基于非线性优化的补货与定价模型

摘要

蔬菜类商品的定价与补货是一种销售策略，指商家每天所做出的各蔬菜品类的补货与定价决策，以增加收益并满足市场需求。本文通过建立非线性模型，利用遗传方法，分析了各蔬菜品类的销量及其定价策略，并给出相应的补货与定价策略。该问题的研究能为商家的补货与定价提供指导性意见，帮助其最大化收益及更好地满足市场需求。

针对问题一，我们根据单品编码，通过 **VLOOK 函数**对附件 1、附件 2、附件 3 和附件 4，进行合并。然后我们对各品类商品进行了销量的统计，并计算了其在总销量中的占比。接着分析了这些占比随时间的变化趋势，并使用**皮尔逊系数**、**斯皮尔曼系数**和 **Kendall 系数**进行了相关性测试。结果显示，各品类之间的相关性都较高，表明它们之间存在一定的相关性。接下来，我们进一步分析了每个品类中以及不同品类之间相关性最高的单品。通过这些分析，我们可以更好地理解各品类商品之间的关联性，为商超的补货和定价决策提供更有针对性的建议。

针对问题二，我们采用了多种方法来改进补货和定价决策。首先，我们通过拟合销量和成本加成定价之间的关系，建立**对数拟合模型**，得到其基本关系。然后，我们使用 **SARIMA 时间序列预测**模型对下一周的批发价进行了预测，以便在制定定价策略时有更准确的参考。接着，我们构建了一个**非线性规划**模型，以成本作为决策变量，用预测的批发价经过成本加成后作为定价参考，设置合理的搜索区间，并利用**遗传算法**对其进行求解。这个模型综合考虑了销量、成本和定价，以找到最佳的定价策略。

针对问题三，在第二个问题的基础上，我们进一步引入了 **TOPSIS-熵权法**来进行单品选择，并增加了约束条件进行求解。首先，我们选择单品销售总量、单品进货量，单品单次最大进货量，单品损耗率作为指标，使用 **TOPSIS-熵权法**对每个单品进行评估和排序，考虑到多个指标的重要性和权重，以选择最具潜力和竞争力的单品。然后，我们引入了额外的约束条件，以确保所选单品符合 2.5kg 的最小陈列量要求。最后，我们通过求解带有约束条件的优化问题，确定最佳的单品选择方案，以满足商超的需求并实现最大化的效益。

针对问题四，在评价根据当前模型后我们发现其在分析市场需求和供应链因素考虑不足，为此我们建议收集更多这两方面的相关数据，并详细分析了这些数据对模型和决策的改进和优化作用，希望帮助商超在市场中取得更大成功。

最后我们对本文所建立的模型进行了讨论和分析，综合评价模型。

关键词 皮尔逊系数 SARIMA 模型 遗传算法 TOPSIS-熵权法

一、 问题重述

在当今的需求推动下，生鲜商超的生产模式逐步向多品种、小批量迈进，导致商场内蔬菜等种类变得更加琳琅满目，同时也给其库存管理及定价规划增加了很大的难度。如何根据品类繁多的历史销售数据，在蔬菜品类及单品上提供精准的需求预测，成为当前大多数生鲜商超需要重点关注并思考的问题。针对这一现状，本次研究旨在各品类蔬菜产品销售数据，为商场在蔬菜补货和定价方面做出有力的贡献，提供可行的解决方案。

1.1 问题一，研究蔬菜商品的销售数据，了解不同蔬菜品类和单品之间的可能关联。分析各蔬菜品类和单品的销售量分布模式，以及它们之间的相互关系

1.2 问题二，分析各蔬菜品类的销售总量与成本加成定价之间的关系，并提出未来一周（2023 年 7 月 1-7 日）各蔬菜品类的每日补货总量和最优定价策略，以实现最大化的商超收益。

1.3 问题三，制定更精确的补货计划，以确保可售单品总数在 27-33 个之间，并满足每个单品的最小陈列量要求为 2.5 千克。根据 2023 年 6 月 24-30 日的可售品种，提供 7 月 1 日的单品补货量和定价策略，以在尽量满足市场需求的前提下最大化商超的收益。

1.4 问题四，为了更精确地制定蔬菜商品的补货和定价策略，商超需要获取额外数据，探讨这些数据对问题解决的影响。

二、 问题分析

2.1 问题一分析，首先，我们将深入研究各个品类和单品的销售情况，以便全面了解它们在整体销售中的占比。此外，我们还将分析销售随时间的变化趋势，以揭示可能存在的季节性或趋势性模式。在这个阶段，我们还需要考虑从品类和单品之间的关联入手，进行相关性分析，以确定它们之间是否存在相互影响的情况。这一系列分析将为后续建立模型提供必要的基础。

2.2 问题二分析，首要任务是从历史数据中挖掘销售量与成本加成定价之间的关系，并通过曲线拟合来描述它们之间的函数关系。随后，我们将使用时间序列方法来预测下一周的成本。接下来，我们将构建一个优化模型，其中以预测出的成本作为搜索区间的中点。在优化过程中，我们将特别关注是否与前述建立的关系相符。此外，我们还可以考虑使用启发式算法来解决优化问题，以提高模型的效率和准确性。

2.3 问题三分析，在第二个问题的基础上，需要进一步考虑单品的选择。首先，我们将借助评价模型来选择适用的单品。这一选择过程将基于对单品的进货量、销量等指标进行综合评估。一旦确定了适合的单品，我们将继续使用第二个问题中构建的模型来进行进一步的求解和决策。

2.4 问题四分析，当制定商超蔬菜商品的补货和定价策略时，数据的收集和分析至关重要。首先，我们将评估问题一至问题三中所构建模型的局限性，以确定它们在实际应用中可能存在的不足之处。然后，我们将明确需要收集哪些相关数据以支持模型的改进，并进一步提高决策的准确性和实用性。这个过程将有助于我们优化现有模型，以更好地满足商超的需求，同时也能更有效地指导补货和定价策略的制定。

