基于市场原理和数据挖掘的商超蔬菜补货与定价策略研究

摘要

本题要求研究商超出售的蔬菜品类和单品销售量之间的关系与分布规律，随后探究销售总量与成本加成定价的关系，并且基于此求解两个规划问题，最后要求给出采集更多类型数据的建议。本文主要采用皮尔逊检验、ADF 平稳性检验、格兰杰因果检验等方法，建立了**时间序列的需求价格变动、规划模型**，采用了**随机森林预测、遗传算法**等解决了蔬菜自动补货与定价决策问题。

首先对附件中的数据进行预处理，删除了突变型异常数据，发现品类的售价出现缺省值，采用**指数平滑方式**处理缺省值，使得数据更为可靠。

对于问题一，首先研究蔬菜单品的分布规律，做出时间序列折线图，观察得知蔬菜单品销售量随时间的变化存在周期型、突变型、互补型三种分布规律。然后采用 **ADF 平稳性检验**的方式，研究出各蔬菜品类销售量随时间变化的平稳性，最终得出花叶类外其他品类销售量的时间序列都是平稳的结论。最后通过画出各品类的堆积柱状图分析各品类销售量三年内的分布规律和相关关系，然后采用皮尔逊检验的方式检测品类之间和单品之间的相关性，做出热力图，对相关性分析结果做出相应的分析。

对于问题二，首先研究了销售总量与成本加成定价的关系，创新之处在于采用了多种方法对其关系进行探索，例如，**皮尔逊检验、ADF 平稳性检验、格兰杰因果检验**等。最后得到一个探究各品类销售总量与成本加成定价的关系的方向。随后，本文研究了弹性销售量与利润率之间的关系，做出散点图。随后还研究销售量变化与价格变动之间的关系，做出了散点图并进行了线性拟合，得出了如下结论：除花叶类外，其他品类的销售量变化与定价的变化都成负相关关系，结果比较满足市场原理，故较为满意。然后本文根据数据结果与市场原理创新性地构造了**销售量变化对定价变化的函数**，发现其与目标数据拟合得很好。于是基于此建立**动态规划模型**，以各品类每天补货量和定价为决策变量，求七天的最高收益。最后采用**遗传算法**求解，得到最高收益为 **35477 元**。

对于问题三，我们结合单品特征，筛选出了 2023 年 6 月 24-30 日可售的蔬菜单品。然后通过**随机森林算法**预测出了 7 月 1 日每个品类的需求量和每个单品的进价。之后引入上一问中得到得销售量变化对于定价变化得特征函数，结合规划思想，以单品补货量和定价作为决策变量，建立**规划模型**，求解在满足品类需求的情况下，单日的最大收益。最后采用遗传算法求解，得到最高收益为 **3671 元**。较为合理地解决了蔬菜单日地自动补货与定价决策。

对于问题四，本文考虑选取**竞争对手的定价**和**补货量**作为关键数据，决定自己的定价和补货量。对于此，本文参考**博弈理论**，创造性地提出了**竞争强度方程**、定价差对潜在顾客源变化的相关函数，用于评估与竞争对手定价的差异对于客源的影响。

关键词：遗传算法，随机森林算法，规划思想，时间序列检验

目录

一、 问题重述

1.1 问题背景

本题研究的是生鲜超市中蔬菜类商品的补货定价策略问题。

生鲜超市的蔬菜类商品有几类显著的特征，一是商品的保质期比较短，品相随时间的增加而变差，导致大部分品种如当日未售出，隔日就无法再出售，这就导致商品补货

与定价的策略是具体到每一天的策略；二是商品的销售量通常与时间存在关联关系，往往呈现出一定的季节性与周期性，且长期观察的结果应该显示出一定的平稳性，短期上来看又可能呈现出一定的突变性，这就需要深入研究商品销售量的随时间的分布规律；三是商品的供应具有明显的季节性，在 4 月到 10 月蔬菜类商品的品种较为丰富，其他月份则不然，在不同月份中作决策时，需要考虑的因素不尽相同；四是商品的销售空间有限，而不同品类之间的蔬菜的销售量往往呈现出一定的相关性，例如不同的蔬菜之间可能存在搭配，人们在购买时往往会成对购买。所以，选择合理的销售组合能够较好地提升销售量，有利于商超收益的提高。基于以上蔬菜类商品的特征，需要研究不同蔬菜商品之间的相互关系，研究蔬菜商品随时间的分布规律，再根据不同月份具体时间，设计合理的补货策略。

蔬菜的定价往往采用“成本加成定价”方法，同时对于运损与品相变差的商品进行打折销售，定价策略的指定需要考虑上述因素，再综合蔬菜商品的进货进行合理设计。

1.2 题目的信息

题目给出了某商超经销的 6 个蔬菜品类的商品信息；给出了从 2020 年 7 月 1 日至 2023 年 6 月 30 日各个商品的销售流水和批发价格的相关数据；还给出了通过近期盘点周期数据计算出的各商品近期的损耗率数据。

1.3 需要解决的问题

问题一：分析蔬菜商品各个品类以及单品销售量的分布规律以及相互关系；

问题二：以品类为单位做补货计划，分析各蔬菜品类的销售总量与成本加成定价的关系，并给出各蔬菜品类未来一周（2023 年 7 月 1-7 日）的日补货总量和定价策略，使得商超收益最大。

问题三：为进一步制定商品补货计划，要求可售单品总数控制在 27-33 个，且各个单品订购量满足最小陈列量 2.5kg。现需要根据 2023 年 6 月 24 日至 30 日的可售品种，给出 7 月 1 日的单品补货量和定价策略，在尽量满足市场对各品类蔬菜商品的前提下，使得商超收益最大。

问题四：为了更好制定蔬菜商品的补货和定价决策，商超还需要采集哪些相关数据，对解决上述问题有何帮助，给出意见与理由。

二、 模型的假设

- 题目中所给出的数据基本可信。
- 从长期来看，消费者对于各类蔬菜的需求应该是平稳的。
- 从长期来看，各类蔬菜的价格应该是平稳的。

- 假设三年内该商家的补货货量基本与销售量相当，用销售量估计补货量不会产生较大误差。
- 假设商家在 7 月 1 日这天没有库存蔬菜。
- 在研究范围内，市场价格指数没有发生显著变化。

三、 符号说明

符号	意义	单位
N_{ij}	第 i 日第 j 品类的销量	千克（kg）
M_{it}	第 i 品类第 t 天的进货量	千克（kg）
P_{it}	第 i 品类第 t 天的价格	元/千克
Q_{it}	第 i 品类第 t 天的需求量	元/千克
Ω_{ij}	第 i 日第 j 品类的目标利润率	%
S_{it}	第 i 品类第 t 天的库存量	千克（kg）
l_k	每个单品的损耗率	%
θ	折扣	%
C_{it}	第 i 品类第 t 天的成本	元/千克
Pr_{it}	第 i 品类第 t 天的收益	元
Q_i	第 i 品类的预计需求量	千克（kg）
Q_{ijt}	第 i 品类第 j 单品第 t 天的预计需求量	千克（kg）
P_{ijt}	第 i 品类第 j 单品第 t 天的定价	元/千克
C_{ij}	第 i 品类第 j 单品的成本	元/千克
m_{ij}	第 i 品类第 j 单品的补货量	千克 (kg)
l_{ij}	第 i 品类第 j 单品的损耗率	%
Pr_{ij}	第 i 类第 j 单品的收益	元

四、 问题分析

蔬菜类商品具有几类明显的特性，导致补货定价决策的颗粒度细化到每一天，每一天的补货定价决策中，都需要考虑不同的销售组合，不同品种当日的市场需求量，来确定当日不同品类的补货量和定价策略。从这个角度考虑问题，问题三是比较贴近现实生活的核心问题，在解决问题一和问题二的基础上，通过了解蔬菜各品类和单品的相互关系，和市场上销售总量与成本加成定价的关系，再从细节上，考虑一些进一步的约束条件，如问题三所示，建立规划模型，研究设计每一天的补货与定价策略。最后，题目还将问题导向纵深，让人思考还需要采集哪些相关数据，以满足解决问题的需求。总之，本题是基于数据挖掘的一个考虑多层面统计因素影响的规划问题。

4.1 问题一分析

问题一要求分析蔬菜各品类以及单品销售量的分布规律以及相互关系。直观上来看，从各蔬菜之间的相互关系考虑，蔬菜各品类以及各单品之间，可能存在互补或者互斥的关系，即某些品类的蔬菜往往与另一些品类的蔬菜搭配，从而更加受到消费者的青睐；而又有某一些品类的蔬菜与其他品类的蔬菜功能或者定价上比较重复，导致消费者往往会只取其一。从时间的维度考虑，各品类蔬菜的销售量应该会呈现出一定的周期性与季节性，而不同的单品的销售量可能会有突变性。题目给出了 2020 年 7 月 1 日到 2023 年 6 月 30 日，各类商品的销售流水和各个品类的商品信息。于是可以画出各单品销售量随时间的变化折线图，从统计意义上总结各单品销售量的分布规律。各品类的销售量可以用品类内每一个单品销售量加和作估计，再画出随时间的折线图，总结分布规律。由于销售量随时间变化是连续型变量，我们采用皮尔逊相关系数来检验各单品以及各品类之间的相关性。由于时间序列数据的特性，我们还对品类数据采用平稳性检验的方法来研究其分布规律。

4.2 问题二分析

问题二要求研究各品类销售总量与成本加成定价的关系，并基于此给出各品类未来一周的日补货量和定价策略，使得收益最大。本问首先需要研究成本加成定价的含义，通过查阅资料可以得到相应的公式。对此，需要分别研究弹性销售量和利润率的关系，以及销售量变化与定价变动的关系。然后建立销售量变化与定价变动的函数，并基于此建立规划模型，以日补货量和定价作为决策变量，以七天总收益最大为目标，采用遗传算法求解得到结果。

4.3 问题三分析

问题三要求根据可售品种，在满足市场对于各品类需求的情况下，给出单品补货量和定价的策略，使得单日收益最大。首先需要根据条件挑选出 2023 年 6 月 24-30 日的可售单品，然后采用随机森林预测出 7 月 1 日的各品类需求量。仍然采用销售量变化与定价变动函数来做约束，最后建立规划模型，使用遗传算法得到当日的最大收益。

4.4 问题四分析

本文旨在通过考虑选取竞争对手的定价和补货量作为关键数据，来决定自身的定价和补货量。为了实现这一目标，本文采用了创新的方法，建立了竞争强度方程以及定价差与潜在顾客源变化之间的相关函数。通过这些模型，本文旨在评估与竞争对手定价的差异对于客源的影响。

通过对竞争强度进行量化，我们能够更好地了解市场上的竞争格局和力度。这种定量的评估方式可以帮助我们判断竞争对手在定价和补货量方面的优势和劣势。据此，我们可以制定相应的策略，以更好地应对市场竞争，扩大自身的市场份额。

同时，本文还构建了定价差与潜在顾客源变化之间的相关函数。这个函数可以帮助我们分析和理解不同定价策略对于客源的影响。通过对定价差与潜在顾客源变化的关系进行分析，我们可以更准确地把握定价对市场需求的影响，从而更有针对性地调整自身的定价策略。

五、 数据预处理

本题给出了 251 个单品蔬菜单品的商品信息与近期损耗律与三年共 1085 天的销售流水数据与批发价格，由于数据量比较多，题目所给出的销售数据，避免不了会出现噪声数据。如果不再次对数据处理，直接将其放入模型中，很大程度会增大预测的误差。需要对数据进行进一步的预处理，以便于下一步的分析。[1]

为保留单品销售量的突变型销售量，选择不对单品销售量进行异常值处理。对于蔬菜品类的销售总量，则选择将销售量异常的数据进行剔除处理。观察发现，蔬菜品类中存在销售量异常高的情况，如花叶类蔬菜一般的销售量在 500kg 以下，但是某一天的销售量竟达到 1200kg 以上，保留该数据会对进一步的预测处理造成影响，故予以剔除。

对于蔬菜品类的售价和批发价，也做同样的异常值处理。

六、 问题一的建模与求解

6.1 蔬菜品类与单品的分布规律

6.1.1 蔬菜单品的分布规律

现对销售量数据进行问题一要求分析蔬菜各品类以及单品销售量的分布规律以及相互关系。由于给出的是单品销售量的销售流水，首先画出单品销售量随时间的变化折线图进行观察，下面给出一些典型例子：

