248  5.46550994  4.75828979  5.10163581
  9.80721333  4.40257912  6.56659724  7.29790945 10.53931296  3.26612587
  3.63393682  8.58980893  9.6363405   1.38048504]

[Done] exited with code=0 in 0.502 seconds

[Running] python -u "c:\Users\malia\Documents\Tencent Files\2793055528\FileRecv\问题一.py"
linear fit [-0.00817726  0.02634195  1.02647794]
[10.49822715  5.6917401   2.27809383 14.49097053  4.57650136  9.86933363
  4.04978488  6.13784478 17.52243109  6.66838838  0.4278759   4.74815418
  6.54406637  3.57635593  1.38428789  8.81017796  5.52352667  8.0977921
  2.43349313  5.53050302 11.17190152  4.35665368  1.78273371  4.17013095
  4.45458109  5.06233891  0.36441501  6.01210483  5.33142407  8.37798412
  5.34610939 11.86216155  5.00076081  2.43907823  3.42775049  4.48187594
  5.13741229  8.35535393  4.11308283  8.12811265  4.58071594 15.91483684
```

图二：问题二的运行截图

观察利用预测函数的预测值，发现结果还是很不错的

数据集中一共八类，运用相应方法求解函数关系如下：

其中 u 表示销售量，x, y, z 分别代表单价，库存和折扣。

类名	27050401	27060804	27070809	27112849	27161944	27196225	27237961	27092025
函	-0.0126	0.01122	-0.4435	-0.0081	-0.0	-0.0068	-0.0063	-0.0149

数u	955x+0.02726416y-3.1486769z	731x+0.01576259y-6.1893522z	933x+0.04582742 8y+4.34867183z	7726x+0.02634195y+1.02647794	075x+0.0285y+0.4202z	2118x+0.03168034y+0.00139296z	672x+0.02325746y-4.8574421z	6874x+0.02101431y+0.04200356z

2. 问题二

第二问我们选用的是灰色预测法进行求解。灰色预测法是一种对含有不确定因素的系统进行预测的方法。灰色预测是对既含有已知信息又含有不确定信息的系统进行预则，就是对在一定范围内变化的、与时间有关的灰色过程进行预测。[6]我们利用灰色预测的基本原理设计了算法与 python 程序。这一程序包括：导入序列，求相关数据矩阵，固定预测年数为三年，计算输入数据误差和预测值，以及作图功能。代码放在附录中，计算结果如下所示：

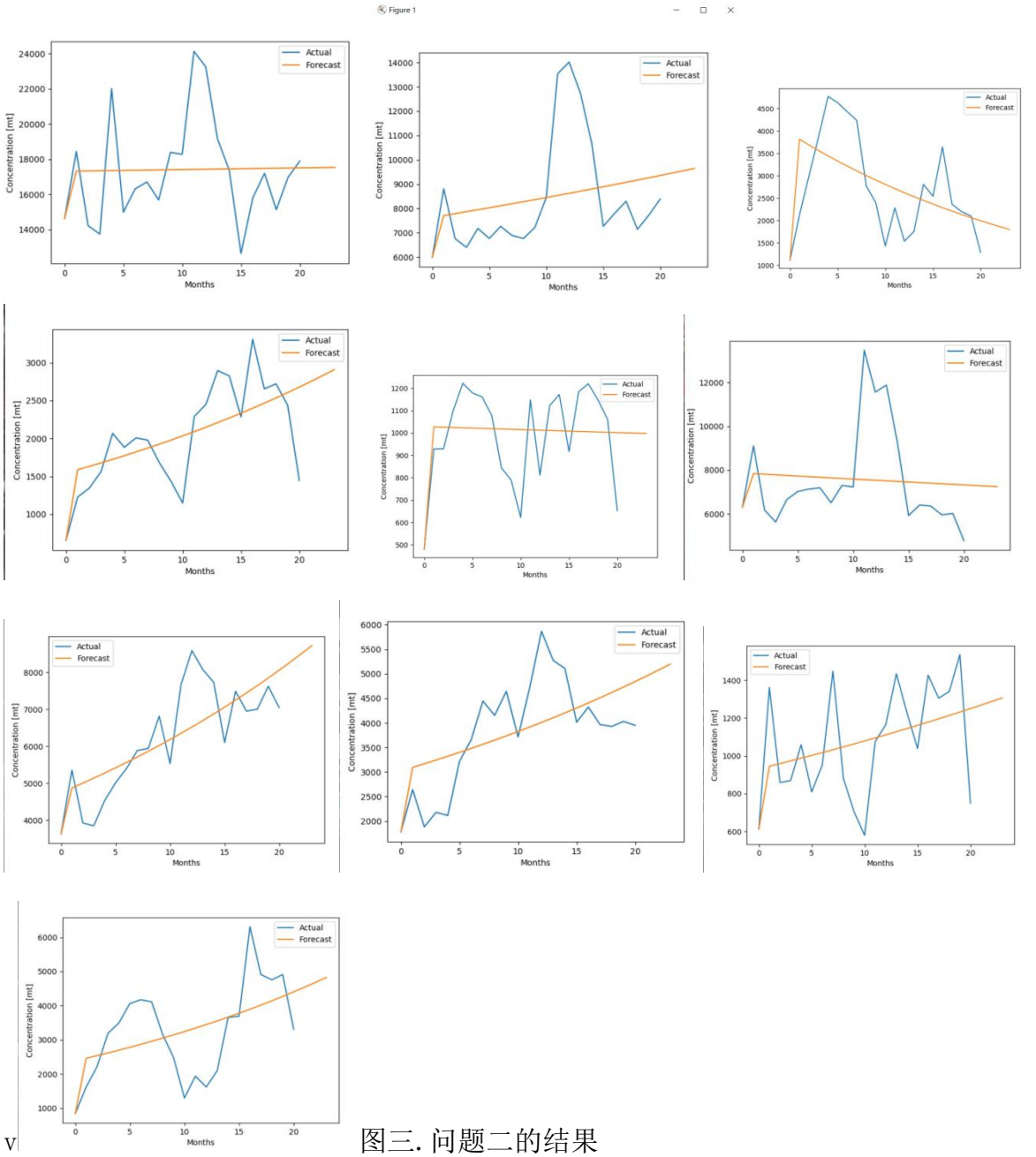
月份\类名	27050401	27060804	27071209	27092025	27102436	27112849	27164944	27196225	27206656	27217089
10	17519.27658	9452.018553	1929.253072	2752.682872	1000.29424	7293.872989	8272.742848	4958.242289	1268.785096	4542.110954
11	17528.83077	9549.00316	1864.651779	2819.502485	998.968574 1	7267.860775	8494.580871	5076.547454	1287.615945	4683.160399
12	17538.39016	9646.98292	1802.213668	2908.465916	997.6446653	7241.941329	8722.367599	5197.675416	1306.726276	4828.589955
MAPE	0.05748276 3	0.13237980 2	1.621199117	1.417881504	1.163131407	0.214250974	0.278284042	0.727740619	1.653959817	1.690923468

表 1.问题二的结果

我们还绘制了预测值与精确值的图像

7

按顺序排列如下：



V 图三. 问题二的结果

3. 问题三

我们利用 ARIMA 模型对第三问进行了建模分析，并设计了相关程序。程序读取清洗后的数据 1.csv，将结果 s 写入 2.csv。存储路径随电脑而定。对于这一程序，我们还有一个细化的分析程序，这两个程序都将在附录当中展示。主程序包括：数据导入和预处理，一阶-二阶差分，ARIMA 模型拟合，以及最后的结果输出。由于数据太多，我们把结果放到了附件中，这里展示一张截图。各个星期的 MAPE 如下所示：

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	27050401	8770.7903	8412.3252	8396.7736	7859.737	7861.3558	7530.4992	7607.9275	7375.5493	7418.6598	7614.6008	7648.1595	7818.0684
2	1.006E+11	0.27323	0.5186068	0.5312098	0.7113426	0.7447556	0.735716	0.7897845	0.8078525	0.8635731	0.9161849	0.9794644	1.0366149