三、 符号说明

符号	说明
C	销售量
r	相关系数
M	最大库存
Q	销售量
a	平均损耗率
L	惩罚率
A	成本
A1	预测成本

四、 模型假设

(1) 假设在研究的范围内商品的销售不受宏观经济环境的影响的干扰
(2) 单品的信息及其所属品类和在所研究的时间段内保持不变
(3) 单品的损耗率在研究的时间段内保持稳定
(4) 假设销售数据真实有效

五、问题一的模型与求解

5.1 数据处理

我们根据单品编码，通过 VLOOK 函数对附件 1、附件 2、附件 3 和附件 4，进行合并。

5.2 品类分布规律

根据处理后的数据，我们首先对各品类的历史销售数据进行整体分析。首先统计各个品类的销量历史变化图。

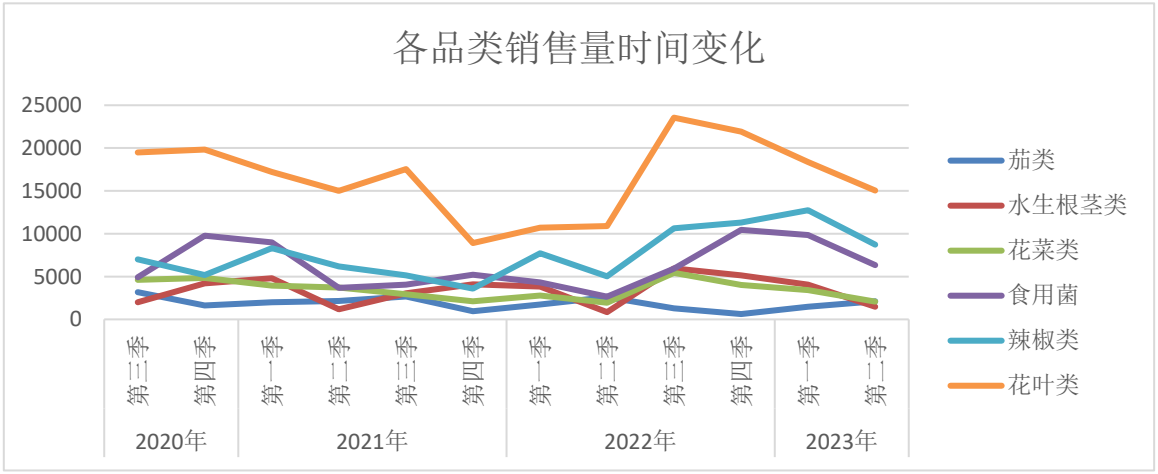


图 1

根据各品类销售量时间变化图得知，从 2020 年第三季度开始到 2022 年第二季度，各品类销量总体呈下降趋势，考虑和疫情及政策因素有关，之后总体开始回升，同经济复苏政策逐渐开放有关，符合预期。

接着我们分析各个品类的销售比例及历史销量最高的十个商品

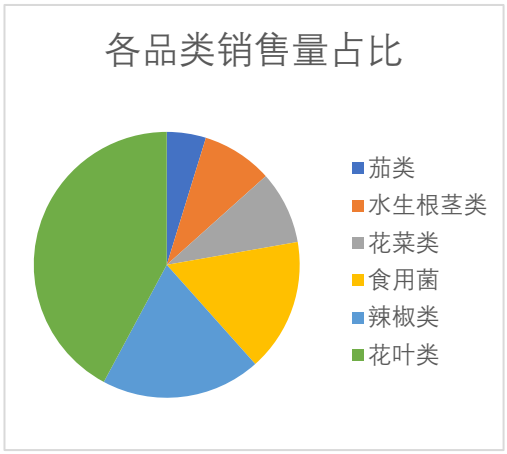


图 2

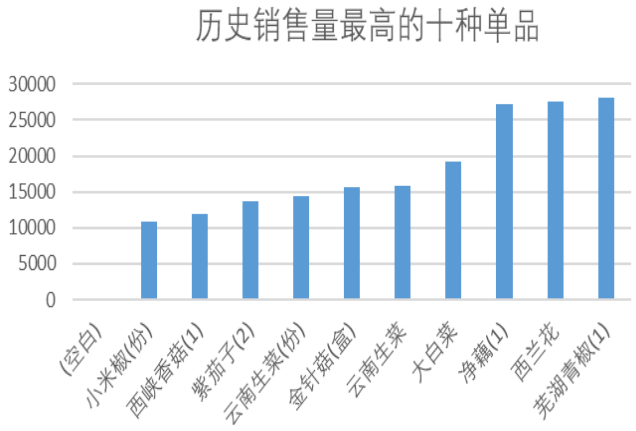


图 3

花叶类数据在六大蔬菜品种 中销量最高，茄类数据销量最少；从整体上 来看，数据波动变化较大；从各品类的比例及部分单品我们得知该商超处于云南地区，除了常见的花叶类外辣椒类和食用菌类销量较高，符合云南地区的饮食习惯。