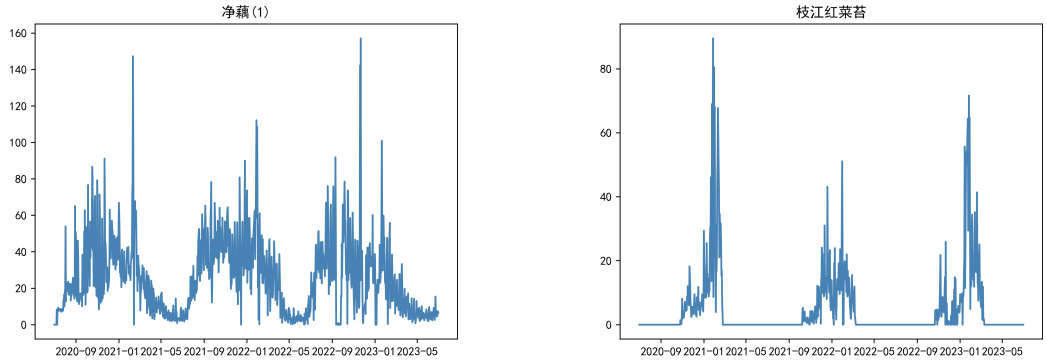

图 **1**　周期型蔬菜

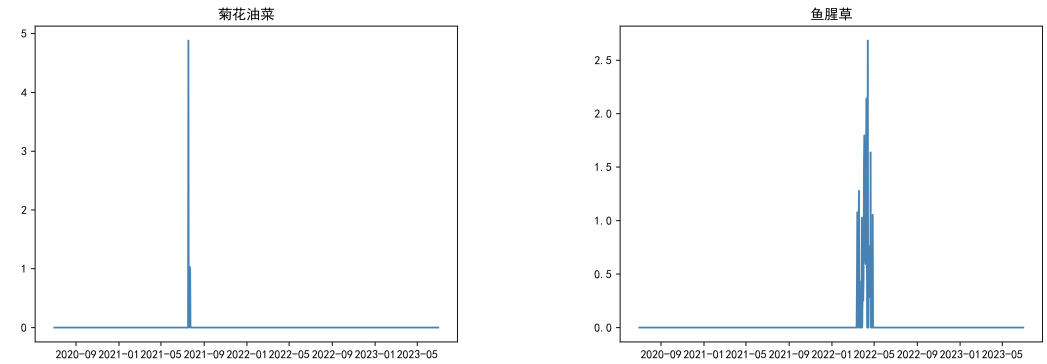

图 **2**　突变型蔬菜

经过大量观察比较后可知，蔬菜单品销售量随时间的变化存在以下三种分布规律，分别是周期型，突变型，与互补型。

如图1所示，净藕与枝江红菜苔就属于周期型蔬菜，其中净藕的周期波动比较大，有明显的旺季与淡季，属于季节性变化的蔬菜，而在单一季节中，又有小幅度的周期波动；而枝江红菜苔的周期性则呈现出明显的季节性，只在每年的 1 月前后，出现销售量的高峰。

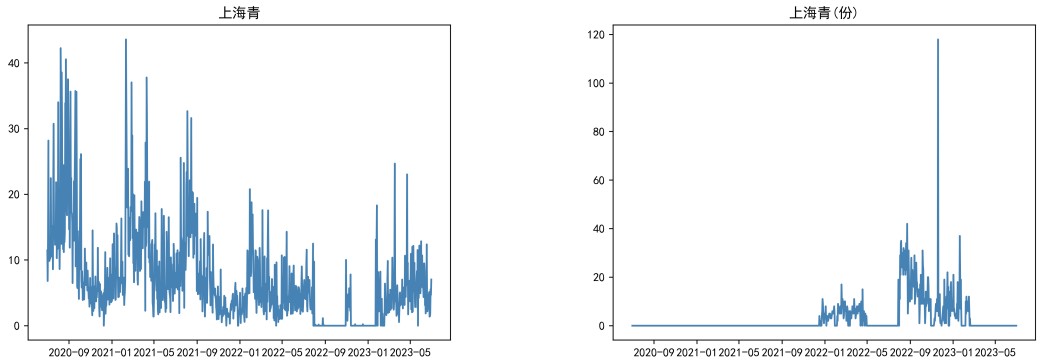

图 **3**　互补型蔬菜

又如图2所示，有菊花油菜与鱼腥草的销售分布属于突变型的数据。菊花油菜只在2021 年 7 月左右出售，而鱼腥草只在 2022 年 3-4 月出售，这可能是由于超市想要引进相应的蔬菜单品，但是由于效益不佳，于是果断放弃相关蔬菜的进购。上述类型的数据在下一步的预测中具有较大的不确定性，不能代表各品类的蔬菜的总体趋势。

如图3所示，上海青与上海青 (份) 存在明显的互补关系，上海青销售本来呈现一个比较良好的波动，但是在 2022 年的 9 月到 2023 年的 2 月这段时间，销售量却反常地接近零；反观上海青 (份)，本来销量为零，却恰好在这段时间里销售量激增。可以推测，这段时间里超市为了激发消费，将上海青拆分成份出售，导致原来地单品销售量接近于零。从统计预测的角度上看，可以将这两类蔬菜单品折合为一类单品。

所给数据中有不少互补型数据，我们试探将互补型数据合并，得到了比较合理的结果。如图4所示合成后的蔬菜销售量呈现出比较周期型的结果。于是在后续的研究中，我们将所有互补型蔬菜进行合并。

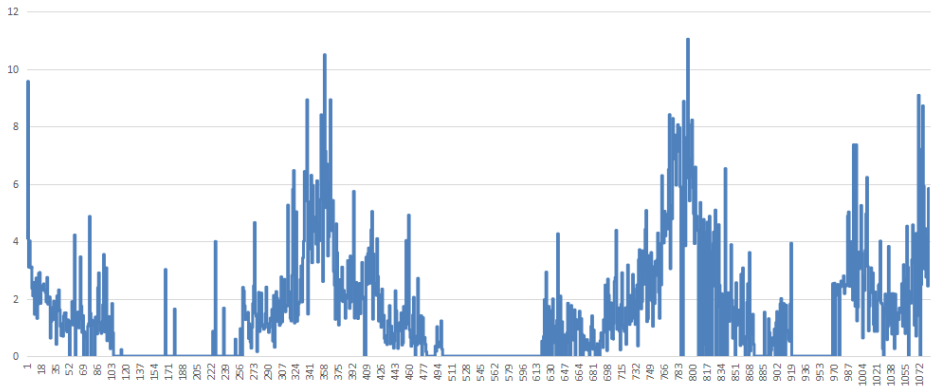

图 **4**　高瓜合成

6.1.2 蔬菜品类的分布规律

将蔬菜品类以内所有的单品的销售量加权相加，权重是每个品类中单品销售量的占比，得到关于蔬菜品类销售量的一个估计，并参照上文的做法，画出各蔬菜品类的销售量随时间变化的分布规律。

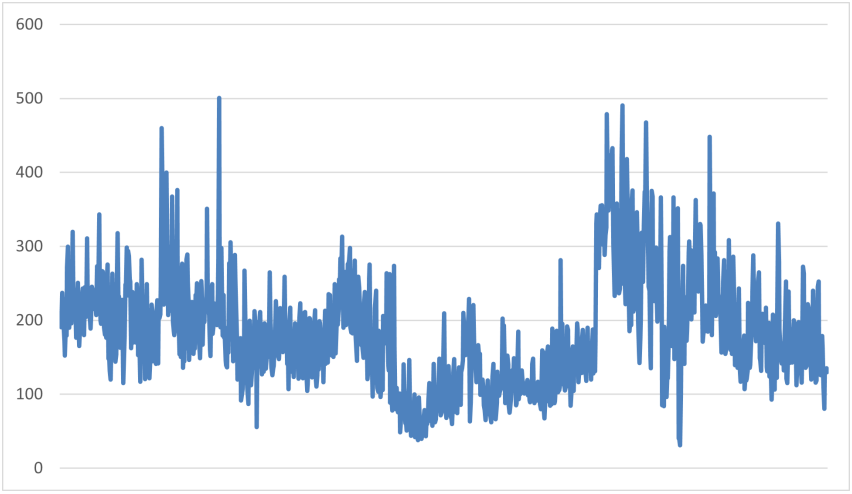

图 **5**　花叶类

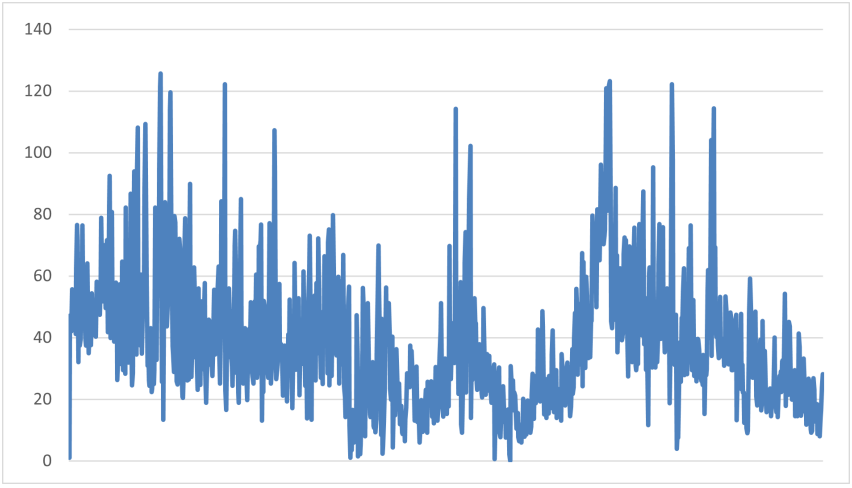

图 **6**　花菜类

如图所示，6 各品类蔬菜的销售量基本呈现出比较明显的周期性变化，有几个值出现异常高的情况，结合具体时间，可以推测是由于突发公共卫生事件如新冠疫情等，导致人们囤货行为而使得销售量激增。定性分析蔬菜品类的销售量，可以估计长期角度下蔬菜销售的时间序列应具有平稳性，于是我们对各品类蔬菜进行平稳性检验。

(1) 平稳性检验——ADF 检验

ADF 检验（Augmented Dickey-Fuller）检验是一种经济计量学中常用的单位根检验方法，用于检测时间序列数据是否具有单位根，判断一个时间序列是否是平稳的。进行