3	1.006E+11	1.5552469	1.7664837	1.867704	1.9924336	2.1484453	2.2029123	2.3874329	2.3797073	2.4109609	2.429351	2.4581049	2.461889
4	1.006E+11	79.743927	81.197195	80.973712	68.996104	63.18683	58.852565	61.046133	62.524725	66.226366	68.96979	67.912457	67.254324
5	1.006E+11	3.9932523	2.5933788	2.4934041	3.568273	5.7240146	6.9732507	6.160236	5.2968354	5.8974897	7.6740365	9.1786705	9.0471521
6	1.016E+11	15.632076	11.21079	13.786803	15.98255	16.12086	17.671919	15.769458	14.301284	15.305236	13.599138	12.86193	13.424218
7	1.016E+11	0.0647646	0.3675192	0.3556938	0.7170587	0.5782764	0.7678425	0.7537573	0.9462269	1.0922648	1.1117065	1.1784979	1.1572748
8	1.016E+11	1.6302963	0.8725958	1.3207766	0.5311642	1.2990725	0.951375	1.3970897	1.0106589	1.419547	0.9859685	1.4836749	1.0521011
9	1.026E+11	0.8730123	1.2944656	1.2598956	1.2263272	0.999502	0.9921217	1.0369729	0.9632583	0.9692925	0.9903609	1.004784	1.0450498
10	1.026E+11	1.084433	1.2350661	1.5013406	1.3987911	1.6845969	1.7532254	1.6782169	1.8128502	1.7749796	1.8212866	1.758669	1.7976247
11	1.026E+11	1.3529546	1.5460216	1.5875378	1.6196394	1.4854853	1.4013593	1.4073037	1.5059325	1.6237563	1.6599576	1.6698676	1.6585462
12	1.026E+11	1.3457991	1.9627834	1.6926075	1.3796363	1.4094913	1.430445	1.3452822	1.4670412	1.4432289	1.3194297	1.291702	1.3294255
13	1.026E+11	1.1716895	1.472652	1.8739561	2.6447556	2.9991261	3.1461348	3.7321896	4.2460676	4.9429391	5.3436484	5.6263446	6.2091756
14	1.026E+11	1.1610643	1.5324533	1.5113513	1.6835552	1.706418	1.7466054	1.8368098	2.0273744	2.09729	2.1412332	2.2175073	2.234089
15	1.026E+11	6.9813896	7.0341168	6.3082625	8.1193201	6.8579333	7.0786713	7.0950545	7.8640451	7.2115192	7.7638874	7.8230721	8.0876967
16	1.026E+11	1.1440431	0.6912775	0.5726842	0.9803488	0.9830026	1.1213463	1.55908	1.4339432	1.3773959	1.6351664	1.6705483	1.7351426
17	1.026E+11	-0.039103	0.1598726	-0.156664	0.1798044	0.2761905	0.1477697	0.3811594	0.3175088	0.4158134	0.5526449	0.6019069	0.6351136
18	1.026E+11	0.1448655	0.2401726	0.2934341	0.3727619	0.5004363	0.5940354	0.700772	0.7824791	0.8751463	0.9658266	1.0600387	1.1243801
19	1.026E+11	0.5200636	0.7371168	0.6413082	0.6676214	0.3875807	0.7261478	0.8829475	1.1217492	1.0028935	0.8957805	0.9404645	1.0390006
20	1.026E+11	4.5365163	3.388059	4.2433644	4.5463523	4.8319681	4.9433086	4.7440231	6.2749868	5.500565	6.0259331	5.8483887	6.1446382
21	1.026E+11	2.5194916	3.5869208	4.7182024	5.6535929	7.1035951	7.573564	7.8156105	9.0366371	9.8132776	10.118121	10.465688	10.882141
22	1.026E+11	2.2030209	2.7918577	1.1813914	2.7330307	2.2673415	2.1726769	2.6531488	2.1329093	2.4389069	2.445861	2.3504825	2.5147197
23	1.026E+11	2.0037062	2.098154	2.2821991	2.3910838	2.6896478	2.8423447	2.8893053	2.9047883	2.911006	2.9501184	2.9610626	2.9630647