5.3 品类的相关性

我们首先评估六个品类之间的线性相关性，即两两组合画出其关系图。结果如所示

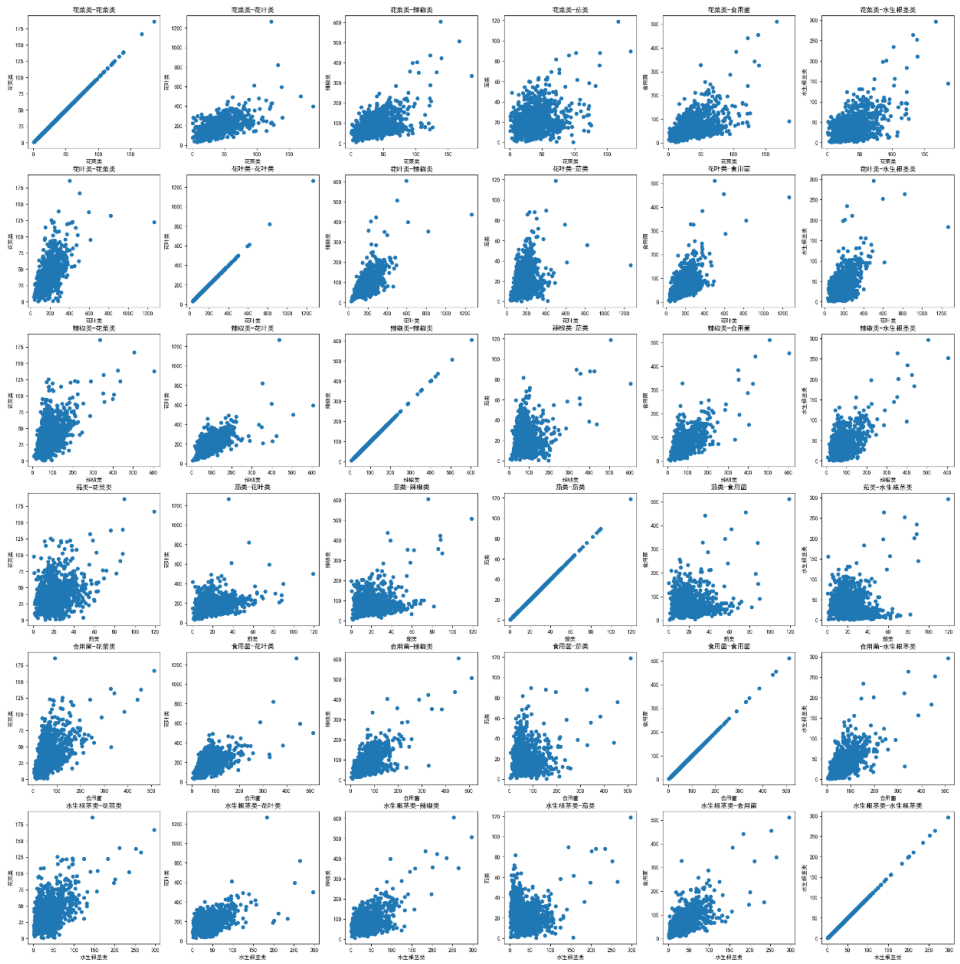


图 4

从图可以发现各品类之间都有一定的正相关性，接着我们分析其皮尔逊相关系数和斯皮尔曼相关系数。

在统计学中，皮尔逊相关系数是用于度量分析适用于服从正态分布的两个变量 X 和 Y 之间的相关性。其计算公式为

$$r = \frac{n\sum(xy) - \sum x \sum y}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}}$$

其中 n 表示总的观测样本数如果 r 为正，则 y 与 xi 正相关，若反之为负则负相关。且 |r| ≤ 1 越接近 1，则说明变量 Y 与 X 之间的线性关系越强。

斯皮尔曼系数相关系数适用于定量变量或定序变量两两之间的相关分析，利用两变量的秩次大小作线性相关分析，对原始变量的分布不作要求其计算公式如下

$$\rho = 1 - \frac{6\sum d^2}{n(n^2-1)}$$

其中d_i 表示第 i 个数据对的位次值之差

其结果 用热度图表示如下，其中颜色越深表示皮尔逊系数或斯皮尔曼系数越大。

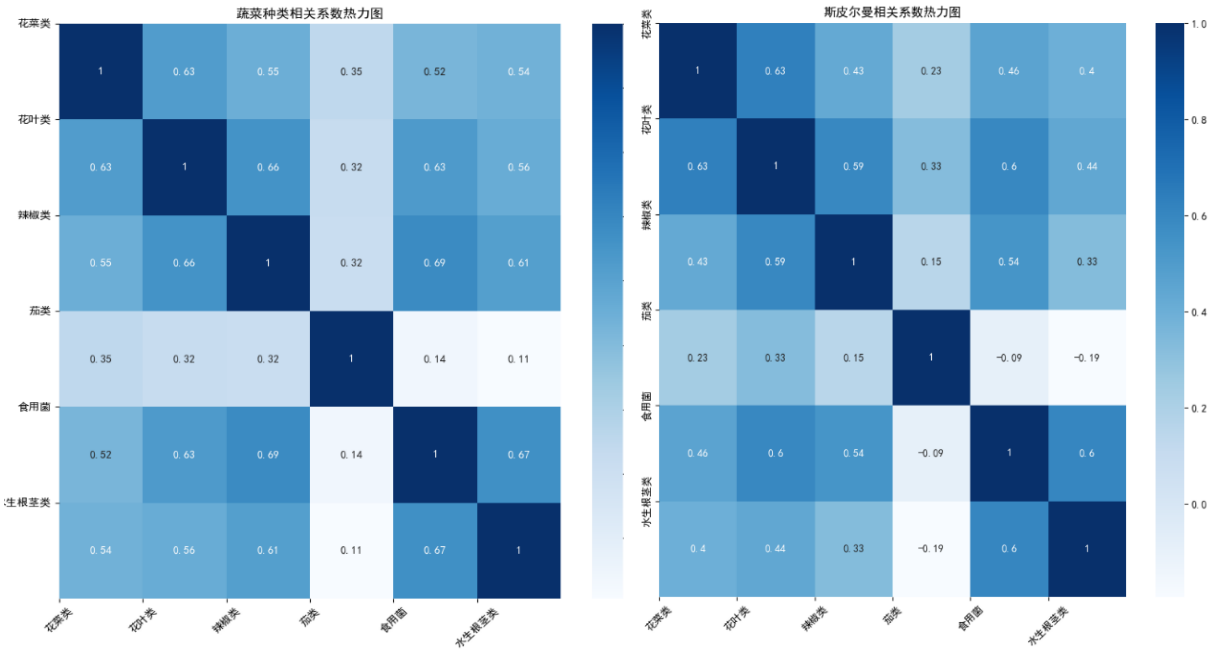


图 5

图中可以看出各种品类相关性都基本较高，最后我们对各品类进行 Kendall 一致性检验结果如下。

Kendall 一致性检验

Kendall's W 分析结果					
名称	秩平均值	中位数	Kendall's W 系数	X^2	P
花菜	2.5	3512.713	0.898	53.905	<0.001
花叶	6	15724.608			
辣椒	4.75	7072.063			
茄类	1.417	1813.697			
食用菌	4.25	5512.387			
水生根茎	2.083	3678.321			