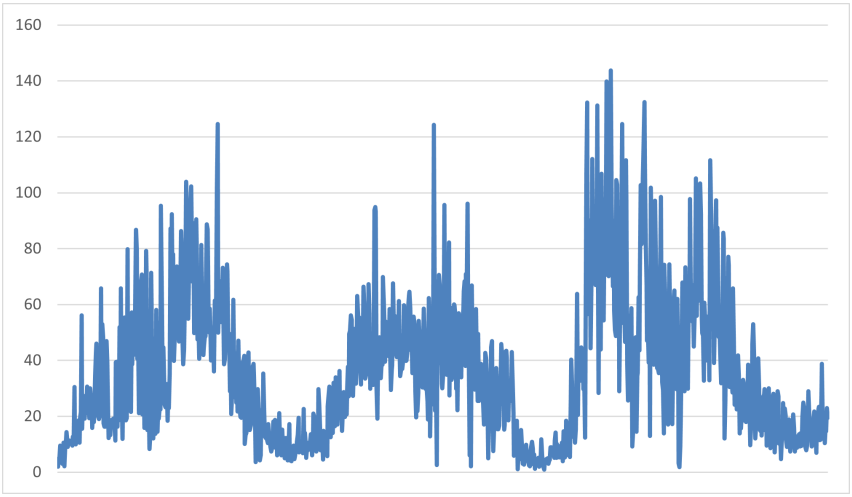

图 **7**　水生根茎类

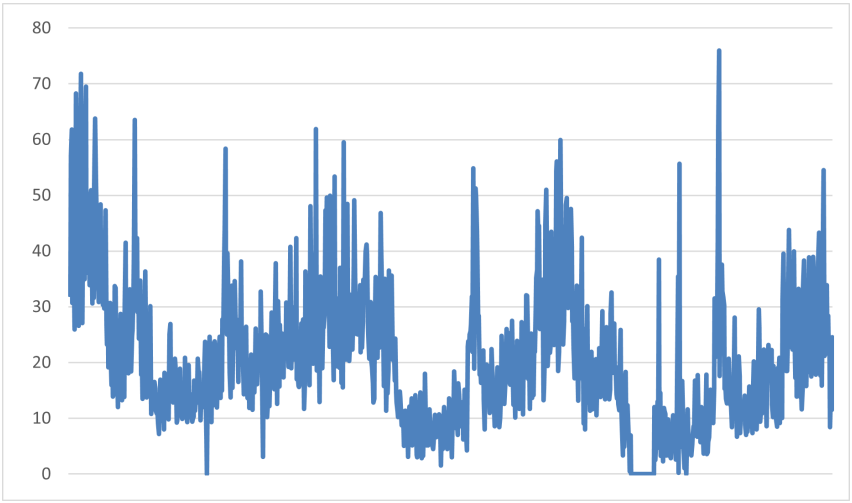

图 **8**　茄类

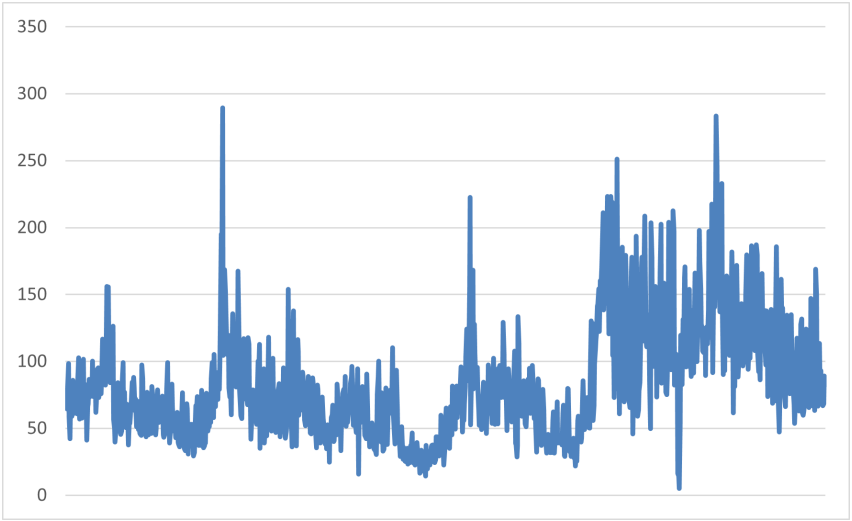

图 **9**　辣椒类

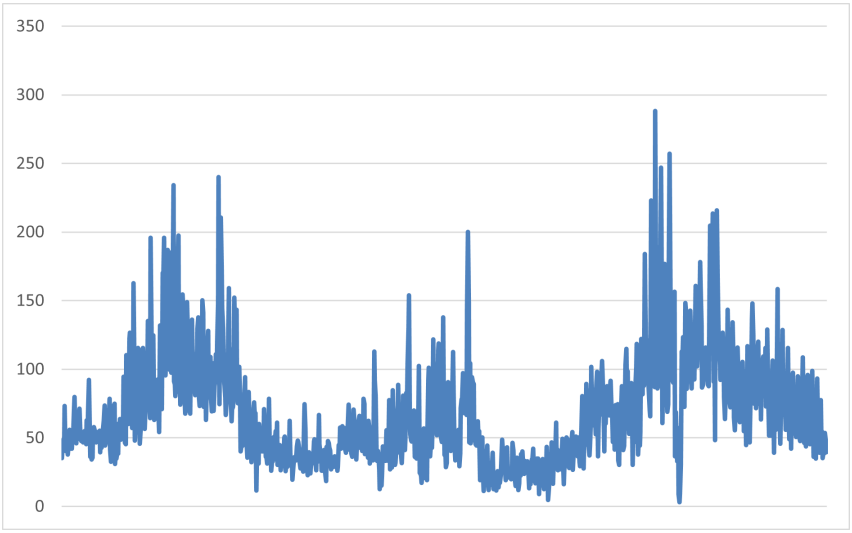

图 **10**　食用菌

ADF 检验时，将时间序列考虑为以下模型, 即 AR（p）模型

$$Y_t = \sum_{i=1}^{p} \beta_i Y_{t-i} + \varepsilon_t. \tag{1}$$

可以证明上述模型可以化为

$$Y_t = \rho y_{t-1} + \sum_{i=1}^{p-1} \varsigma_i \delta y_{t-i} + \varepsilon_t. \tag{2}$$

其中，$\rho = \sum_{i=1}^{p} \beta_i$。

提出假设 $H_0 : |\rho| \geq 1$ 和 $H_1 : |\rho| < 1$

检验统计量

$$t = \frac{\hat{\rho} - 1}{\sigma}.$$

其中，$\hat{\rho}$ 为 ρ 的估计，σ 表示该估计的标准差。t 满足一个特殊分布，通过计算机软件作 ADF 检验，可得到以下结果，如表1所示：

由表可知，在 0.05 的显著性水平下，大部分品类的 p 值都小于 0.05，故可以认为蔬菜品类随时间的变化是平稳的。而唯一 p 值超过 0.05 的花叶类的 p 值为 0.056，与 0.05 比较接近，考虑到数据收集过程中可能出现的误差，故可以认为花叶类的数据也基本满足平稳的假设。

6.2 各蔬菜品类与单品的相互关系

为考察各品类的销售量之间的相互关系，先画出各品类的堆积柱状图，以时间为横轴：由图11可以看出，花叶类蔬菜的占比在三年里始终保持一个较高的为位置，而茄类

表 **1**　**ADF** 检验结果

蔬菜品类	ADF 统计量	p 值
花叶类	-2. 82	0.056
花菜类	-3，13	0.024
水生根茎类	-3.10	0.026
茄类	-3.69	0.004
辣椒类	-3.63	0.005
食用菌	-3.55	0.007

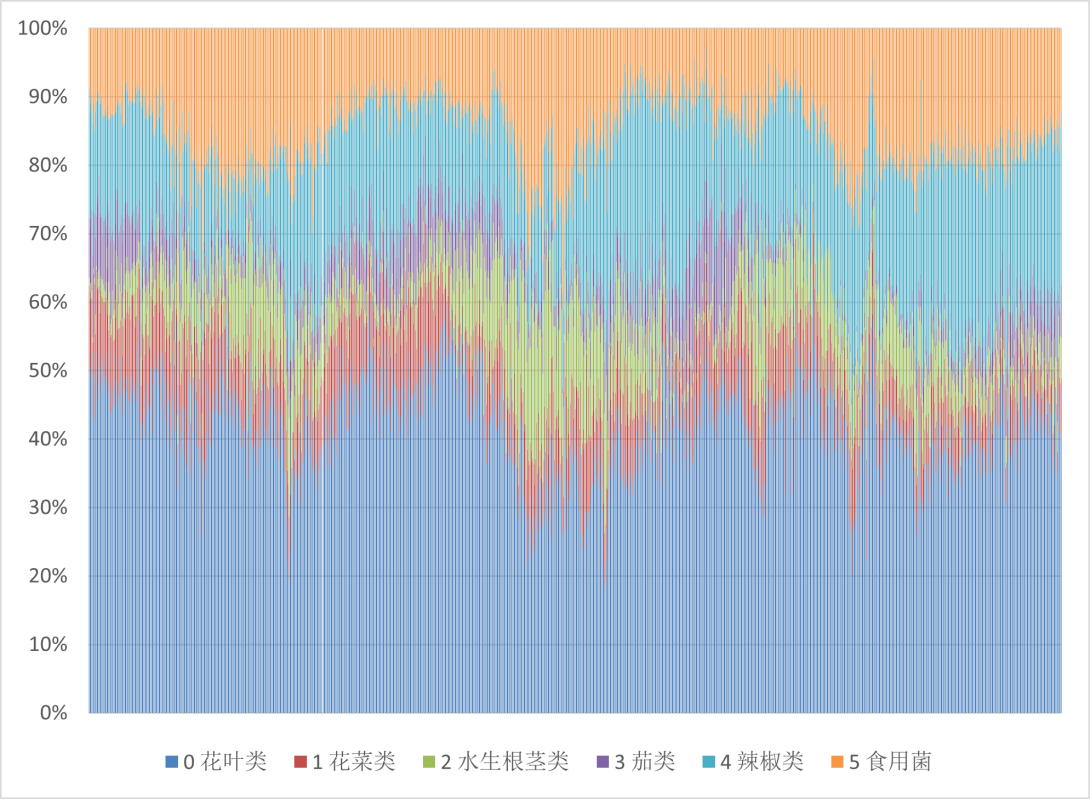

图 **11**　品类堆积百分比柱状图

和花菜类则保持一个较低的水平，从图中可以看出不同品类蔬菜销售量之间的相对差异。

为探究不同品类与单品蔬菜之间的相互关系，首先对数据进行标准化处理，消除数据量级的影响，由于数据分布未知，且为保留数据相对大小，我们采用数据 Min-max 标

准化，具体的计算公式为：

$$x^{'} = \frac{x - x_{\max}}{x_{\max} - x_{\min}}. \tag{3}$$

利用标准化之后的数据，进行相关性检验。由于数据的连续性，这里采用皮尔逊相关系数：

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

计算得到各蔬菜单品与品类的皮尔逊相关系数热力图，如图12和图13所示：

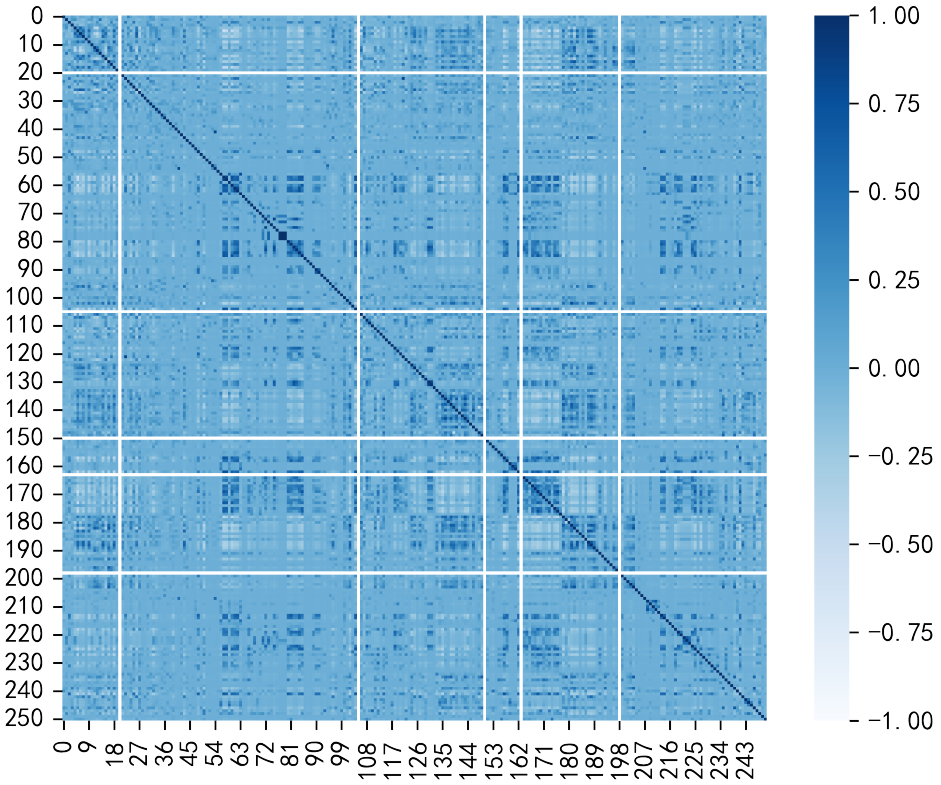

图 **12**　单品相关系数热力图

由图可知，大部分单品的相关性并不显著，但是有小部分单品之间具有比较明显的正相关性，而也有小部分单品之间存在负相关的关系。这与我们之前定性分析的结果一致，说明我们对相关性的分析是基本可信的。

对于蔬菜品类这里分别用 0 代表花叶类，1 代表花菜类，2 代表水生根茎类，3 代表茄类，4 代表辣椒类，5 代表食用菌。由图可知，茄类与其他 5 类的相关性都比较低，结合品类堆积百分比柱状图分析，可能是由于该商超所在地区并没有形成食用茄类蔬菜的广泛习惯，导致茄类蔬菜的销售量相对其他类型的蔬菜而言相对独立且占比较小，而辣椒类与食用菌类的相关系数比较高，可能是由于它们都不是国人食谱中的家常蔬菜，消费者对于这两类蔬菜的需求受到经济，生产周期等多方面因素的影响，会出现较大的波

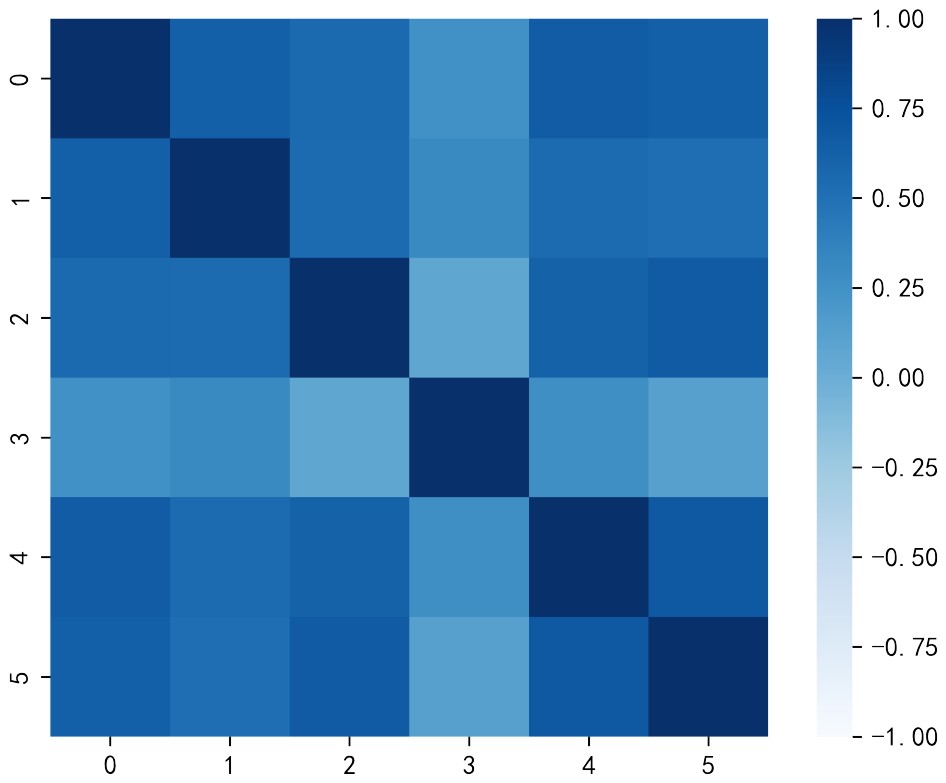

图 **13**　品类相关系数热力图

动，导致相关系数比较高。而花叶类和花菜类蔬菜则是家常菜居多，消费者对其需求一般不会有太大的波动。

七、 问题二的建模与求解

在问题一中，本文研究了不同蔬菜品类与单品之间的分布规律以及相互关系，发现各品类的销售量变化主要在时间上呈现一定的规律。问题二在问题一结论的基础上，进一步研究各品类销售总量的变化与成本加成定价的关系，并给出各蔬菜未来一周内，即2023 年 7 月 1-7 日的日补货总量和定价策略，使得商超的收益最大。