24	1.026E+11	2.0644129	2.481142	2.6653575	2.4230498	2.8293474	3.1188639	2.7362858	2.9123132	2.9387789	2.920147	2.7675574	2.8554822
25	1.026E+11	39.404631	38.158738	36.676566	36.785762	32.381605	33.056963	33.800872	36.230062	35.109969	36.030006	36.098706	36.996402
26	1.026E+11	0.547169	0.6930415	0.9529164	0.9958714	1.2102193	1.2881208	1.2860378	1.2853655	1.282601	1.3136883	1.2756535	1.2694174
27	1.026E+11	4.4786839	4.8454197	5.5543249	5.7602465	5.73186	5.7072702	5.5380168	5.5854961	5.6817807	5.8434902	5.8652404	5.8292827
28	1.026E+11	10.121547	9.4840913	8.664428	7.4597098	5.4553003	5.0759503	5.9536371	6.9859377	7.3741997	7.3142108	6.7999936	6.1027425
29	1.026E+11	0.3599396	-0.035183	0.569484	0.6474665	0.7892836	1.092139	1.1488534	1.264911	1.3355272	1.2714172	1.2667769	1.2421728
30	1.026E+11	0.3772276	0.8101373	0.7052645	0.6450306	1.2372474	1.0776394	1.2072925	1.1830113	1.2880212	1.1979521	1.1893608	1.0846159

图四：问题三的结果

MAPE：0.5907694，0.5962351，0.6678765，0.756654，0.7834612，0.8046413，0.8583756，0.8199445，0.8518542，0.8282496，0.820861，0.8003677

七.结果分析

1. 问题一

使用多元线性回归，预测值与真实值也在程序中表现出来。我们选取一组数据计算方差：

选用 20792025 类的数据，计算方差得 13.0817 。而原数据为 21，4，4，2，7，13，7，8，这个方差值显得似乎有些大。我们再选一组数据：

选用 27050401 类的数据，计算方差得 15.8031 .而原数据为

[23,14,13,5,10,11,13,6,6,12,20,13,10,8,12,5,4,1,10,2,1,1,4,5,7,3,4,2,3,0,8,6,5,2,3,1,16,4,5,2,0,7,16,6,2,1,5,8]说明这一结果似乎还不错。这也说明，在数据较多且呈良好线性状态下的情况下线性回归更有效

2. 问题二

相对而言，问题二的数据我们也可以从图 2 中看出，无论是线性关系还是其他函数关系都不太明显。而且预测图像和原始图像差别还是有点大的。这说明这一序列具有高度不确定性。MAPE 值中已经有超过 1 的项，说明预测效果仍然没有想象中那么理想，但

又有一定参考价值。

3. 问题三

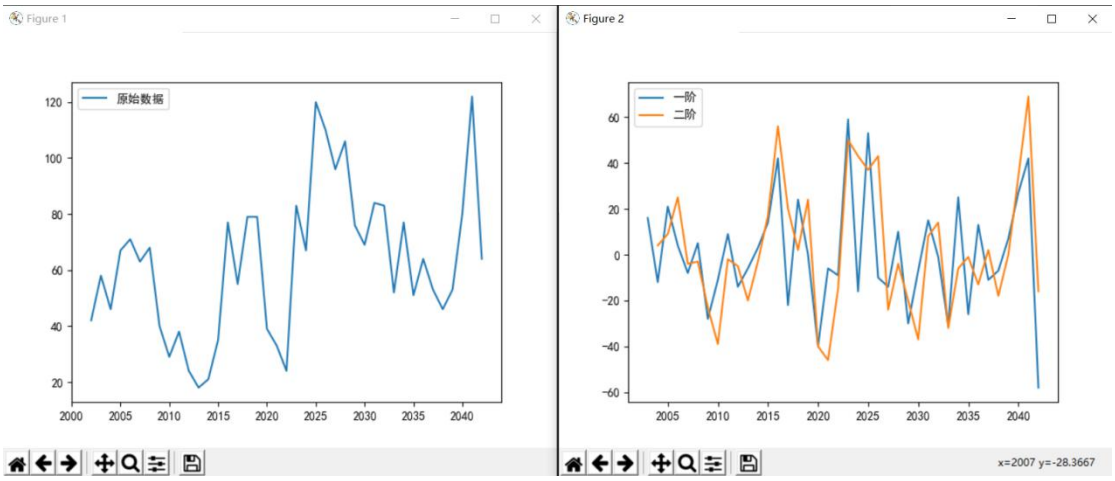
问题三的 MAPE 值总体表现都还比较不错，在上千条数据中能够保持相当小的误差，我们认为使用 ARIMA 模型进行批量处理方法是成功的。

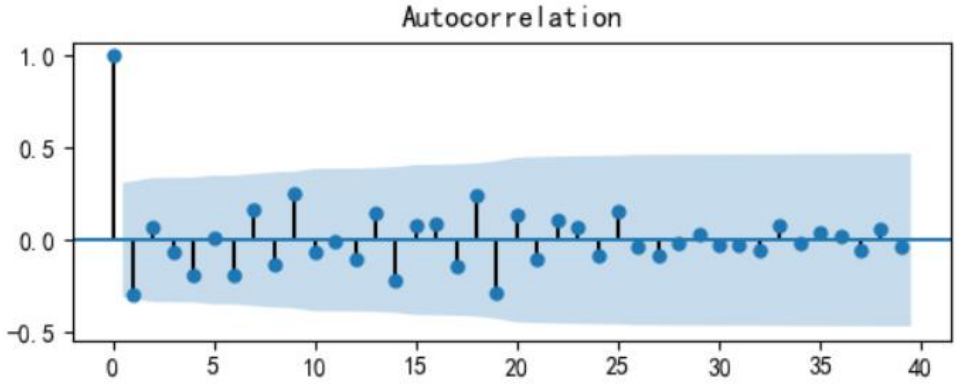
对于问题三，我们抽取了一组数据并编写了相应 python 程序分析误差：

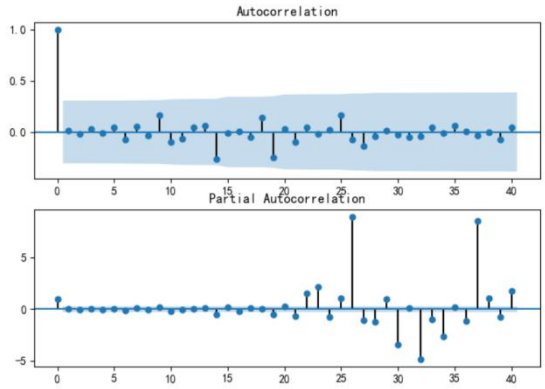
分析

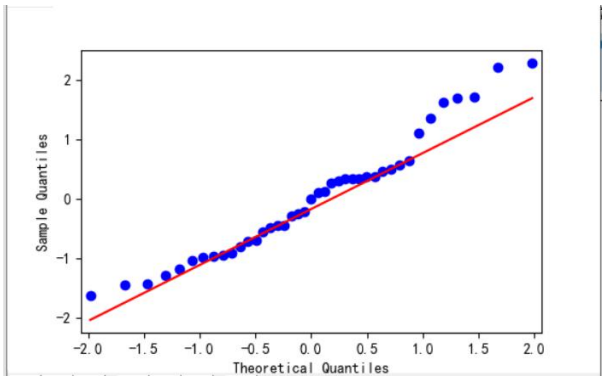
42,58,46,67,71,63,68,40,29,38,24,18,21,35,77,55,79,79,39,33,24,83,67,120,110,96,106,76,69,

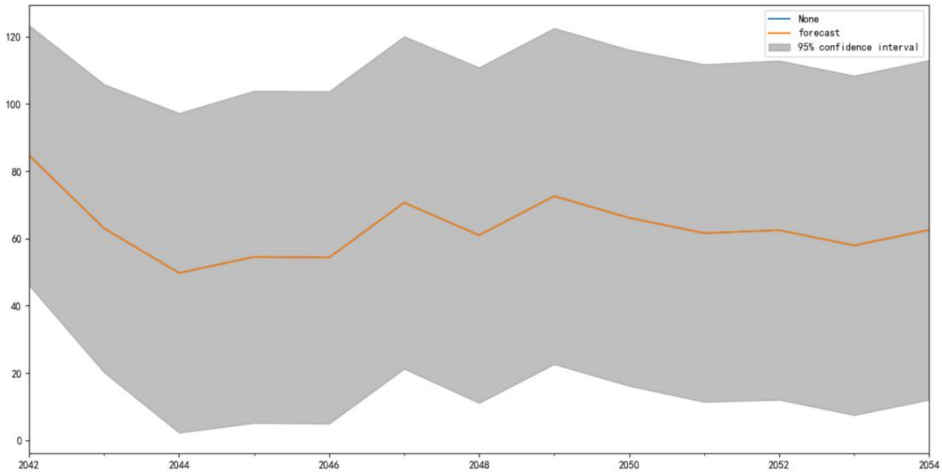
84,83,52,77,51,64,53,46,53,80,122,64 这组数据的误差情况，绘制曲线如图：











图五：问题三数据的分析，图像依次为原始数据，一二阶差分图像，ACF,PACF 图，残差检验图，以及预测图。

看的出来，预测效果良好

八.模型优缺点

优点：

对于问题一，我们使用的是多元线性回归。这一模型是时间序列分析中应该是最容易的一种。执行起来方便。

对于问题二，我们使用的是灰色预测算法。这一算法对时间序列分析有很强大的作用。开始使用线性回归的效果并不好，但是我们换用这一方法以后精确度明显提高了很多。

问题三的这个 ARIMA 模型，也是一种非常强大的模型。对数据的批量处理，还有精确度，相比于灰色预测，就更近了一步。

缺点：

1. 问题一的线性回归模型：有很多情况下线性相关程度不强，这时再用误差较大，普适性一般

2. 问题二数据本身不确定性太强，导致 j 预测效果仍然是不尽人意

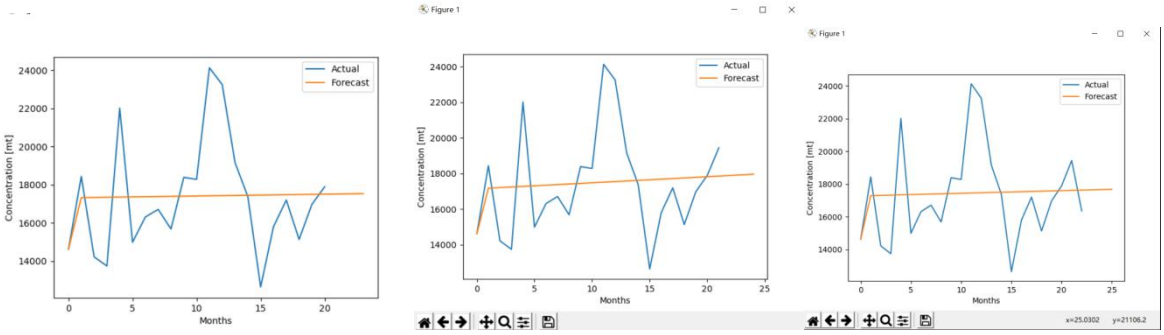
3. 问题三 z 在数据清洗这一环节可能清洗了很多本可以预测好的数据，也可能保留了效果不太好的数据。另一方面，处理清洗好的数据，计算机运行了 5866.126 秒才处理完毕，运算速度和效率应该设计算法大大强化。

九. 模型的改进

在经历几天的尝试与学习后，我们发现，ARIMA 模型在处理时间序列分析问题时有很强的功能与很好的效果。所以，我们完全可以将问题二也用 ARIMA 实现。

另外，就算不使用 ARIMA，问题二仍然有优化策略。即：每次仅预测一个值，然后将对应真实值加入后再预测下一个值。我们也对其中一组数据进行了相应操作：

运行结果如下：（图六）



17519.2765796　　17891.059873　　17649.0197879

结果看上去还是不错的

参考文献