由分析结果得知，总体数据的显著性 P 值小于 0.001，水平上呈现显著性，拒绝原假设，因此数据呈现一致性，同时模型的 Kendall 协调系数W值为 0.898，因此相关性的程度为几乎完全一致性。 说明六个品类的相关性程度都较高。

4.4 单品的相关性

接下来我们对附件一中各品类进行两两分析其皮尔逊系数，找出各个类别中相关性最大的单品如下：

花菜类：青梗散花-枝江青梗散花，紫白菜-紫 白菜-西兰花

花叶类：上海青-黄白菜，娃娃菜-菠菜

辣椒类：芜湖青椒-螺丝椒，红椒-小米椒，小米椒-小皱皮

茄类：青茄子-紫茄子，圆茄子-长线茄

食用菌类：白玉菇(袋)- 西峡香菇(1)，银耳(朵)- 杏鲍菇(袋)

水生根茎类：净藕(1)- 洪湖莲藕(粉藕)，荸荠-高瓜（2）

而在不同品类单品间相关性较高的有小皱皮(份)-云南生菜(份) ,青梗散花-金针菇(1) ,青尖椒-云南油麦菜，姬菇(包)-黄心菜(1)，红杭椒-云南油麦菜，青茄子(1)-红薯尖等，其中多为常见的家常菜搭配组合，符合预期。

六、问题二模型建立与求解

6.1 探究关系

6.1.1 数据处理

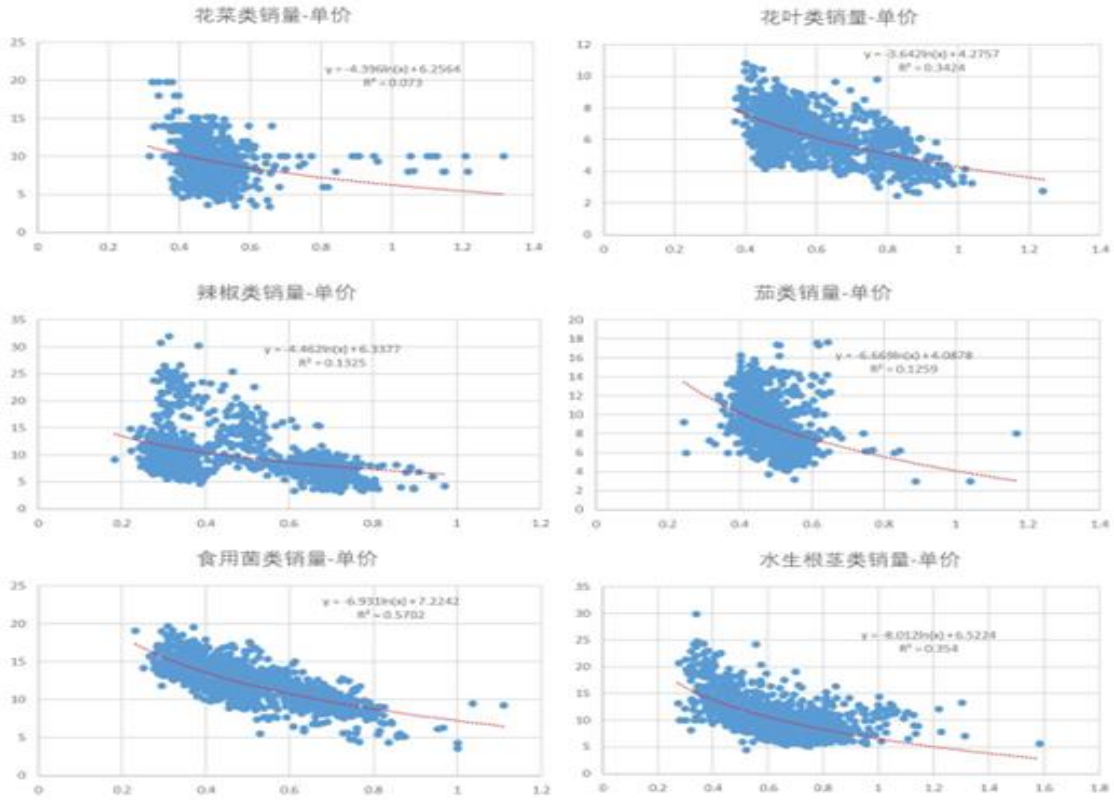
由于我们需要分析各个品类的成本加成定价，我们考虑计算各个品类的加权定价。首先统计每个单品每天的销量的求和，以其相对于该品类在该天销量的百分比作为权重，并在同一品类进行加权求和代表该品类的成本加成定价。