7.1 建模的准备

首先查询关于成本加成定价的资料，查询有关资料和百度百科后得知，成本加成定价法指产品价格要补偿生产和销售成本并得到合理回报。该理论假设销售者决定价格的主导权，购买者仅能影响加成率。[2] 使用成本加成定价有以下优点定价简单。不需要历史边际成本和边际收入信息，便于操作，生产商和顾客均易理解，有利于产销各方的合作；价格稳定。通常市场中，升高价格会招致顾客的抵制，降低价格则会带来竞争企业的恶性价格竞争，而成本加成定价法有利于价格稳定；便于价格领导。成本加成定价法

向竞争者传达的信息是成本增加需要提高价格，有利于合作定价。这对生产企业是极其有利的。

成本加成定价的模型可以归纳为：

$$P = C'(1 + w). \tag{4}$$

其中，P 是价格，C’ 是平均成本，w 是目标利润率。在完全成本加成定价中，C’ 可以通过历史数据估计。在这里我们采用以下方法估计：

$$C' = \frac{C}{M} + Q \tag{5}$$

其中，C 是商家的固定成本，M 是总补货量，Q 是单品的批发价。

由于上式中有太多变量未知，无法直接分析各个品类与成本加成定价的关系。我们先试探性地探究销售总量与销售价格之间地关系。这里以花叶类的销售总量与销售价格为例。其中，蔬菜品类的销售价格我们采取加权平均的形式，即考虑该品类中每一种单品在品类中的相对占比，以此为权重，计算品类的销售价格。

首先探究花叶类售价与销售总量之间的相关关系，使用皮尔逊检验法，利用 SPSSPRO 软件，我们得到以下热力图：

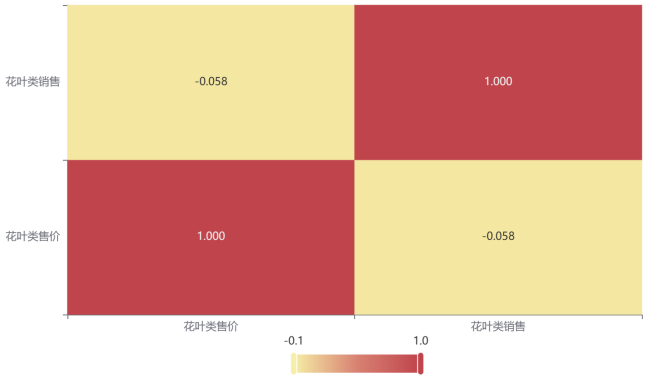

图 **14**　花叶类相关系数热力图

由图可知，两者不呈现出显著的相关关系。然而这可能是由于时间序列的滞后性导致的，于是我们进一步探究了二者之间的因果关系，所谓因果关系就是指一组时间序列是否会引起另一组时间序列的变化。

为探究两个时间序列的因果关系，我们计划采用格兰杰因果检验，但是格兰杰因果检验要求两个时间序列要是平稳时间序列。于是我们预先使用 ADF 检验法，对两个时间序列的平稳性进行探究。

对于花叶类的销售总量而言，在差分阶数为 0 时，算出 P 值为 0.033 左右，小于给定的显著性水平 0.05，故可以拒绝原假设，花叶类的销售总量是一个平稳的时间序列。

对于花叶类的售价而言，在差分阶数为 0 时，P 值为 0.063，大于显著性水平，故可以认为花叶类的售价是一个不平稳的时间序列。在差分阶数为 1 时，P 值小于 0.001，可以认为花叶类售价的一阶差分是一个非常平稳的时间序列。如下图所示：

图 **15**　花叶类售价原始序列

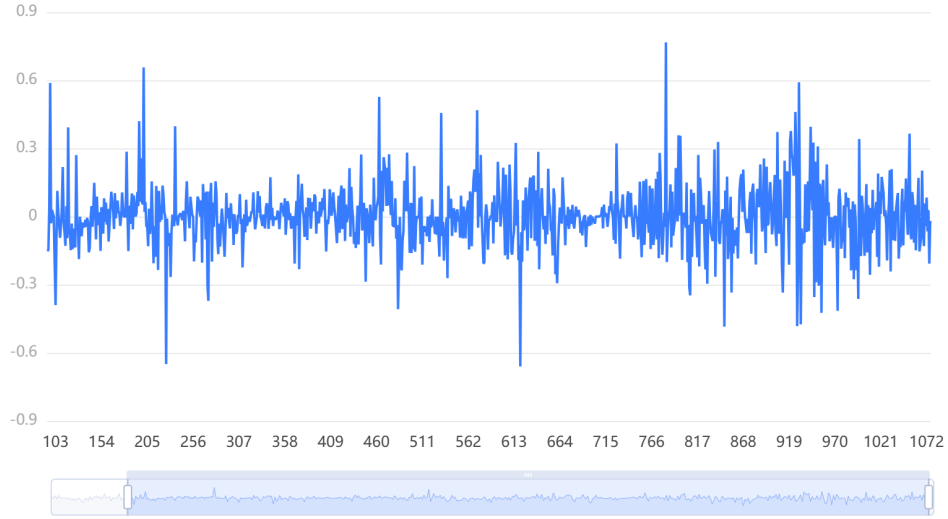

图 **16**　花叶类售价一阶差分

由于花叶类售价的原始序列经过 ADF 检验后，P 值为 0.063, 在显著性水平为 0.1 的情况下，也可以认为是平稳序列，即花叶类售价的不平稳性比较不显著。结合图像观测，花叶类的售价三年内整体呈现一个向上增长的趋势，不排除是由于物价增长导致的价格增加。

故可以对二者进行格兰杰因果检验，利用 SPSSPRO 进行计算。得到如下结果：

基于变量花叶类销售与花叶类售价，显著性 P 值为 0.036，呈现显著性，拒绝原假

表 **2**　格兰杰因果检验结果

配对样本		F	P
花叶类销售	花叶类售价	4.404	0.036
花叶类售价	花叶类销售	1.59	0.208

设，花叶类销售可以引起花叶类售价变化。基于变量花叶类售价与花叶类销售，显著性 P 值为 0.208，不呈现显著性，不能拒绝原假设，花叶类售价不可以引起花叶类销售变化。基于销售量可以引起价格变动，我们接下来探究二者之间的数量关系

7.2 模型的建立

我们分别探究利率的变化对于弹性销售量的影响，以及价格变动对于销售总量变化的影响。然后通过得到的结论，结合市场实际情况，建立各品类销售总量的变化与价格变动程度的函数。最后基于此建立规划模型，以 2023 年 7 月 1-7 日的总收益为规划目标，以各品类每日的补货量和定价为决策变量，采用遗传算法求解，得到连续七日的最大收益以及相应的决策结果。相关量的预测我们采用集成学习中的随机森林算法。

(1) 随机森林算法

随机森林是属于集成学习的算法，采用 Bagging 的思想，所谓的 Bagging 就是：

（1）每次有放回地从训练集中取出 n 个训练样本，组成新的训练集；

（2）利用新的训练集，训练得到 M 个子模型；

（3）对于分类问题，采用投票的方法，得票最多子模型的分类类别为最终的类别；对于回归问题，采用简单的平均方法得到预测值。

之所以选择随机森林算法，是因为随机森林算法具有许多不容忽视的优点。由于采用了集成算法，本身精度比大多数单个算法要好。由于两个随机性的引入，使得随机森林不容易陷入过拟合（样本随机，特征随机）且具有一定的抗噪声能力，对比其他算法具有一定优势。另一方面，随机森林可以处理非线性数据，本身属于非线性分类（拟合）模型。它能够处理很高维度的数据，并且不用做特征选择，对数据集的适应能力强：数据集无需规范化，训练速度快，可以运用在大规模数据集上。

(2) 遗传算法

遗传算法在求解规划问题时，是一个比较高效的算法，利用计算机的快速计算，迅速得到一个比较满意的结果。遗传算法的过程可以表示为如下：

遗传算法求解过程

Step 1 设定初始化状态，参数以及编码，导入数据；

Step 2 初始化，开始迭代进化；

Step 3 对可行解进行选择、重组、变异和合并;

Step 4 依据目标函数值的大小分配适应度值；

Step 5 计算出当代最优个体的序号；

Step 6 重复 3，4，5 三个步骤，直至进化到符合最优或达到遗传代数；

Step 7 解码输出。

第二问具体的思维导图如下图17：

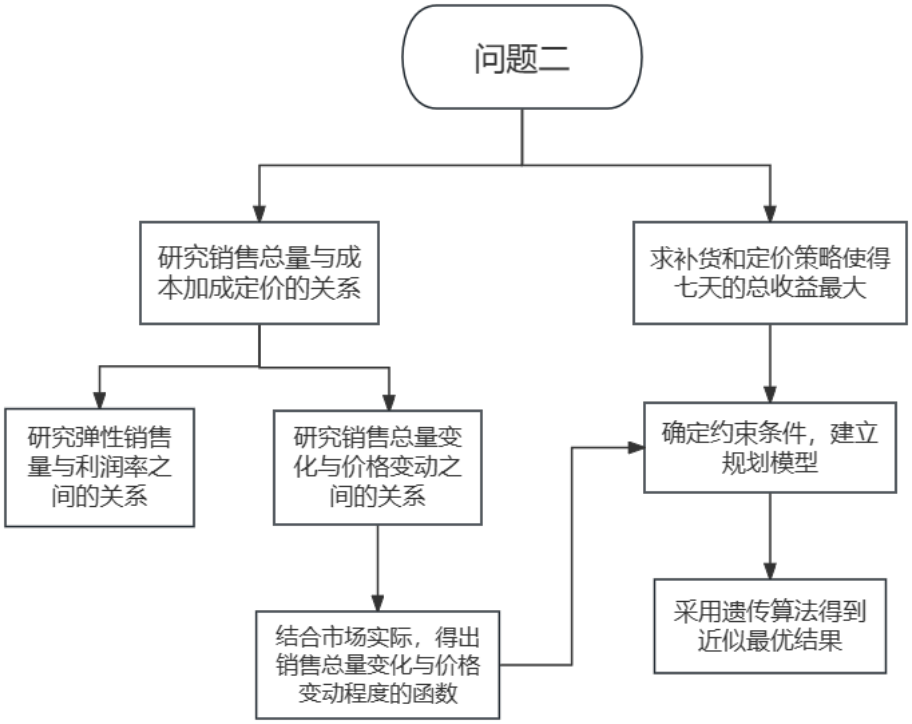

图 **17**　问题二思维导图

7.2.1 探究弹性销售量和利润率之间的关系

我们首先对商品的销售量的时间序列进行建模，我们认为蔬菜商品的销售量分为:

$$N_t = N_0(t) + N_\epsilon(t). \tag{6}$$

其中，N_t 为原始时间序列，$N_0(t)$ 表示价格随时间的长期变动趋势，它受到物价因素，市场宏观调控等因素的影响，在短期一定时间内应该呈现比较平稳的状态。N_ϵ 是短期影响因子，它应该受到短期内批发价，销售量与市场价的影响。我们称 $N_0(t)$ 为平衡销售量，$N_\epsilon(t)$ 为弹性销售量。

由于商品一段时间的平衡销售量与季节等因素有关，所以如果需要研究销售总量的变化与利润率的关系，需要消除时间因素的影响，得到减去固有销售量的基础上弹性销售量与利润率的关系。为了得到平衡销售量的数值，将各品类销售总量随时间的变化得到的折线图做平滑处理。

此处采用 LOESS 局部加权回归的平滑方式，其基本思路是对于原始数据的每个观测值，使用相邻的若干已知观测估计值得到的函数来估计，应用于全局之后得到整体的平滑曲线，以花菜类为例。