【 1 】 braveryCHR ， 线 性 回 归 与 梯 度 下 降 ， https://blog.csdn.net/bravery_again/article/details/81076621?ops_request_misc=%257B%2522request%255Fid%2522%253A%252215903239461972686905332 2%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=159032394619726869053322&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v25-3-81076621.first_rank_v2_rank_v25&utm_term=%E6%A2%AF%E5%BA%A6%E4%B8%8B%E9%99%8D%E8%A7%A3%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92 ， 2020 年 5 月 23 日

【 2 】 i-kernel, 灰 色 预 测 — — 从 模 型 到 实 例 ， https://blog.csdn.net/weixin_39212776/article/details/82221230?ops_request_misc=&request_id=&biz_id=102&utm_term=%E7%81%B0%E8%89%B2%E9%A2%84%E6%B5%8Bpython

12
&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-2-82221230
2020 年 5 月 23
【3】司守奎，孙玺菁《数学建模算法与应用》北京，国防工业出版社，2011 年
【4】百度百科，平稳时间序列，https://baike.baidu.com/item/平稳时间序列/12722171?fr=aladdin，2020 年 5 月 23
【5】TJUC，arima，https://blog.csdn.net/TU_JCN/article/details/88130820?ops_request_misc=&request_id=&biz_id=102&utm_term=arima%20&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-88130820，2020 年 5 月 24 日
【6】mangoit，灰色预测，https://blog.csdn.net/qq_36666756/article/details/82080174?ops_request_misc=%257B%2522request%255Fid%2522%253A%252215902903061972484503403 1%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=159029030619724845034031&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v25-3-82080174.first_rank_v2_rank_v25&utm_term=%E7%81%B0%E8%89%B2%E9%A2%84%E6%B5%8B，2020 年 5 月 24 日

附录

一．建议信：

尊敬的某企业:

你好

我们分析了贵公司生产的部分产品在华东区内的相关数据，得到了一些有用的数据，希望可以帮到贵公司。

我们利用多元回归分析，对几个类中销量与库存，标价，折扣的关系进行建模，得到如下表达式：根据此，我们可以观察到，这里 x,y,z 分别代表标价，库存，折扣。标价的系数多为负数，而库存系数为正。这也就是说，标价低一点，内存多一点就会使销量增加。另外，折扣系数的绝对值是三项最大，折扣对销量的影响最大，应该合理把控好。

名	27050401	27060804	27070809	27112849	27161944	27196225	27237961	27092025
数u	-0.0126955x+0.02726416y-3.1486769z	0.01122731x+0.01576259y-6.1893522z	-0.4435933x+0.045827428y+4.34867183z	-0.00817726x+0.02634195y+1.02647794	-0.075x+0.0285y+0.4202z	-0.00682118x+0.03168034y+0.00139296z	-0.00063672x+0.02325746y-4.8574421z	-0.01496874x+0.02101431y+0.04200356z

我们通过了灰色预测和时间序列模型(ARIMA)分析预测了热卖小类和热卖 skc，通过小类 27050401 的 10 月前的历史数据，预测 10 月到 12 月的销售量为 17519，17528，17538，这与实际的误差是比较小的，同时我们又预测了部分 skc，如 skc100572118316，由历史数据对 10 月 1 号后的 12 周销量预测结果为 79.7439271681.19719465　　80.97371244　　68.99610373　　63.18682979　　58.85256458　　61.04613288　　62.52472511　　66.22636585　　68.96979042　　67.91245653　　67.25432409,与实际相差也在可以接受的范围内。

灰色预测法是一种对含有不确定因素的系统进行预测的方法。灰色预测是对既含有已知信息又含有不确定信息的系统进行预则，就是对在一定范围内变化的、与时间有关的灰色过程进行预测。而 ARIMA 模型是时间序列预测分析方法之一，非平稳时间序列，在消去其局部水平或者趋势之后，其显示出一定的同质性，也就是说，此时序列的某些部分 与其它部分很相似。这种非平稳时间序列经过差分处理后可以转换为平稳时间序列，借此得以预测数据。

以上两种方法都是数据预测分析的常见方法，所得预测值 MAPE 大部分在 1 以内，说明方案是合理的。贵企业想要采用我们的方法，可以做一些优化。如使用更详细具体的销售信息，对销售信息做更好的预处理，合理选择 ARIMA 模型差分阶数等。通过我们的预测数据，贵企业可以更好的了解市场销售情况，根据市场调整营销策略，以得到更好的销售结果。

二：代码

使用软件：VScode,EXCEL,python IDLE

语言：python3.8.2

14

问题一代码：