6.1.2 进行算数拟合

然后我们考虑多种拟合方法对六种品类的评价销量-售价进行拟合，包括指数，对数，乘幂，多项式，结果对比如下

	指数	对数	多项式	乘幂
R 方平均值	0.2184	0.3809	0.2612	0.2393

对比方向对数拟合的 R 方平均值最大，说明拟合效果较好，用其对六个品类进行拟合效果如下。



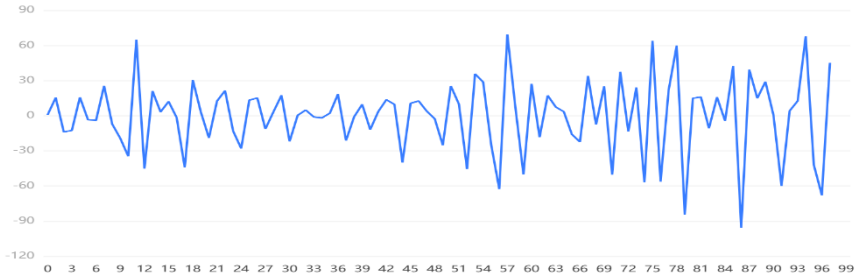
图六

6.2 预测

由上面的数据处理我们得到了 2020 年 7 月到 2023 年 6 月每一天各个品类的成本加成定价接下来我们需要对其对其进行时间序列预测。

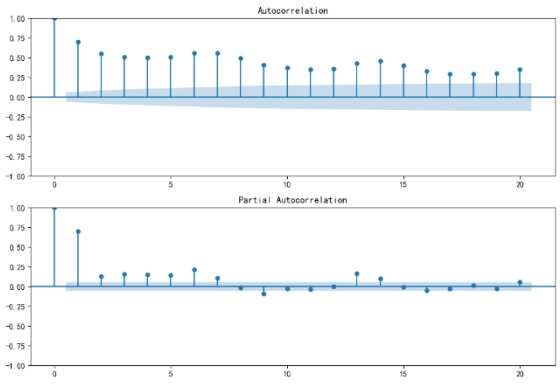
(1) 首先由于数据不是平稳的，首先进行季节性差分操作，直到获得平稳的时间序列。

这里我们以花菜类数据为例为例，可以看到经过二阶差分后波动已经趋于平稳



图七

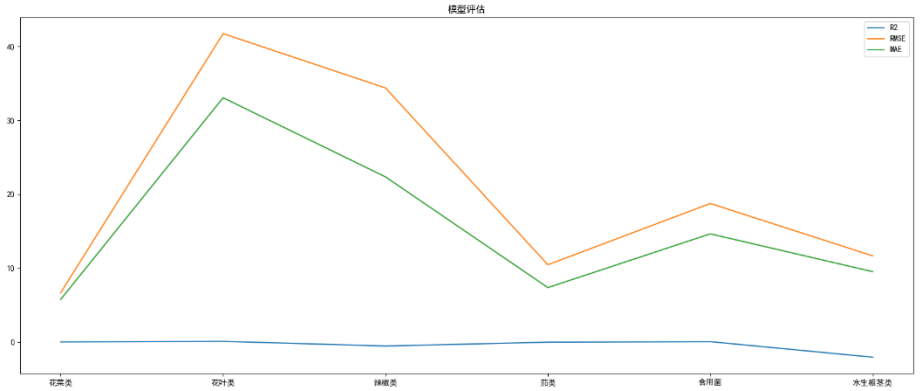
(2)确定模型阶数：使用自相关函数（ACF）和偏自相关函数（PACF）分析来确定 SARIMA 模型的阶数。通过图表确定自回归（AR）和移动平均（MA）的阶数，以及季节性 AR 和季节性 MA 的阶数。



图八

(3)根据图像和阶数选择，建立初始的 SARIMA 模型。其中包括选择 AR、I、MA 和季节性 AR、季节性 I、季节性 MA 的阶数。用所选的 SARIMA 模型阶数对数据进行拟合。

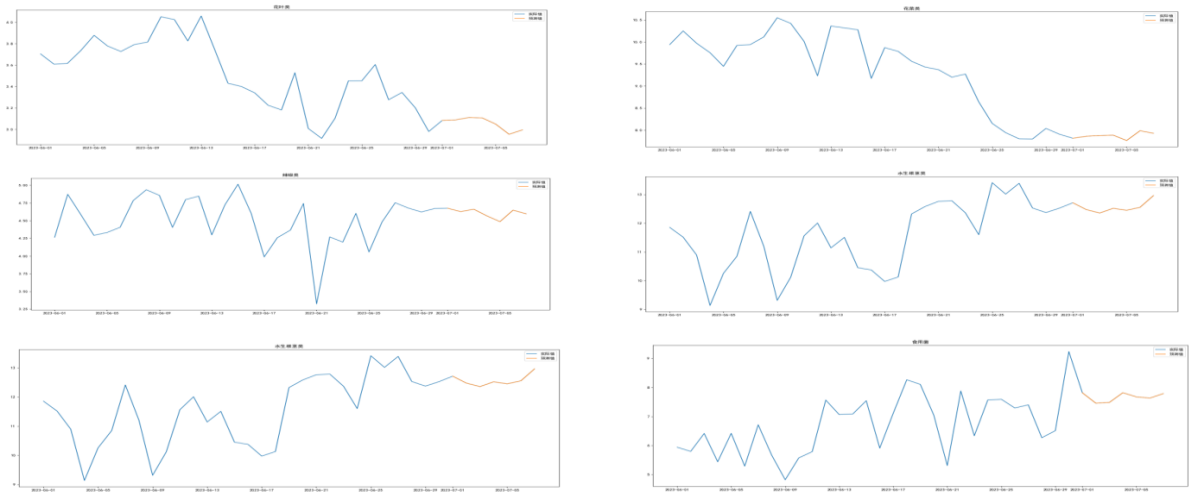
(4)模型评估：通过 R 方、均方根误差（RMSE）、平均绝对误差（MAE）、来检查模型是否符合基本假设。