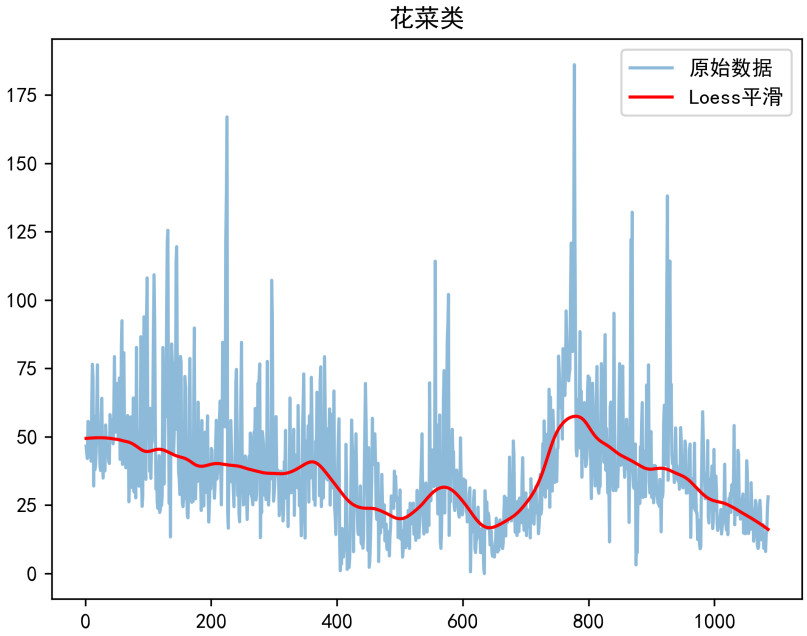

图 **18**　花菜类 **Loess** 平滑曲线图

得到六种品类弹性销售量随时间变化的序列后，我们做出品类每天利润率与弹性销售量的散点图19，以辣椒类为例。

由于成本加成定价有许多因素未知，这里利润率我们采用如下方法估计：

$$\omega = \frac{(P_t - Q_t)}{Q_t}. \tag{7}$$

可以看出，大量的点集中在零点附近，非常密集，无法看出相应规律。进一步线性拟合后我们发现大部分品类的利润率与弹性销售量成正相关关系，说明利润率越高反而销售量越高，这是不符合市场规律的，说明了人群的消费行为与销售者选择的期望利润

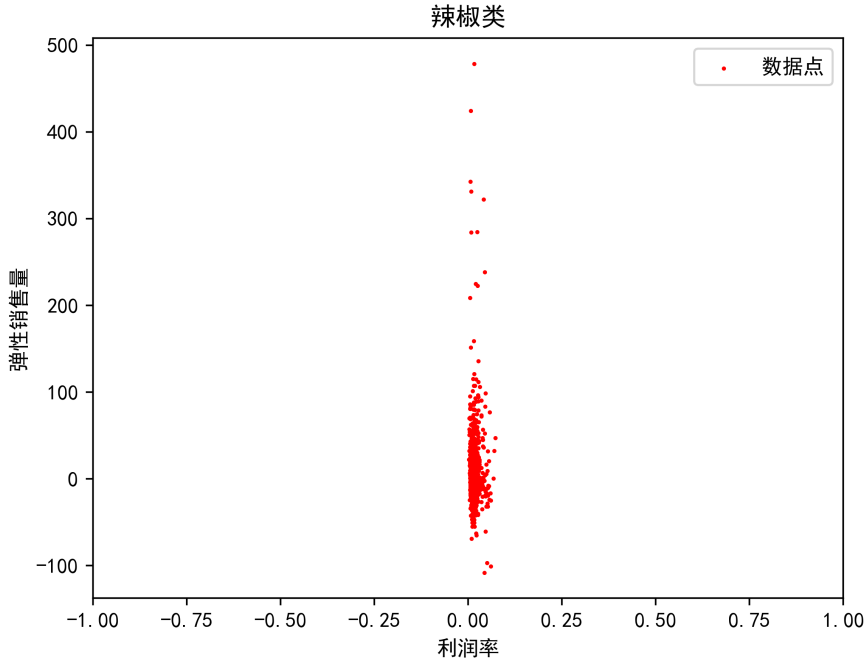

图 **19**　辣椒类利润率与弹性销售量的散点图

无关。但是推测人群对一个商品的销售意愿大概与商品的定价直接相关。于是，本文进一步研究品类销售总量的变化与价格变动的关系。

7.2.2 研究销售总量的变化与价格变动的关系

通过查阅资料，我们得知商品销售主要服从于需求价格弹性的规律：

$$E_{dp} = -\lim_{\Delta p \to 0} \frac{\frac{\Delta Q_d}{Q_d}}{\frac{\Delta P}{P}} = -\frac{dQ_d}{dP}\frac{P}{Q_d}. \tag{8}$$

上式说明：当天定价相对于前一日定价的变化会对销售量产生负影响。亦即，价格比昨天更高，人们就更不愿意买；价格比昨天更低，人们就更愿意买。为了研究具体的数量关系，本文做出每个品类连续两天的定价变化与销售总量变化的散点图，进行线性拟合，观察相应规律。

得到的结果图如下图20所示，以水生根茎类为例：

可以看出除了花叶类意外，其他散点图都呈现销售量变化对于定价的变化呈现负相关的关系，符合客观经济规律。说明定价变化确实显著地影响了销售量的变化。对于花叶类，它的 α 值计算后呈现正相关，拟合效果比较差。由之前的探索可知，可能是因为花叶类蔬菜是相对大宗的商品，每天人群对其的需求量几乎稳定，在此基础上会有轻微的无规律波动，所以无法得到花叶类销售总量与成本加成定价的具体关系。

在前一部分的讨论中，我们得到了除花叶类外，每个品类销售量变化与定价变动的大体关系，在本问的规划模型中，为了增加对定价和需求量估计的约束（用销售量估计

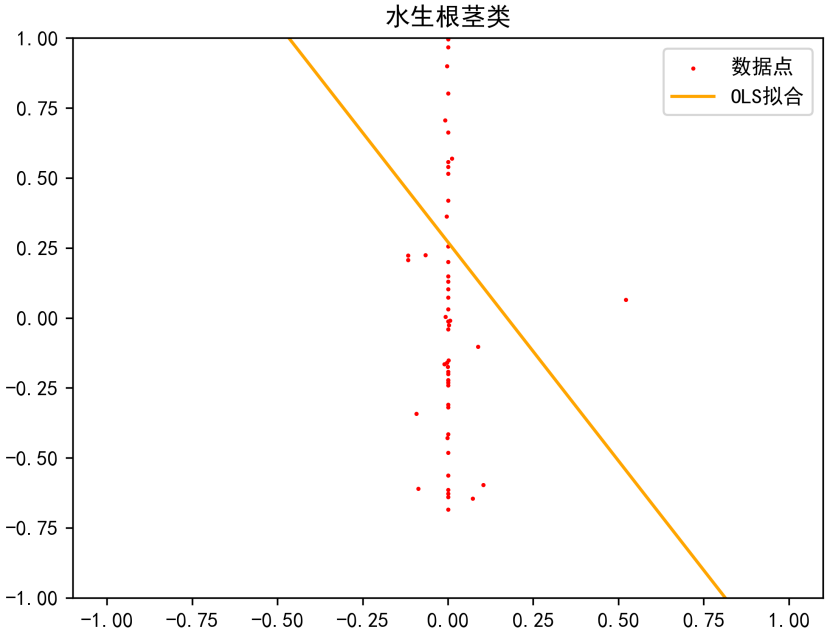

图 **20**　水生根茎类定价销售拟合图

需求量），必须要设定准确的定价变动对销售量变化的影响描述。如果直接采用线性拟合的结果，就会有很强的不合理性，比如当定价很高时会出现销售量变为负的情况，当定价很低时会出现销售量无限制增长的情况。为此需要更换模型进行描述。

考虑到当价格变动不大时，销售量变化与定价变化线性相关性很强。而且根据实际情况，当价格很高时，应该会无人购买；价格低到一定程度后需求量会趋于平稳。因此，本文采用以下函数描述定价变动与销售量变化的关系（其中花叶类不适用该函数表达）：

$$dQ_t = \begin{cases} -\frac{1}{\pi} arcsin(dP_t), dP_t < 1.376 \\ -0.3, x \geq 1.376 \end{cases} \tag{9}$$

其中

$$dQ_t = \frac{Q_t - Q_{t-1}}{Qt - 1}.$$

$$dP_t = \frac{P_t - P_{t-1}}{P_{t-1}}.$$

7.2.3 确定规划模型求解七天的最大收益

(1) 模型的决策变量

本问需要指定相应的补货策略和定价策略，所以决策变量为每个品类每天的补货量 M_{it} 和每天的定价 P_{it}。其中花叶类的每天定价使用以往数据的预测值。

(2) 模型的约束条件确定

(a) 价格变动对于需求量变化的影响

由于上文给出的是价格变动对于销售量变化的影响，而此处需要用到需求量的概念。于是我们用销售量的作为需求量的估计值，于是得到价格变动对于需求量变化的函数关系，关系如上式(9)

(b) 品类损耗率和打折率的估计

根据对数据的观察，我们得到每个品类的折合损耗率（加权结果）以及对打折率的估计:

表 **3**　品类折合损耗律

品类	折合损耗率
花叶类	12.5472919
花菜类	10.87422379
水生根茎类	8.243525238
茄类	6.358469636
辣椒类	8.059162223
食用菌	6.824567767

经过大量观察数据，我们确定将打折率设为 6 折，便于计算。

(c) 每日剩余：

当每日的补货量大于需求量时会产生剩余，公式为：

$$S_{it} = M_{it} - Q_{it}. \tag{10}$$

其中 S_{it} 是剩余量，M_{it} 是补货量，Q_{it} 是需求量。由于第一天未售出的商品第三天不会继续出售，所以式中不含前一天的销售量。

(d) 每日批发价

我们认为批发价对于市场价格短期内不会过于敏感，故可以用预测值进行同意估计，这里记为 C_{it}. 表示第 i 品类第 t 天的批发价，并使用集成学习算法进行预测。

(e) 每日收益

$$Pr_{it}=\begin{cases}0, & Q_{it}<0\\ Q_{it}\theta P_{it}-C_{it}M_{it}, & S_{i(t-1)}>Q_{it}\\ S_{i(t-1)}\theta P_{it}+(Q_{it}-S_{i(t-1)})P_{it}[(1-l_i)+l_i\theta]-C_{it}M_{it}, & Q_{it}-M_{it}<S_{i(t-1)}<Q_{it}.\\ S_{it}\theta P_{it}+M_{it}P_{it}[(1-l_i)+l_i\theta]-C_{it}M_{it}, & S_{i(t-1)}\le Q_{it}-M_{it}.\end{cases} \tag{11}$$

7.2.4 目标函数与公式的汇总

目标函数：

$$max\left\{\sum_i\sum_t Pr_{it}\right\} \tag{12}$$

约束条件：

$$s.t.\begin{cases} Pr_{it}=\begin{cases}0, & Q_{it}<0\\ Q_{it}\theta P_{it}-C_{it}M_{it}, & S_{i(t-1)}>Q_{it}\\ S_{i(t-1)}\theta P_{it}+(Q_{it}-S_{i(t-1)})P_{it}[(1-l_i)+l_i\theta]-C_{it}M_{it}, & Q_{it}-M_{it}<S_{i(t-1)}<Q_{it}.\\ S_{it}\theta P_{it}+M_{it}P_{it}[(1-l_i)+l_i\theta]-C_{it}M_{it}, & S_{i(t-1)}\le Q_{it}-M_{it}.\end{cases}\\ S_{it}=M_{it}-Q_{it}.\\ dQ_t=\begin{cases}-\dfrac{1}{\pi}arcsin(dP_t), dP_t<1.376\\ -0.3, x\ge 1.376\end{cases}\end{cases} \tag{13}$$

7.3 规划模型的求解与分析

由于本问题是非线性多约束的规划模型，于是本文采用遗传法求出近似最优解。算法求解本模型的迭代过程如下：

收益和决策变量的结果详见附录。

从结果可以看出，每天的每种品类的定价都较为稳定，订货量也较为稳定。说明为了使收益最大，就不能随意变动价格，稳定的供求关系保证了收益的最大化。

八、 问题三的建模与求解

问题三要求根据 2023 年 6 月 24-30 日的可售品种，给出 7 月 1 日的单品补货量和定价策略，在尽量满足市场对各品类需求和商场销售空间的约束的前提下，使得商超的收益最大。这里我们使用规划的思想构建数学模型，再使用遗传算法求解。

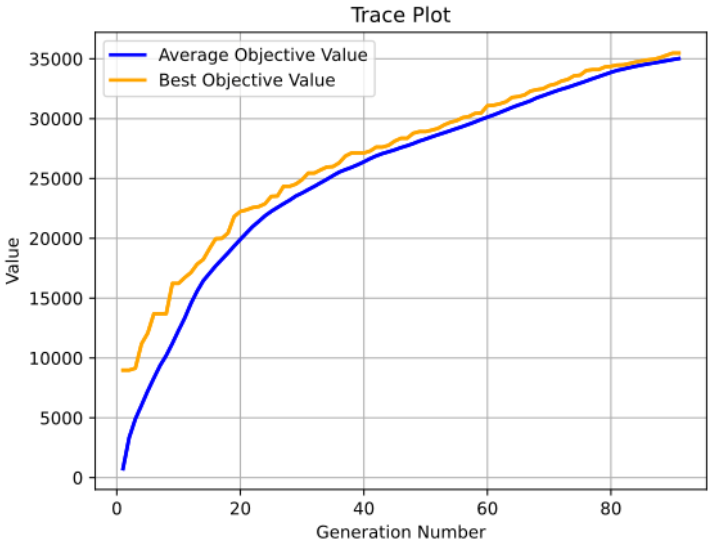

图 **21**　算法迭代图

8.1 建模前的准备

首先筛选出 2023 年 6 月 24-30 日可售的蔬菜单品，其编号如下表4所示 我们经过深思熟虑考虑了上述组合，是出于以下几点考量：

1. 所示单品大部分是在 2023 年 6 月 24-30 日中的可售单品。
2. 考虑到蔬菜品类的突变型，我们特意从往年同期中找寻具有突变销售的蔬菜单品，它们虽不在 2023 年 6 月 24-30 日可售，但每年夏天都会出现销售量激增，不能排除其在 7 月 1 日可售的可能性。
3. 某一些蔬菜只在 2023 年 6 月 24-30 日出售，其互补型蔬菜却在其他时间段有售出，考虑到在预测时我们需要考虑三年的变动，故我们将互补型蔬菜当作同一类单品进行预测，在决策时再取近期可售的单品进行决策。