```
from scipy.optimize import leastsq
import numpy as np
x1=np.array([[248.75,311,0.151472],[123.75,315,0.131313],[123.75,337,0.14325069],[123.75,266,0.0962
0253], [98.75,567,0.09620253],[123.75,215,0.13275613],[248.75,282,0.10552764],[123.75,221,0.3
0976431], [123.75,244,0.06375758],[123.75,206,0.065656],[98.75,413,0.1514937],[123.75,122,0.0
80808], [248.75,229,0.92864322],[123.75,226,0.2025317],[123.75,218,0.242424],[123.75,263,0.15
1515], [98.75,426,0.2025165],[123.75,166,0.23569024],[248.75,202,0.09547739],[123.75,270,0.11
447811], [123.75,192,0.12121212],[123.75,300,0.11976912], [98.75,436,0.25316456],[123.7
5,200,0.08369408]])
y1=np.array([7,12,11,13,10,7,2,3,1,6,1,1,1,3,3,6,5,3,4,3,1,7,2,7])
x2=np.array([[98.75, 114, 0.124472574],[98.75, 281, 0.113080169],[123.75, 303, 0.109427609],[
198.75, 300, 0.122641509],[98.75, 187, 0.130801688],[98.75, 203, 0.041139241],[123.75, 321,
 0.134680135],[198.75,278,0.110062893],[98.75,218,0.219936709],[98.75,226, 0.303797468],[123.75,3
21,0.303030303],[198.75,235,0.301886792],[98.75,163,0.125175809],[98.75, 265, 0.189873418],[123.7
5, 322,0.084848485],[198.75,204,0.194968553]])
y2=np.array([6,15,12,6,6,4,3,2,8,9,7,1,9,7,10,1])
x3=np.array([[73.75,416,0.1393597],[223.75,280,0.141899441],[161.25, 131,0.142118863],[73.75, 569,
 0.102754237],[123.75, 207, 0.132154882],[161.25, 422, 0.069767442],[161.25, 199, 0.1230620
16],[173.75, 281, 0.152517986],[173.75, 714, 0.131594724],[123.75, 318, -0.678451178], [
248.75, 87, 0.165829146],[248.75, 242, 0.396984925],[73.75, 265, 0.162227603],[223.75, 200, 0.
134078212],[161.25, 87, 0.400516796],[73.75, 355, 0.060263653],[123.75, 242, 0.156565657],[16
1.25, 355, 0.063307494],[161.25, 137, 0.139534884],[173.75, 258, 0.151079137], [173.75,
 474, 0.103866906],[123.75, 241, -0.954545455],[248.75, 140, 0.125628141],[248.75, 204, 0.809
045226],[73.75, 192, 0],[223.75, 251, 0.272944932],[161.25, 58, 0.151162791],[73.75, 241, 0.25
9887006],[123.75, 229, 0.303030303],[161.25, 353, 0.387596899], [173.75, 251, 0.1510791
37],[173.75, 495, 0.237410072],[123.75, 244, -0.404040404],[248.75, 161, 0.226130653],[248.75,
 186, 0.547738693],[73.75, 189, 0.103578154],[223.75, 259, 0.140782123],[73.75, 337, 0.0790
96045],[123.75, 187, 0.193939394],[161.25, 356, 0.067183463], [173.75,222,0.149640288],[1
73.75,654, 0.105215827],[123.75, 219, -0.75],[248.75, 198, 0.153266332],[248.75, 169, 0.4556113
9]])
y3=np.array([9,10,3,16,12,3,8,5,24,6,5,3,7,1,3,9,2,6,1,1,16,2,8,1,1,7,2,9,2,1,2,12,3,2,1,9,5,12,5,3,5,16,4,4,
3])
x0 = np.array([[98.75,630,0.093], [98.75,410,0.1184448], [73.75,472,0.108210382], [173.75,417,0.169784
173], [123.75,362,0.13737374], [98.75,171,0.11162255], [73.75 ,652,0.091264668], [223.75,329,
0.16201117], [123.75,200,0.12289562], [173.75,395,0.12889688], [198.75,609,0.136163522], [16
1.25,275,0.10494931], [98.75,482,0.14556962], [98.75,250,0.12025316], [73.75,373,0.08474576],
 [173.75,319,0.116546763], [123.75,265,0.10858586], [98.75,297,0], [73.75,561,0.12372881], [223.75,23
1,0.14804469], [123.75,128,0.15151515], [173.75,348,0.15107914], [198.75,413,0.12421384], [1
61.25,255,0.12093023], [98.75,367,0.27486433], [98.75,178,0.30379747],[73.75,336,0.22881355
9],[173.75,266,0.30215827],[123.75,226,0.3030303],[98.75,322,0],[73.75,413,0.26694915], [223.
75,258,0.284916201],[123.75,91,0.26262626],[173.75,332,0.25899281],[198.75,419,0.26415094],[161.25,
254,0.90697674],[98.75,365,0.10522152], [98.75,218,0.05063291],[73.75,325,0.08474576],[173.
```