图九

分析得知模型总体效果较好，但在某些类别上存在较为明显波动。

使用建立好的 SARIMA 模型进行未来数据的预测。通过 2020 年 7 月 1 日至 2023 年 6 月



30 日各蔬菜种类加权批发价格的相关数据，预测各蔬菜品类未来一周(2023 年 7 月 1-7 日)的批发价格。

6.3 优化求解定价策略

根据要求，我们建立非线性规划模型，求出日补货总量和定价策略，使商超的收益最大。

（1） 确定决策变量：x,y 分别表示销量和定价

（2） 约束满足条件 ：

x,y 需要在以预测值为区间中点，半径为预测曲线的标准差变化；

销售量量 C 不能大于最大库存

X,Y 关系和 6.1 中拟合的 F(X)误差不大

$$x \sim (\hat{x_i} \pm 2\sigma_1)$$

$$y \sim (\hat{y_i} \pm 2\sigma_2)$$

$$\frac{y - f(x)}{f(x)} < \varepsilon$$

$$0 \leq C \leq Q_{MAX} \times (1 - \beta)$$

$$0 \leq C \leq M_{max}$$

损耗率有所限制，即：

$$0 \leq \eta \leq 1$$

（3） 目标函数：商超取得最大收益，成本关系式：

$$A_1 = (1 + a)\overline{A_1^{'}}$$

通过在目标函数之后加入一个惩罚项 L 来约束，避免其正相关导致的定价无限大的情况。当定价高的时候可以使其其进货量降低，当定价低的时候，使其进货量变高， 惩罚项的表示如下：

$$P^{'} = \frac{P}{1 + L}$$

其中 L 为：

$$L = (\frac{P - \bar{\mu}}{\sigma})^2$$

最大收益函数：

$$max((p' - A) \times C)$$

6.4 使用遗传算法求解

遗传算法的基本步骤

Step1 初始化种群：随机生成初始种群，通常表示为 P, 其 中 N 代表种群大 小。P=[X₂. ...,Xv] 。本文将采用二 进制编码的方式，用 0 和 1 组成的数字串模拟染色体，可以很方便地实现基因交叉、变异等操作。

Step2 设置适应度函数：定义一个适应度函数 f(X) 用 于评估每个个体 x 的适应度。适应度函数 (Fitness Function) 的选取直接影响到遗 传算法的收敛速度以及能否找最优解，以适应度函数为依据，利用种群每个个体的适应程度来指导搜索。

Step3 选择： 使 用选择操作来确定哪些个体将被保留为父代。本文使用锦标赛方法选择，其中 P 代表第 1 代的种群。

Step4 交叉 ：选择两个个体， 然后使用交叉操作来生成子代。交叉操作通 常表示为 C。

Step5 变异：对某些个体应用变异操作，以引入新的遗传信息

Step6 替换:使用新的子代和变异个体替代旧的个体，形成下一代种群。

Step7 设置终止条件：适应度达到阈值来决定是否停止迭代。

得出结果如下所示

蔬菜品类		第1天	第2天	第3天	第4天	第5天	第6天	第7天
花叶类	定价	118.5	119	117.5	120.3	118.5	119	121.4
	补货量	180.1	181.2	180.3	180.6	182	181.2	183.5
水生根茎类	定价	80.2	79.5	78.6	81	82.3	80.4	79.6
	补货量	13.6	14	13.8	13.1	13.7	13.8	13.5
茄类	定价	27.1	28.9	26.5	28	27.7	28.1	29.4
	补货量	19.7	19.4	19.5	19.6	19.3	19.6	19.8
辣椒类	定价	43.8	44.2	43.1	42.8	41.3	43.5	44.3
	补货量	73.1	73.5	73.6	74.1	74.2	73.2	73.5
食用菌类	定价	39.2	39.4	39.5	38.2	38.4	39.1	39.5
	补货量	50.8	51.1	53.1	53.1	50.8	50.9	50.2
花菜类	定价	11.5	11.8	11.1	11.6	11.2	12.1	12
	补货量	21.6	21.8	21.9	22	22.3	21.5	21.6

七、问题三模型建立与求解

因蔬菜类商品的销售空间有限，要求可售单品总数控制在 27-33 个，应该对各可售单品进行筛选，对此我们通过多个维度指标进行单品的选择。

评价指标用于刻画被评价对象的特征或属性。每一项所选取的评判指标，均不同方面反映评价对象所具有某种特征大小的一个度量。而在多个评价指标构成指标体系时，构建评价指标体系的原则在于系统性、科学性、可比性、可测性和独立性。

依据此原则，分析单品种蔬菜的历史数据， 构建能够该种蔬菜重要性的指标。本文从单品种蔬菜的销售数据与存储两个角度出发，构建某品种蔬菜总销量，某品种蔬菜进货量和单次最大进货量用于反映单品种蔬菜的销售数据；平均损失率反映该商品的存储；以这 4 个指标，建立多指标评价模型， 以模型的输出结果作该品种蔬菜重要性的体现

指标计算：

单品蔬菜总销量，用来衡量该品种蔬菜在市场上的受欢迎程度以及销售情况

单品蔬菜进货量，可以提供有关市场需求、供应链稳定性、价格趋势和销售趋势等方面的关键信息

单次最大进货量、平均损失率

然后我们使用 Topsis-熵权法进行单品的评价与选择

TOPSIS 能用于多属性评估和排序。而熵权法则用于确定属性权重， 指标的变异程度越小，所反映的信息量也越少，其对应的权值也应该越低。确定了能有效描述单品种蔬菜重要指标后，构建单品种蔬菜重要性评价模型，通过熵权法方法求解出各个指标的权重，建立基于 TOPSIS 的多指标评价模型，该得分即反映出单品种蔬菜重要性，值越大，则重要性程度越高。

熵权法是根据指标变异性的大小来确定客观权重。一般来说，若某个指标的信息熵越小，表明指标值得变异程度越大，提供的信息量越多，在综合评价中所能起到的作用也越大， 其权重也就越大。相反，某个指标的信息熵越大，表明指标值得变异程度越小， 提供的信息量也越少，在综合评价中所起到的作用也越小， 其权重也就越小。

TOPSIS 法是一种常用的组内综合评价方法， 能充分利用原始数据的信息， 其结果能精确地反映各评价方案之间的差距。基本过程为基于归一化后的原始数据矩阵，采用余弦法找出有限方案中的最优方案和最劣方案，然后分别计算各评价对象与最优方案和最劣方案间的距离，获得各评价对象与最优方案的相对接近程度，以此作为评价的好坏。