8.2 模型的建立与求解

8.2.1 决策变量的选取

本问需要研究使得日收益最高的补货量和定价决策，因此选择单品补货量 m_{it} 和单品定价 P_{it} 作为决策变量。

8.2.2 约束条件的确定

2.1 满足市场对各品类蔬菜商品的需求

在此假设 7 月 1 日之前没有存货，那么需要 7 月 1 日的每个品类补货总量都要大于等于市场需求量。其中各品类的需求量由预测得知，实际可能由于价格的影响而不是这

表 **4**　可售单品编号

花叶类	花菜类	水生根茎类
102900005115762	102900005116714	102900005116899
102900011030059	102900011034026	102900005118824
102900005115786		102900051000944
102900005115823	茄类	食用菌
102900005115908	102900011022764	106949711300259
102900005115946	102900051000463	102900011031926
102900011030097	辣椒类	106971533450003
102900011033920	102900005116257	102900011031100
102900011030110	102900005116509	102900011034330
102900005118831		102900011032022
102900005119975		102900011016701
102900011006948		102900011032251
102900011008164		102900011032237
102900011023464		
102900051010455		

个值，但是这个条件是最根本的要保证的。

$$\sum_{j} m_{ij} \geq Q_i, \tag{14}$$

2.2 满足最小陈列量的要求

为了发挥商场空间的经济效益，有必要要求每个单品的进货量都大于一个值。

$$2.5 \leq m_{ij} \tag{15}$$

2.3 符合市场价格需求变动的规律

当日定价会对实际单品需求量有所影响，因此继续考虑前文的销售量变化与定价变

动的函数作为约束。

$$Q_{ijt} = (1 + f(\frac{P_{ijt} - P_{ij(t-1)}}{P_{ijt}}))Q_{ij(t-1)}, f(x) = \begin{cases} -\frac{1}{\pi}tan(x), & x < 1.376 \\ -0.3, & x \geq 1.376 \end{cases}$$

8.2.3 目标函数的建立

本问要求最大收益，于是确定目标函数如下：

$$max\left\{\sum_i \sum_j Pr_{ij}\right\}$$

其中当日单品补货量大于需求量时，只能卖出需求量的值，此时由于会优先卖出无损耗的商品，所以不考虑打折处理的影响。而当单品补货量小于需求量时，就要用补货量计算收益，并且需要考虑损耗商品的影响。

规划模型可以总结如下 **目标函数：**

$$max\left\{\sum_i \sum_j Pr_{ij}\right\} \tag{16}$$

约束条件：

$$s.t. \begin{cases} \sum_j m_{ij} \geq Q_i, \\ 2.5 \leq m_{ij} \leq 150, \\ \textbf{其中:} \\ Q_{ijt} = (1 + f(\frac{P_{ijt} - P_{ij(t-1)}}{P_{ijt}}))Q_{ij(t-1)}, \\ f(x) = -\frac{1}{\pi}tan(x), \qquad x < 1.376 \\ -0.3, \qquad x \geq 1.376 \\ Pr_{ij} = \begin{cases} Q_{ijt}P_{ijt} - C_{ij}m_{ij}, & m_{ij} \geq Q_{ijt}, \\ m_{ij}P_{ijt}l_{ij}(1-\theta) - C_{ij}m_{ij}, & m_{ij} < Q_{ijt} \end{cases} \end{cases} \tag{17}$$

8.3 模型的求解

本问继续采用遗传算法求解，得到最终的最高收益为。。。决策变量的取值为。。。（给出遗传迭代图）（给出收益和决策变量的表格）

8.4 结果的评价与分析

最终得到的收益是一个相对合理的值。但是观察决策变量数据发现，很多商品的定价都很高，说明对定价高导致需求量减少的程度设定可能不够，需要改进销售量变化与

定价变动的函数。此外还有个别定价较小的单品，可能是由于对前段时间定价预测的不合理导致的。最后观察发现，进货量的数据都相对合理。

九、 问题四的拓展与思考

问题四拓展问题的维度，引发思考：为了更好地制定蔬菜商品地补货与定价决策，商超还需要采集哪些相关数据？并且思考这些数据对解决上述问题有何帮助，给出意见与理由。

9.1 历史补货量与固定成本

结合上文讨论，我们认为历史补货数据和商超固定成本是一个决策中很重要的数据支撑。由于我们认为成本加成定价的计算公式应为

$$P = (\frac{C}{M} + Q)(1 + \omega).$$

其中，P 为单品蔬菜价格，C 为固定成本，M 为总进货量，Q 为单品批发价，ω 为目标利润率。

为了探究蔬菜售价与批发价的关系，我们还需要知道该商超的历史补货量以及固定成本。在上文中，我们仅仅通过预测或通过假设，隐去了这个因素的影响。实际问题中，这个未知量对于模型的结果可能有很大的影响。通过固定成本与历史补货的数据，再加上已知数据，我们可以清晰地刻画目标利润率随时间地变化，从而更好地作出策略。

9.2 同行竞争者的定价数据

市场经济的竞争关系是长期存在的，上文中规划问题的解决由于缺乏同类型超商（竞争对手）定价数据，所以没有考虑到竞争关系的约束。事实上，销售者在制定定价策略时会受到同行竞争者定价的影响。比如对于具有重合服务范围的 A、B 两个商超，如果 A 商超的某个商品的价格低于 B，那么位于重合服务范围的消费者会更倾向于选择去商超 A 购买该商品。这也就是俗话说的薄利多销。如果定价比竞争者高就会出现相反的情况。为了更具体的描述这种竞争带来的影响，我们需要细化研究下面几个问题。

9.2.1 服务范围的重合程度对于竞争因素的影响

服务范围是指某种企业或者商店接受服务的人口在地域空间上的分布范围。当多个超商的服务范围出现重合区域时，位于重合区域的消费者就会面临选择问题。

重合区域的大小直接决定了商超之间的竞争强度，重合区域越大的两个超商，其竞争也会越强烈。当竞争特别强烈的情况下，往往会出现商品低价贱卖以换取客源量的情况。这个时候得知竞争对手的定价策略就显得尤为重要。假设竞争强度用 Str 表示，重

合区域大小用 a 表示，那么可以得到竞争强度关于重合区域的一个函数 Str=F(a)。其中重合区域于竞争强度是正相关的，具体的函数形式需要结合具体数据确定。

9.2.2 潜在顾客变化量

当商超的某件商品定价比竞争对手低时就会出现潜在的顾客源收益，定价比竞争对手高时就会有潜在的顾客源损失。为了估计增益或损失的具体量，我们构造如下公式来描述：

$$\Delta R = Str \times T(\Delta p) \tag{18}$$

其中 ΔR 为顾客源变化，Δp 价格差，T 为待建函数。

由于价格差异不大时，不会有太大的顾客源变化，敏感性较低；价格差异大到某一程度的时候顾客源变化较大，敏感性很强；价格差异特别大的时候，也不会引起顾客源太强烈的变化，敏感性不会很高。因此采用 Logistic 回归方程来描述这个函数较为合理。

$$T(\Delta p) = \begin{cases} -(\dfrac{e^{\theta+\beta\Delta p}}{1+e^{\theta+\beta\Delta p}} - b), \Delta p \geq 0 \\ \dfrac{e^{\theta+\beta\Delta p}}{1+e^{\theta+\beta\Delta p}} - b, \Delta p < 0 \end{cases} \tag{19}$$

9.2.3 潜在顾客源变化带来的收益

一方面定价比竞争对手低可以带来潜在的顾客源增益，另一方面降价出售也会使得每件商品的利润有所降低。反之，高价出售一方面增加商品利润，另一方面也减少了潜在客源。为此需要根据竞争对手的定价制定合理的定价策略，使得收益最大，下面给出了价格差得到潜在收益的函数。

$$Pr_0 = (N + \Delta R)(P + \Delta P - C) \tag{20}$$

其中，N 为库存需求，P 为定价，C 是成本，Pr_0 是潜在收益。

对于此需要挖掘更多的数据得到方程的确切形式以及内部的相应参数，最终利用竞争者的定价数据求得收益最大的结果。

9.2.4 获取竞争者的补货数据研究消费者对不同商品的需求度

对于那些其他竞争者补货量很高的商品，可以断定该商品销路应该是很好的。因此超商决策者可以选择大量购入该商品，加入到同行的竞争行列中。但是这会面临卖不出去的风险（竞争失败）

补货量较低的商品会有两种情况，一种是消费者需求量不高的商品，也可能是存在销售风险导致其他竞争者不敢购进的商品。对于前者，当然最好不要大量购进。对于后者，销售者可以尝试补货后投入市场，有可能正好满足了消费者的需求空缺，获得可观的利润，最后形成服务特征分化。

十、 模型评价与改进

从总体上看，针对蔬菜自动补货与定价策略，我们以销售量和售价的关系为切入点，引入时间序列刻画销售量与售价的变动，再通过相关性分析得到不同单品之间与不同品类之间销售量的相互关系，得到一个比较完全的对于问题的理解。再利用规划的思想，对商超决策进行合理设计，得到一个不错的结果。但由于某方面的数据缺失，我们的结果肯定存在与现实的某些偏差，这些是不可避免的。现实中，蔬菜的补货与定价策略可能还要考虑市场竞争。并且，蔬菜生产很大程度上甚至受到气候的影响，这就导致价格的波动可能不会完全符合市场规律，呈现出一定的随机性，给我们模型的应用性带来很大的挑战。

10.1 模型的优点

我们模型的优点在于：

1. 充分探索了销售总量与成本加成定价的关系；
2. 创造性地给出了销售总量与成本加成定价之间的函数关系；

10.2 模型的缺点

我们的模型当然也有很大的不足，这体现在：

1. 对于模型的求解进行了较多的简化；
2. 数据的选取比较特殊，可能不具备可推广性。

参考文献

[1] 张鹤. 基于集成学习的蔬菜销售预测 [D]. 云南师范大学,2021.DOI:10.27459/d.cnki.gynfc.2021.000144.

[2] 韩俊华, 干胜道. 成本加成定价法评介 [J]. 财会月刊,2012(22):74-75.DOI:10.19641/j.cnki.42-1290/f.2012.22.034.

十一、 附录

1. 附录 A：支撑文件列表
2. 附录 B：中间过程
3. 附录 C-E：源代码

附录 A　支撑文件列表

1. C 题.zip ；包含题目原材料。
2. 图片文件夹：包含论文中的所有图片。
3. 源代码：
4. 参考文献：包括《成本加成定价法评介》,《基于集成学习的蔬菜销售预测》

附录 B　问题二与问题三中间过程

表 **5**　**Add caption**

品类	7 月 1 日	7 月 2 日	7 月 3 日	7 月 4 日	7 月 5 日	7 月 6 日	7 月 7 日
花叶类	90.2608	100.3164	97.7327	98.2163	114.6566	112.2091	21.4253
花菜类	31.5796	36.7457	46.7558	4.5789	77.9452	107.2253	100.3457
水生根茎类	17.3549	27.5502	46.5906	48.9357	32.9216	46.4923	43.8787
茄类	52.5542	60.4253	87.0976	113.8974	93.7039	96.8000	102.8704
辣椒类	106.7602	108.2445	117.6981	102.3052	103.3915	119.3568	96.5662
食用菌	80.9436	74.3386	71.7331	33.4153	110.3852	103.2091	111.8575

品类	7 月 1 日	7 月 2 日	7 月 3 日	7 月 4 日	7 月 5 日	7 月 6 日	7 月 7 日
花叶类	5.7939	5.7939	5.7939	5.7939	5.7939	5.7939	5.7939
花菜类	15.9602	8.5137	19.9033	5.6220	13.0999	19.6387	19.1747
水生根茎类	19.9619	11.9822	19.9062	18.3946	13.1110	16.1419	18.6547
茄类	16.1898	7.8692	18.3006	18.6899	17.2636	18.7340	18.3099
辣椒类	17.5073	18.4382	19.5841	16.2452	19.0893	19.7278	18.8032
食用菌	18.6907	19.2646	19.0429	4.4686	9.7738	19.9105	18.4237