```
15
75,220,0],[123.75,213,0],[98.75,311,0.13019892],[73.75,430,0.11970339], [223.75,243,0.123836
13],[123.75,69,0.22727273],[173.75,297,0.23741007],[198.75,445,0.11949686],[161.25,209,0.12112403]]
)
y0 = np.array([23,14,13,5,10,11,13,6,6,12,20,13,10,8,12,5,4,1,10,2,1,1,4,5,7,3,4,2,3,0,8,6,5,2,3,1,16,4,5,2,
0,7,16,6,2,1,5,8])
x4=np.array([[123.75,460,0.1338],[161.25,275,0.124],[123.75,200,0.1248],[123.75,258,0.5774],[161.25,19
2,0],[123.75,256,0.312],[123.75,162,0.281],[123.75,275,0.303],[161.25,154,0.10078],[123.75,274,133.84]]
)
y4=np.array([12,9,8,12,1,7,9,5,1,8])
x5=np.array([[123.75,632,0.123334],[198.75,531,0.4804],[123.75,502,0.303],[198.75,396,0.3019],[123.75,
476,0.1313],[198.75,400,0.1258]])
y5=np.array([19,13,1,8,7,5])
x6=np.array([[73.75,751,0.1025],[223.75,258,0.1229],[73.75,585,0.87899],[223.75,242,0.74477],[148,487,
0.75],[73.75,608,0.50129162],[148,554,0.3452381],[223,0.75,269]])
y6=np.array([21,4,4,2,7,13,7,8])
def main():
# data provided
x=x6
y=y6
# here, create lambda functions for Line fit
# tpl is a tuple that contains the parameters of the fit
funcLine=lambda tpl,x: np.dot(x, tpl)
# func is going to be a placeholder for funcLine,funcQuad or whatever
# function we would like to fit
func = funcLine
# ErrorFunc is the diference between the func and the y "experimental" data
ErrorFunc = lambda tpl, x, y: func(tpl, x)-y
#tplInitial contains the "first guess" of the parameters
tplInitial=[1.0,1.0, 1.0]
# leastsq finds the set of parameters in the tuple tpl that minimizes
# ErrorFunc=yfit-yExperimental
tplFinal, success = leastsq(ErrorFunc, tplInitial, args=(x, y))
print('linear fit', tplFinal)    print(funcLine(tplFinal, x))
if __name__=="__main__":
main()
```

问题二代码：