最终我们选择了娃娃菜，小皱皮等 30 个单品

接下来由于最小陈列量要求为 2.5kg，于是我们在第二问基础上加上约束条件，

进行求解，得到结果如下：

	奶白菜	竹叶菜	娃娃菜	菠菜	小青菜(1)	木耳菜	海鲜菇（包）
进货量	9.5	13	6	4.5	9.5	17	23
售价	4.9	3.6	6.6	13.9	5.1	5.7	2.4
	云南生菜	西兰花	云南油麦菜	上海青	云南油麦菜（份）	云南生菜（份	七彩椒
进货量	13.5	19	24	3.5	19	13	27
售价	8.9	11.5	6.9	7.8	4.2	4.4	18.9
	姜蒜小米椒组合	红椒	芜湖青椒	小米椒(份)	青红杭椒组合装(份)	虫草花(份)	螺丝椒
进货量	19.5	5	8	14	29	9.5	16.5
售价	4.3	18.9	5.3	5.8	5.5	3.7	11.4
	青线椒(份)	西峡花菇(1)	金针菇(盒)	菱角	紫茄子	青茄子(1)	野生粉藕
进货量	22	23.5	19	28.5	27	4.5	24
售价	4.2	23.5	1.9	14	9.2	5.9	25.9

八、问题四-对企业的建议

当前模型在市场需求建模、供应链因素的处理方面存在局限性。虽然我们已经在前文提出了数据收集的建议，但现在让我们更深入地探讨每个建议的内容，以及为什么这些数据对于改进模型的准确性和实用性至关重要。

8.1 市场需求数据

市场趋势数据的收集和分析对于预测未来需求至关重要。市场趋势是市场行为和消费者习惯的反映，它们可以在很大程度上影响蔬菜销售。收集市场趋势数据的重要性体现在以下几个方面：

消费者购买习惯的变化： 消费者购买蔬菜的习惯可能会随着时间而变化。例如，某一品类的蔬菜可能在某个季节或健康风险的考虑下变得更受欢迎。了解这些变化可以帮助我们调整库存和供应计划，以满足市场需求。

季节性需求的波动： 不同季节，消费者对蔬菜的需求可能会有显著的波动。例如，夏季可能会带来对沙拉材料的高需求，而冬季则可能增加对根茎蔬菜的需求。市场趋势数据可以帮助我们预测季节性波动，从而更好地规划补货和定价策略。

新兴蔬菜品类的兴起： 市场趋势数据还可以揭示新兴蔬菜品类的兴起。了解哪些蔬菜品类在市场上崭露头角可以让我们及早调整采购和供应链策略，以满足潜在的高需求。竞争对手行为。这些数据可以让我们更好地了解竞争环境，以及竞争对手的策略如何影

响我们的补货和定价策略。这些数据包括竞争对手的销售及定价数据，促销活动等，解竞争对手的以上数据可以帮助我们分析他们在市场上的表现，我们可以考虑调整自己的供应和定价策略，以更好地与竞争对手竞争，吸引更多消费者。

8.2 供应链因素数据

在考虑其他关键因素数据时，我们需要深入了解与供应链决策相关的因素，这有助于我们更全面地建模和优化供应链决策，确保满足实际业务运营的要求，以更好地优化我们的模型和策略。以下是其他关键因素数据的详细内容：

运输成本数据：运输成本是供应链中的一个重要因素，直接影响着蔬菜的最终成本。通过收集有关运输成本的详细数据，我们可以更准确地计算成本，并在补货和定价决策中考虑运输成本的影响。这有助于我们选择最经济的供应来源和运输方式。

库存成本数据： 库存成本是管理库存的关键因素。了解库存成本将帮助我们确定最佳的库存水平，以避免库存过多或过少。这可以减少库存持有成本，并确保我们能够及时满足市场需求。

生产能力数据： 如果我们的蔬菜供应商有生产能力的限制，了解这些限制是至关重要的。收集有关供应商生产能力的数据可以帮助我们规划供应链，确保能够满足市场需求。

最小订购量数据：收集供应商对不同蔬菜品类的最小订购量要求是确保供应链顺畅的关键因素。如果我们无法达到最小订购量，可能会影响供应商的供货能力，因此这些数据对我们的决策非常重要。

最大库存容量数据： 不同仓库或销售点可能有不同的最大库存容量限制。通过收集这些数据，我们可以在库存管理方面设置适当的上限，以避免库存过剩。

通过对上述数据的全面收集和分析，并将其纳入我们的模型中，我们可以更好地了解市场需求的变化、供应链的约束条件，并最终优化补货和定价决策。这些数据的使用将提高我们模型的准确性和实用性，为实际供应链决策提供更有力的支持，帮助商超在竞争激烈的市场中取得更大的成功。总之，数据的广泛收集和分析是不可或缺的，它们为决策制定提供了更全面、更可靠的依据，有助于实现更高效、更智能的蔬菜商品补货和定价策略。

九、 模型的评价与推广

优点：

（1）非线性规划能够更好地处理问题中存在非线性关系的情况。它可以应对更复杂、更现实的问题。并且，非线性规划可以提供更高的问题精确度，在需要高精度的应用中非常重要。

（2）SARIMA 模型适用于各种时间序列数据，包括具有趋势和季节性成分的数据。SARIMA 模型考虑了数据中的季节性成分，它可以捕捉季节性波动和趋势变化，并且能进行长期预测。

（3）TOPSIS-熵权法综合了 TOPSIS 和信息熵权重，能够更好地处理属性权重的不确定性和属性数据的分布特性。

缺点：

（1）SARIMA 模型对数据的要求相对较高，需要具有一定的时间序列模式。对于非常嘈杂或难以建模的数据，SARIMA 不好处理。

（2）在某些特定情况下，非线性规划问题可能不存在可行解，或者难以找到可行

解。这使得问题更加复杂。

（3）题目所给数据类型不充分，无法很好考虑到市场和供应链方面因素。

模型推广：

非线性规划模型适用于解决许多实际问题，包括工程设计、经济规划、生产计划、物流优化等各个领域，适用性广泛

十、参考文献

[1]孙颖,池宏,贾传亮.多路径下应急资源调度的非线性混合整数规划模型[J].运筹与管理,2007(05):5-8.