附录 C　问题一代码

表 6　Add caption

品类编号	定价	补货量
102900005115762	14.2561	12.8658
102900011030059	14.5280	59.2776
102900005115786	14.9647	19.1954
102900005115823	10.3946	6.2868
102900005115908	14.6304	2.9594
102900005115946	14.3540	9.1672
102900011030097	14.5723	29.5392
102900011033920	13.0982	48.9917
102900011030110	14.8294	40.5568
102900005118831	14.3399	14.1465
102900005119975	13.6256	11.0209
102900011006948	2.9799	2.7121
102900011008164	14.0656	14.5308
102900011023464	14.7814	8.9879
102900051010455	11.3039	8.9291
102900005116714	14.6165	10.3535
102900011034026	13.6272	40.6321
102900005116899	14.8269	7.5197
102900005118824	14.3953	9.1582
102900051000944	14.6901	2.7514
102900011022764	12.0622	4.9925
102900051000463	7.8187	4.5392

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from sklearn.preprocessing import MinMaxScaler

plt.rc('font', family='SimHei')
plt.rc('axes', unicode_minus=False)

# 读取文件
data1 = pd.DataFrame(pd.read_excel('data/附件1.xlsx'))
data2 = pd.DataFrame(pd.read_excel('data/附件2.xlsx'))

# 建立单品编号与单品类别的映射字典
data1_map = data1.set_index('单品编码')['单品名称'].to_dict()
data1_category = {}
category1 = []
category2 = []
category3 = []
category4 = []
category5 = []
category6 = []
for i in range(data1.shape[0]):
if data1.iloc[i, 2] == 1011010101:
category1.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010201:
category2.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010402:
category3.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010501:
category4.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010504:
category5.append(data1.iloc[i, 0])
else:
category6.append(data1.iloc[i, 0])

data1_category['花叶类'] = category1
data1_category['花菜类'] = category2
data1_category['水生根茎类'] = category3
data1_category['茄类'] = category4
data1_category['辣椒类'] = category5
data1_category['食用菌'] = category6

# 统计各个单品每日的销售量和定价
item_num = data1.iloc[:, 0]
```

```
# 记录时间戳
times = []
time = data2.iloc[0, 0]
single_day = {}
single_sale = {}

for i in range(len(item_num)):
single_sale[item_num[i]] = 0
for i in range(len(item_num)):
single_day[item_num[i]] = 0

data11 = pd.DataFrame(pd.read_excel('data/销售量统计.xlsx'))
data14 = data11.copy()
count = 1

for i in range(data2.shape[0]):
temp_time = data2.iloc[i, 0]

if time != temp_time:
time = temp_time
for j in range(data11.shape[0]):
data11.iloc[j, count] = single_day[data11.iloc[j, 0]]
data14.iloc[j, count] = single_sale[data11.iloc[j, 0]]
for k in range(len(item_num)):
single_day[item_num[k]] = 0
count += 1

single_day[data2.iloc[i, 2]] += data2.iloc[i, 3]
single_sale[data2.iloc[i, 2]] = data2.iloc[i, 4]

if i == data2.shape[0] - 1:
for j in range(data11.shape[0]):
data11.iloc[j, count] = single_day[data11.iloc[j, 0]]
data14.iloc[j, count] = single_sale[data11.iloc[j, 0]]

if temp_time not in times:
times.append(temp_time)

data11.to_excel('result/problem1/单品单日销售量统计.xlsx')
data14.to_excel('result/problem2/单品单日定价统计.xlsx')

# 绘制每个单品的销售量随时间变化图
for i in range(251):
plt.figure(num=i)
plt.plot(times, data11.iloc[i, 1:], c='#4682B4')
title = data1_map[data11.iloc[i, 0]]
```

```
plt.title(title)
plt.savefig('result/problem1/pictures1/' + title + '.png', dpi=400)
plt.close()

# 统计每个品类每天的销售量
category = list(data1_category.keys())
categories = {}
for i in range(len(category)):
categories[category[i]] = 0

data12 = pd.DataFrame(pd.read_excel('data/品类统计.xlsx'))

for j in range(1, data11.shape[1]):
for i in range(data11.shape[0]):
cate = None
for key in data1_category.keys():
if data11.iloc[i, 0] in data1_category[key]:
cate = key
categories[cate] += data11.iloc[i, j]
for k in range(data12.shape[0]):
data12.iloc[k, j] = categories[data12.iloc[k, 0]]
for k in range(len(category)):
categories[category[k]] = 0

data12.to_excel('result/problem1/单日品类销售统计.xlsx')

# 绘制每个品类的销售量随时间变化图
for i in range(6):
plt.figure(num=(i + 252))
plt.plot(times, data12.iloc[i, 1:], c='#4682B4')
title = data12.iloc[i, 0]
plt.title(title)
plt.savefig('result/problem1/pictures2/' + str(title) + '.png', dpi=400)
plt.close()

# 计算品类销售量列和
col_sum = []
for j in range(1, data12.shape[1]):
mini_sum = 0
for i in range(data12.shape[0]):
mini_sum += data12.iloc[i, j]
col_sum.append(mini_sum)

# 绘制单品堆积柱状图
colors = ['blue', 'green', 'black', 'red', 'yellow', 'purple']
plt.figure(num=252, figsize=(10, 8))
```

```
percentages = []
for i in range(data12.shape[0]):
percentage = data12.iloc[i, 1:] / col_sum
percentages.append(percentage)
for i in range(data12.shape[0]):
if i == 0:
plt.bar(times, percentages[0], color=colors[0], width=0.4, label=data12.iloc[i, 0],
    zorder=5)
else:
plt.bar(times, percentages[i], bottom=percentages[i - 1], color=colors[i], width=0.4,
    label=data12.iloc[i, 0],
zorder=5)
plt.ylim(0, 1.01)
plt.yticks(np.arange(0, 1.2, 0.2), [f'{i}' for i in range(0, 120, 20)])
plt.grid(axis='y', alpha=0.5, ls='--')
plt.legend(frameon=False, bbox_to_anchor=(1.01, 1))
plt.tight_layout()
plt.legend()
plt.savefig('result/problem1/pictures3/品类堆积柱状图.png', dpi=600)

# 计算单品和品类的相关系数矩阵
scaler = MinMaxScaler()
corr1 = np.corrcoef(scaler.fit_transform(data11.iloc[:, 1:]))
corr2 = np.corrcoef(scaler.fit_transform(data12.iloc[:, 1:]))

# 绘制热力图
plt.figure(num=500)
fig1 = sns.heatmap(corr1, annot=False, vmin=-1, vmax=1, square=True, cmap="Blues")
fig1.get_figure().savefig('result/problem1/pictures4/单品相关系数热力图.png',
    bbox_inches='tight', transparent=True,
dpi=600)

plt.figure(num=501)
fig2 = sns.heatmap(corr2, annot=False, vmin=-1, vmax=1, square=True, cmap="Blues")
fig2.get_figure().savefig('result/problem1/pictures4/品类相关系数热力图.png',
    bbox_inches='tight', transparent=True,
dpi=600)

# 进行品类的ADF检验
data13 = pd.DataFrame(columns=['ADF Statistic', 'p-value'])
for i in range(0, data12.shape[0]):
adf_result = adfuller(data12.iloc[i, 1:])
data_13 = {'ADF Statistic': adf_result[0], 'p-value': adf_result[1]}
data13 = pd.concat([data13, pd.DataFrame([data_13])], ignore_index=True)
data13.to_excel('result/problem1/品类ADF检验结果.xlsx')
```

附录 D　问题二第一小问代码

```
import math
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import statsmodels.api as sm

# 提取数据
data1 = pd.DataFrame(pd.read_excel('data/附件1.xlsx'))
data3 = pd.DataFrame(pd.read_excel('data/附件3.xlsx'))
data4 = pd.DataFrame(pd.read_excel('data/附件4.xlsx'))

data21 = pd.DataFrame(pd.read_excel('result/problem1/单品单日销售量统计.xlsx'))
data22 = pd.DataFrame(pd.read_excel('result/problem1/单日品类销售统计.xlsx'))
data23 = pd.DataFrame(pd.read_excel('result/problem2/单品单日定价统计.xlsx'))

data1_category = {}
category1 = []
category2 = []
category3 = []
category4 = []
category5 = []
category6 = []
for i in range(data1.shape[0]):
if data1.iloc[i, 2] == 1011010101:
category1.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010201:
category2.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010402:
category3.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010501:
category4.append(data1.iloc[i, 0])
elif data1.iloc[i, 2] == 1011010504:
category5.append(data1.iloc[i, 0])
else:
category6.append(data1.iloc[i, 0])

data1_category['花叶类'] = category1
data1_category['花菜类'] = category2
data1_category['水生根茎类'] = category3
data1_category['茄类'] = category4
data1_category['辣椒类'] = category5
data1_category['食用菌'] = category6
```

```
# 返回给定编号所属的品类
def cate_select(number):
for key in data1_category.keys():
if number in data1_category[key]:
return key


# 计算单品销量在对应品类中的比重
item_num = data1.iloc[:, 0]
cate_list = list(data1_category.keys())

row_sum = np.sum(data21.iloc[:, 2:], axis=1)
categories_sum = {}
for i in range(len(cate_list)):
categories_sum[cate_list[i]] = []

for i in range(data21.shape[0]):
cate = cate_select(data21.iloc[i, 1])
categories_sum[cate].append(row_sum[i])

cate_sum = {}
for i in range(len(cate_list)):
cate_sum[cate_list[i]] = np.sum(categories_sum[cate_list[i]])

rate = {}
for i in range(data21.shape[0]):
cate = cate_select(data21.iloc[i, 1])
rate[data21.iloc[i, 1]] = np.sum(data21.iloc[i, 2:]) / cate_sum[cate]

# 计算品类加权损耗率
loss_categories = {}
for i in range(len(cate_list)):
loss_categories[cate_list[i]] = 0
for i in range(data4.shape[0]):
num = data4.iloc[i, 0]
cate_flag = cate_select(num)
loss_categories[cate_flag] += data4.iloc[i, 2] * rate[num]

pd.DataFrame(loss_categories,
    index=[0]).T.to_excel('result/problem2/品类加权损耗率统计.xlsx')


# 计算加权和
def weight_sum(data, flag=True, n=0):
return_data = pd.DataFrame(pd.read_excel('data/品类统计.xlsx'))

temp_categories = {}
```

```
for i in range(len(cate_list)):
temp_categories[cate_list[i]] = 0

if flag:
n = 1
for j in range(n + 1, data.shape[1]):
for i in range(data.shape[0]):
num = data.iloc[i, n]
cate_flag = cate_select(num)
temp_categories[cate_flag] += data.iloc[i, j] * rate[num]
for k in range(return_data.shape[0]):
return_data.iloc[k, j - n] = temp_categories[return_data.iloc[k, 0]]
for k in range(len(cate_list)):
temp_categories[cate_list[k]] = 0
return return_data


# 计算品类单日加权定价
data25 = weight_sum(data23)
data25.to_excel('result/problem2/品类单日加权定价统计.xlsx')

# 统计单品每日进价
times = []
time = data3.iloc[0, 0]
count = 1
single_cost = {}
for i in range(len(item_num)):
single_cost[item_num[i]] = 0
data26 = pd.DataFrame(pd.read_excel('data/销售量统计.xlsx'))
for i in range(data3.shape[0]):
temp_time = data3.iloc[i, 0]

if time != temp_time:
time = temp_time
for j in range(data26.shape[0]):
if count < data26.shape[1]:
data26.iloc[j, count] = single_cost[data26.iloc[j, 0]]
for k in range(len(item_num)):
single_cost[item_num[k]] = 0
count += 1

single_cost[data3.iloc[i, 1]] = data3.iloc[i, 2]

if temp_time not in times:
times.append(temp_time)

data26.to_excel('result/problem2/单品每日进价统计.xlsx')
```

```
# 计算品类每日加权定价
data26 = weight_sum(data26, flag=False)
data26.to_excel('result/problem2/品类加权进价统计.xlsx')

# 计算品类加权损失率
cate_loss = {}
for i in range(len(cate_list)):
cate_loss[cate_list[i]] = 0

for i in range(data4.shape[0]):
cate = cate_select(data4.iloc[i, 0])
cate_loss[cate] += data4.iloc[i, 2] * rate[data4.iloc[i, 0]]

data27 = (data25.iloc[:, 1:] - data26.iloc[:, 1:]) / data26.iloc[:, 1:] / 100
data27.columns = range(1, 1086)
data27.to_excel('result/problem2/品类单日利润率统计.xlsx')

profit_rate = {'花叶类': data27.iloc[0, :], '花菜类': data27.iloc[1, :], '水生根茎类':
    data27.iloc[2, :],
   '茄类': data27.iloc[3, :], '辣椒类': data27.iloc[4, :], '食用菌': data27.iloc[5, :]}

# 进行Loess平滑处理
smoothed_sale = {}
for i in range(len(cate_list)):
smoothed_sale[cate_list[i]] = []

for i in range(data22.shape[0]):
loess_smoothed = sm.nonparametric.lowess(data22.iloc[i, 2:], range(1, 1086), frac=0.1)
smoothed_x, smoothed_y = loess_smoothed.T

title = data22.iloc[i, 1]
smoothed_sale[title] = smoothed_y

plt.figure(num=(600 + i))
plt.rc('font', family='SimHei')
plt.rc('axes', unicode_minus=False)

plt.plot(range(1, 1086), data22.iloc[i, 2:], label='原始数据', alpha=0.5)
plt.plot(smoothed_x, smoothed_y, color='red', label='Loess平滑')
plt.title(title)
plt.legend()
plt.savefig('result/problem2/Loess平滑/' + str(title) + '.png', dpi=400)
plt.close()