```
import numpy as np
import matplotlib.pyplot as plt
# 导 入 时 间 序 列
X=[[14629,18440,14223,13742,22020,14989,16313,16711,15684,18390,18285,24136,23254,19164,17378,
12646,15800,17201,15136,16976,17899], [5994,8812,6755,6396,7179,6765,7260,6884,6764,7217,8450,
13542,14020,12741,10661,7262,7798,8295,7143,7723,8387], [1119,2126,3000,3868,4768,4630,4429,42
```

```
16
36,2773,2398,1430,2283,1539,1754,2807,2542,3640,2363,2200,2100,1288], [656,1226,1345,1560,2067,1883,2007,1978,1681,1435,1148,2291,2450,2896,2825,2284,3309,2654,2721,2439,1445], [480,928,929,1099,1222,1179,1161,1078,844,790,622,1148,811,1122,1172,917,1184,1220,1148,1061,652], [6303,9103,6169,5622,6649,7017,7131,7189,6505,7301,7223,13474,11562,11880,9240,5921,6399,6352,5953,6012,4775], [3632,5356,3926,3847,4532,5014,5397,5887,5940,6814,5532,7666,8587,8072,7726,6100,7488,6950,7002,7623,7042], [1783,2642,1883,2179,2114,3217,3658,4446,4153,4644,3715,4725,5867,5272,5108,4013,4323,3965,3928,4030,3950], [612,1362,859,869,1059,809,952,1448,879,704,579,1076,1165,1435,1228,1039,1427,1305,1341,1536,750], [844,1622,2218,3195,3497,4064,4175,4113,3163,2486,1301,1941,1622,2100,3657,3693,6311,4913,4756,4913,3312]]
for x in X:
lan = []
for i in range(len(x)):
if i == len(x) - 1:
continue
# 求级比
lan.append(x[i] / x[i + 1])
x_1 = np.cumsum(x)
# 构造数据矩阵 B 及数据向量
B = np.array([-1 / 2 * (x_1[i] + x_1[i + 1]) for i in range(len(x) - 1)])
B = np.mat(np.vstack((B, np.ones((len(x) - 1,)))).T)
Y = np.mat([x[i + 1] for i in range(len(x) - 1)]).T
u = np.dot(np.dot(B.T.dot(B).I, B.T), Y)
[a, b] = [u[0, 0], u[1, 0]]
a_new, b = x[0] - b / a, b / a
# 输入需要预测的年数
year = 3
year += len(x)
x_predict = [x[0]]
x_predict = x_predict + [a_new * (np.exp(-a * i) - np.exp(-a * (i - 1))) for i in range(1, year)]
#计算相对误差，如果较小，则可以使用
print("The relative error is calculated as follows:")
print((np.array(x_predict[:len(x)])-np.array(x[:len(x)]))/np.array(x[:len(x)]))
print("predictions below:")
print(x_predict[len(x):])
#注意这里数字要根据时间序列的具体年份更改
plt.plot(range(21),x[:len(x)],range(24),x_predict)
plt.xlabel('Months')
plt.ylabel('Concentration [mt]')
plt.legend(['Actual','Forecast'])
plt.savefig('1.png',dpi=600)
plt.show()
```

问题三代码：