[2]孙峻. 非线性模型预测控制理论及应用研究[D].西北工业大学,2002.

[3]石国春. 关于序列二次规划（SQP）算法求解非线性规划问题的研究[D].兰州大学,2009.

[4]徐磊. 基于遗传算法的多目标优化问题的研究与应用[D].中南大学,2007.

[5]金芬. 遗传算法在函数优化中的应用研究[D].苏州大学,2008.

[6]杜普燕. ARMA 模型参数估计算法改进及 SARIMA 模型的应用[D].燕山大学,2010.

[7]焦亮,李文杰,赵超敏等.基于 Eviews 的季节时间序列(SARIMA)预测模型[J].河南预防医学杂志,2015,26(05):349-352.DOI:10.13515/j.cnki.hnjpm.1006-8414.2015.05.008.

[8]李岩,袁弘宇,于佳乔等.遗传算法在优化问题中的应用综述[J].山东工业技术,2019(12):242-243+180.DOI:10.16640/j.cnki.37-1222/t.2019.12.210.

[9]荀梦君,李进岚,黄爱菊等.ARIMA 模型和 Holt-Winters 指数平滑法在贵州省肺结核发病预测中的应用[J].中国预防医学杂志,2023,24(07):678-682.DOI:10.16506/j.1009-6639.2023.07.011.

[10]耿娟娟. 基于 X-12-ARIMA 和 SARIMA 模型及其组合模型的 CPI 预测研究[D].西南石油大学,2015.

[11]安致远,何恩球.基于 SARIMA 时间序列模型的区域快递需求预测——以江苏省为例[J].物流科技,2022,45(20):63-66+70.DOI:10.13714/j.cnki.1002-3100.2022.20.017.

十一、附录

部分非线性优化及遗传算法代码如下

```
import numpy as np
from scipy.optimize import minimize
from scipy.optimize import Bounds
from scipy.optimize import NonlinearConstraint
from scipy.optimize import differential_evolution

# 定义目标函数
def objective_function(x):
    # 这里替换成你的目标函数
    return your_objective_function(x)
```

```
# 定义约束条件
def constraint_function(x):

    return constraint_function(x)

# 定义变量的边界
bounds = Bounds([lower_bound1, lower_bound2, ...], [upper_bound1, upper_bound2, ...])

# 定义非线性约束
nonlinear_constraint = NonlinearConstraint(constraint_function, lb=[lb_constraint1,
lb_constraint2, ...], ub=[ub_constraint1, ub_constraint2, ...])

# 使用 SciPy 的 minimize 函数进行非线性规划，采用遗传算法方法
result = differential_evolution(objective_function, bounds,
constraints=[nonlinear_constraint])

# 输出最优解和最优值
print("最优解：", result.x)
print("最优值：", result.fun)
import numpy as np
import random

# 目标函数（x*y 表示最大化收益）
def objective_function(x, y):
    return x*y

# 初始化种群
def initialize_population(population_size, x_bounds, y_bounds):
    population = []
    for _ in range(population_size):
        x = random.uniform(x_bounds[0], x_bounds[1])
        y = random.uniform(y_bounds[0], y_bounds[1])
        population.append((x, y))
    return population

# 计算适应度
def calculate_fitness(individual):
    x, y = individual
    return objective_function(x, y)

# 选择操作（轮盘赌选择）
def selection(population, num_parents):
    fitness_values = [calculate_fitness(individual) for individual in population]
    selected_parents = []
```

```
    for _ in range(num_parents):
        total_fitness = sum(fitness_values)
        selection_probabilities = [fitness / total_fitness for fitness in
fitness_values]
        selected_index = np.random.choice(len(population), p=selection_probabilities)
        selected_parents.append(population[selected_index])

    return selected_parents

# 交叉操作（单点交叉）
def crossover(parent1, parent2):
    crossover_point = random.randint(1, len(parent1) - 1)
    child1 = parent1[:crossover_point] + parent2[crossover_point:]
    child2 = parent2[:crossover_point] + parent1[crossover_point:]
    return child1, child2

# 变异操作
def mutation(individual, mutation_rate, x_bounds, y_bounds):
    if random.random() < mutation_rate:
        x, y = individual
        x = random.uniform(x_bounds[0], x_bounds[1])
        y = random.uniform(y_bounds[0], y_bounds[1])
        return (x, y)
    return individual

# 主遗传算法函数
def genetic_algorithm(population_size, num_generations, x_bounds, y_bounds,
num_parents, mutation_rate):
    population = initialize_population(population_size, x_bounds, y_bounds)

    for generation in range(num_generations):
        parents = selection(population, num_parents)
        offspring = []

        for i in range(0, len(parents), 2):
            parent1 = parents[i]
            parent2 = parents[i + 1]
            child1, child2 = crossover(parent1, parent2)
            child1 = mutation(child1, mutation_rate, x_bounds, y_bounds)
            child2 = mutation(child2, mutation_rate, x_bounds, y_bounds)
            offspring.extend([child1, child2])

        population = offspring
```

```
    # 寻找最优解
    best_solution = min(population, key=lambda x: calculate_fitness(x))
    best_fitness = calculate_fitness(best_solution)

    return best_solution, best_fitness

# 设置算法参数
population_size = 100
num_generations = 100
x_bounds = (-5, 5)
y_bounds = (-5, 5)
num_parents = 50
mutation_rate = 0.1

# 运行遗传算法
best_solution, best_fitness = genetic_algorithm(population_size, num_generations,
x_bounds, y_bounds, num_parents, mutation_rate)

print("最优解：", best_solution)
print("最优值：", best_fitness)
```