```
17
import statsmodels.api as sm
from statsmodels.graphics.api
import qqplot
plt.rcParams['font.sans-serif']=['SimHei'] #用来画图时正常显示中文标签
plt.rcParams['axes.unicode_minus']=False  #用来画图时正常显示坐标轴负数
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
file1=open("C:\\Users\\malia\\Documents\\Tencent  Files\\2793055528\\FileRecv\\1.csv",encoding='utf-8
')
file2=open("C:\\Users\\malia\\Documents\\Tencent  Files\\2793055528\\FileRecv\\2.csv",'w',encoding='u
tf-8')
for row in file1.readlines():
row=row.strip('\n').split(',')
row=row[:40]
dta=list(map(int, row))
print(dta)
dta=np.array(dta,dtype=np.float)  #转换数据类型
dta=pd.Series(dta)
#print(type(dta))
dta.index=pd.Index(sm.tsa.datetools.dates_from_range('2001','2040'))
diff1=dta.diff(1)
diff2=dta.diff(2)
#diff11=np.diff(dta)
#plot_acf(diff11,lags=40,ax=ax1)
try:
arma_mod80 = sm.tsa.ARMA(dta,(8,0)).fit()
except Exception:
file2.write('exception\n')
continue
predict_sunspots = arma_mod80.predict('2040', '2052', dynamic=True)
for i in range(0,11):
file2.write(str(predict_sunspots[i])+',')
file2.write(str(predict_sunspots[11])+'\n')
file1.close()
file2.close()
```

问题三验证准确性代码：

```
#基本数据导入

from __future__ import print_function

import pandas as pd

import numpy as np
```

```
from scipy import stats
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.api import qqplot
plt.rcParams['font.sans-serif']=['SimHei'] #用来画图时正常显示中文标签
plt.rcParams['axes.unicode_minus']=False  #用来画图时正常显示坐标轴负数
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

dta=[42,58,46,67,71,63,68,40,29,38,24,18,21,35,77,55,79,79,39,33,24,83,67,120,110,96,106,7
6,69,84,83,52,77,51,64,53,46,53,80,122,64]
dta=np.array(dta,dtype=np.float)  #转换数据类型 dta=pd.Series(dta)#d 阶查分运算
dta.index=pd.Index(sm.tsa.datetools.dates_from_range('2001','2041'))
plt.plot(dta,label='原始数据')
plt.legend()  #让 label 生效
#plt.show()
plt.figure()
#新开一个图窗
diff1=dta.diff(1)
plt.plot(diff1,label='一阶')
diff2=dta.diff(2)
plt.plot(diff2,label='二阶')
plt.legend()
```

```
plt.show()

#根据得到差分后的平稳序列的 acf 图和 pacf 图，选择合适的 p、q。

fig=plt.figure()

ax1=fig.add_subplot(211) # 画子图

plot_acf(diff1[1:],lags=39,ax=ax1)

plt.show()

#diff11=np.diff(dta)

#plot_acf(diff11,lags=40,ax=ax1)

ax2=fig.add_subplot(212)

plot_pacf(diff1[1:],lags=40,ax=ax2)

plt.show()arma_mod70 = sm.tsa.ARMA(dta,(7,0)).fit()

arma_mod01 = sm.tsa.ARMA(dta,(0,1)).fit()

arma_mod71 = sm.tsa.ARMA(dta,(7,1)).fit()

arma_mod80=sm.tsa.ARMA(dta(8,0)).fit()

print(arma_mod70.aic,arma_mod70.bic,arma_mod70.hqic)

print(arma_mod01.aic,arma_mod01.bic,arma_mod01.hqic)

print(arma_mod71.aic,arma_mod71.bic,arma_mod71.hqic)

print(arma_mod80.aic,arma_mod80.bic,arma_mod80.hqic)

#对残差进行自相关检验

resid = arma_mod80.resid#残差

fig = plt.figure(figsize=(12,8))

ax1 = fig.add_subplot(211)

fig = sm.graphics.tsa.plot_acf(resid.values.squeeze(), lags=40, ax=ax1)
```

```
ax2 = fig.add_subplot(212)

fig = sm.graphics.tsa.plot_pacf(resid, lags=40, ax=ax2)

print(sm.stats.durbin_watson(arma_mod80.resid.values))

#残差 qq 图

fig = plt.figure(figsize=(12,8))

ax = fig.add_subplot(111)

fig = qqplot(resid, line='q', ax=ax, fit=True)

plt.show()

#DW 检验

print(sm.stats.durbin_watson(arma_mod80.resid.values))

#Ljung-Box 检验：LB 检验则是基于一系列滞后阶数，判断序列总体的相关性或者说随机性是否存

在。

r,q,p = sm.tsa.acf(resid.values.squeeze(), qstat=True)

data = np.c_[range(1,41), r[1:], q, p]

table = pd.DataFrame(data, columns=['lag', "AC","Q",

"Prob(>Q)"])print(table.set_index('lag'))

#预测

predict_sunspots = arma_mod80.predict('2042', '2054', dynamic=True)

print(predict_sunspots)

fig, ax = plt.subplots(figsize=(12, 8))

predict_sunspots.plot(ax=ax)

fig = arma_mod80.plot_predict('2042', '2054', dynamic=True, ax=ax,plot_insample=False)

plt.show()
```