# 计算弹性销售量
flexible_sale = {}
```

```
for i in range(len(cate_list)):
k = 0
for j in range(data22.shape[0]):
if data22.iloc[j, 1] == cate_list[i]:
k = j
break
flexible_sale[cate_list[i]] = data22.iloc[k, 2:].to_numpy() -
    np.array(smoothed_sale[cate_list[i]])

# 绘制弹性销售量与利润率散点图
for i in range(len(cate_list)):
title = cate_list[i]
Y = flexible_sale[title]
X = profit_rate[title]

Y = list(Y)
X = list(X)

plt.figure(num=(700 + i))
plt.rc('font', family='SimHei')
plt.rc('axes', unicode_minus=False)
plt.scatter(X, Y, marker='o', s=1, color='red', label='数据点')
plt.xlabel('利润率')
plt.ylabel('弹性销售量')
plt.xlim(-1, 1)
plt.legend()
plt.title(title)
plt.savefig('result/problem2/弹性销售量与利润率散点图/' + str(title) + '.png', dpi=400)

data27 = pd.DataFrame(pd.read_excel('result/problem2/品类单日加权定价统计.xlsx'))

# 对时间序列进行差分操作
rate_diff = {}
sale_diff = {}
for i in range(len(cate_list)):
k = 0
for j in range(data22.shape[0]):
if data22.iloc[j, 1] == cate_list[i]:
k = j
break
a = data22.iloc[i, 2:]
b = data27.iloc[i, 2:]
a = a.fillna(a.mean()).to_numpy()
b = b.fillna(a.mean()).to_numpy()
diff_sale = np.diff(a)
diff_rate = np.diff(b)
den1 = data22.iloc[i, :].to_numpy()[2:data22.shape[1] - 1]
```

```
        den2 = data27.iloc[i, :].to_numpy()[2:data27.shape[1] - 1]

        sale_diff[cate_list[i]] = diff_sale / den1
        rate_diff[cate_list[i]] = diff_rate / den2


        # 绘制销售量比率与定价比率散点图
        for i in range(len(cate_list)):
        title = cate_list[i]
        Y = sale_diff[title]
        X = rate_diff[title]

        Y = list(Y)
        X = list(X)

        for j in range(len(X)):
        if math.isnan(X[j]) or math.isinf(X[j]):
        X[j] = np.median(X)
        if math.isnan(Y[j]) or math.isinf(Y[j]):
        Y[j] = np.median(Y)
        df = {'y': np.array(Y), 'x': np.array(X)}
        model1 = sm.formula.ols('y~x', data=df).fit()
        params = model1.params

        x = np.linspace(-1, 1, 1000)
        y1 = params[0] + params[1] * x

        plt.figure(num=(800 + i))
        plt.rc('font', family='SimHei')
        plt.rc('axes', unicode_minus=False)
        plt.scatter(X, Y, marker='o', s=1, color='red', label='数据点')
        plt.plot(x, y1, color='orange', label='OLS拟合')
        plt.xlim(-1, 1)
        plt.ylim(-1, 1)
        plt.legend()
        plt.title(title)
        plt.savefig('result/problem2/销售量与定价散点图/' + str(title) + '.png', dpi=400)
```

附录 E　问题二第二小问代码

```
        import numpy as np
        import math
        import pandas as pd
        from sklearn.ensemble import RandomForestRegressor
        import geatpy as ea
```

```
# 提取数据
cate_list = ['花叶类', '花菜类', '水生根茎类', '茄类', '辣椒类', '食用菌']
data21 = pd.DataFrame(pd.read_excel('result/problem2/品类单日加权定价统计.xlsx'))
data22 = pd.DataFrame(pd.read_excel('result/problem1/单日品类销售统计.xlsx'))
data23 = pd.DataFrame(pd.read_excel('result/problem2/品类加权损耗率统计.xlsx'))
data24 = pd.DataFrame(pd.read_excel('result/problem2/品类加权进价统计.xlsx'))


# 预测单类后七天的数据
def single_predict(data, start):
pred = []
sequence = data.iloc[start, 2:]
X = range(1, 1086)
X_2d = [[x] for x in X]
sequence = sequence.fillna(sequence.mean())
model = RandomForestRegressor(random_state=start, max_depth=5)
model.fit(X_2d, sequence)
pred.append(model.predict([[1086], [1087], [1088], [1089], [1090], [1091], [1092]]))
return [item for sublist in pred for item in sublist]


# 均分列表
def split_into_sublists(input_list, sublist_size):
big_list = []
for i in range(0, len(input_list), sublist_size):
sublist = input_list[i:i + sublist_size]
big_list.append(sublist)
return big_list


# 拟合系数
alpha = [[0.169674, -5.177816], [0.186577, -4.010956], [0.337228, -5.668421],
[0.06551, -0.33593], [0.087314, -1.279035]]


# 定义销售量变化与定价变动函数
def Atan(x):
if x < 1.376:
return math.tan(x) * (1 / math.pi)
else:
return -0.3


# 预测各品类七天进价
C = []
for i in range(data24.shape[0]):
```

```
C.append(single_predict(data24, i))
C = np.array(C).T.tolist()

# 预测花叶类七天定价和销售量
P_leaves = single_predict(data21, 0)
Q_leaves = single_predict(data22, 0)
print(P_leaves)
print(Q_leaves)

# 提取6月30日数据，作为初始值
P0 = list(data21.iloc[1:, data21.shape[1] - 5])
Q0 = list(data22.iloc[1:, 1086])

# 设置常数
loss = list(data23.iloc[:, 1])
for i in range(len(loss)):
loss[i] = loss[i] / 100
discount = 0.6
n = 77


# 遗传算法
# 目标函数构建
@ea.Problem.single
def evalVars(Vars):
P = Vars[:35] # 定价
M = Vars[35:] # 进货量
S = [[] for _ in range(7)] # 库存量
P = split_into_sublists(P, 5)
M = split_into_sublists(M, 6)
Q = [[] for _ in range(7)] # 进价
Pr = [[] for _ in range(7)] # 利润
for i in range(0, 7):
if i == 0:
Q[0].append(Q_leaves[0])
for j in range(0, 5):
q = Q0[j] * (1 + Atan((P[0][j] - P0[j]) / P0[j]))
Q[0].append(q)
for k in range(0, 6):
S_k = M[0][k] - Q[0][k]
if S_k > 0:
S[0].append(S_k)
else:
S[0].append(0)
for m in range(0, 6):
rate = 1 - loss[m] + loss[m] * discount
cost = C[0][m] * M[0][m]
```

```
        if m == 0:
        if Q[0][m] < 0:
        Pr[0].append(Q[0][m] * P_leaves[0] * discount - cost)
        elif S[0][m] > 0 and Q[0][m] > 0:
        Pr[0].append(Q[0][m] * P_leaves[0] * rate - cost)
        elif S[0][m] <= 0:
        Pr[0].append(M[0][m] * P_leaves[0] * rate - cost)
        else:
        Pr[0].append(0)
        else:
        if Q[0][m] < 0:
        Pr[0].append(Q[0][m] * P[0][m - 1] * discount - cost)
        elif S[0][m] > 0 and Q[0][m] > 0:
        Pr[0].append(Q[0][m] * P[0][m - 1] * rate - cost)
        elif S[0][m] <= 0:
        Pr[0].append(M[0][m] * P[0][m - 1] * rate - cost)
        else:
        Pr[0].append(0)
        else:
        Q[i].append(Q_leaves[i])
        for j in range(0, 5):
        q = Q[i - 1][j + 1] * (1 + Atan((P[i][j] - P[i - 1][j]) / P[i - 1][j]))
        if (q - Q[i - 1][j + 1]) / Q[i - 1][j + 1] > 2:
        q = 1 * Q[i - 1][j + 1]
        Q[i].append(q)
        for k in range(0, 6):
        S_k = M[i][k] - Q[i][k]
        if S_k > 0:
        S[i].append(S_k)
        else:
        S[i].append(0)
        for m in range(0, 6):
        rate = 1 - loss[m] + loss[m] * discount
        cost = C[i][m] * M[i][m]
        flag = S[i - 1][m] + M[i][m] - Q[i][m]
        if m == 0:
        if Q[i][m] < 0:
        Pr[i].append(0)
        elif S[i - 1][m] > Q[i][m]:
        Pr[i].append(Q[i][m] * P_leaves[i] * discount - cost)
        elif flag > 0 and S[i - 1][m] < Q[i][m]:
        Pr[i].append(
        S[i - 1][m] * P_leaves[i] * discount + (Q[i][m] - S[i - 1][m]) * P_leaves[i] * rate -
            cost)
        elif flag <= 0:
        Pr[i].append(S[i - 1][m] * P_leaves[i] * discount + M[i][m] * P_leaves[i] * rate - cost)
        else:
```

```
        Pr[i].append(0)
        else:
        if Q[i][m] < 0:
        Pr[i].append(0)
        elif S[i - 1][m] > Q[i][m]:
        Pr[i].append(Q[i][m] * P[i][m - 1] * discount - cost)
        elif flag > 0 and S[i - 1][m] < Q[i][m]:
        Pr[i].append(
        S[i - 1][m] * P[i][m - 1] * discount + (Q[i][m] - S[i - 1][m]) * P[i][m - 1] * rate -
            cost)
        elif flag <= 0:
        Pr[i].append(S[i - 1][m] * P[i][m - 1] * discount + M[i][m] * P[i][m - 1] * rate - cost)
        else:
        Pr[i].append(0)
        f = np.sum(np.array(Pr))
        return f


        # 约束条件构建
        D = []
        for sublist in C:
        for item in sublist:
        D.append(item * 0.5)

        D = D[7:]

        types = [0] * n
        lb = D + [0] * 42
        ub = [20] * 35 + [120] * 42

        # 问题参数设置
        problem = ea.Problem(name='solver',
        M=1,
        maxormins=[-1],
        Dim=n,
        varTypes=types,
        lb=lb,
        ub=ub,
        evalVars=evalVars)

        # 调用增强精英保留策略的遗传算法
        algorithm = ea.soea_SEGA_templet(problem,
        ea.Population(Encoding='BG', NIND=300),
        MAXGEN=200,
        logTras=1,
        trappedValue=1e-5,
        maxTrappedCount=10)
```

```
# 求解
res = ea.optimize(algorithm, seed=1, verbose=True, drawing=1, outputMsg=True,
    drawLog=False, saveFlag=True,
dirName='result/problem2/solver')
```

附录 F　问题三代码

```
import math
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import geatpy as ea

# 导入数据
data1 = pd.DataFrame(pd.read_excel('data/附件1.xlsx'))
data4 = pd.DataFrame(pd.read_excel('data/附件4.xlsx'))
data31 = pd.DataFrame(pd.read_excel('result/problem1/单日品类销售统计.xlsx'))
data32 = pd.DataFrame(pd.read_excel('result/problem2/品类单日利润率统计.xlsx'))
data33 = pd.DataFrame(pd.read_excel('data/部分进价统计.xlsx'))
data34 = pd.DataFrame(pd.read_excel('data/部分销售量统计.xlsx'))
data35 = pd.DataFrame(pd.read_excel('result/problem2/单品单日定价统计.xlsx'))

# 设置常量
cate_list = ['花叶类', '花菜类', '水生根茎类', '茄类', '辣椒类', '食用菌']
numbers = list(data33.iloc[:, 1])
L = []
n = 33 * 2
discount = 0.6


# 预测函数，使用随机森林算法
def predict(data, start, day):
pred = []
R2 = {}
for i in range(data.shape[0]):
cate = data.iloc[i, start - 1]
sequence = data.iloc[i, start:]
X = range(1, 1086)
X_2d = [[x] for x in X]
sequence = sequence.fillna(sequence.mean())
sequence.replace([np.inf, -np.inf], 0, inplace=True)
model = RandomForestRegressor(random_state=0, max_depth=5)
model.fit(X_2d, sequence)
pred.append(model.predict([[day]]))
```

```
        R2[cate] = model.score(X_2d, sequence)
        return [item for sublist in pred for item in sublist], R2


        # 预测下面7天的销售量
        sale_pred, sale_R2 = predict(data31, 2, 1086)

        # 计算30日内的各品类销售量均值
        sale_mean = []
        shape = data34.shape[1]
        for i in range(data34.shape[0]):
        mean = np.mean(data34.iloc[i, shape - 30: shape])
        sale_mean.append(mean)

        # 计算各单品7日内定价均值
        P_pred = []
        shape = data35.shape[1]
        for i in range(len(numbers)):
        for j in range(data35.shape[0]):
        if data35.iloc[j, 1] == numbers[i]:
        mean = np.mean(data35.iloc[j, shape - 7: shape])
        P_pred.append(mean)

        # 预测下面7天的各单品进价
        pred = []
        for i in range(data33.shape[0]):
        sequence0 = data33.iloc[i, 1079:]
        X = range(1078, 1086)
        X_2d = [[x] for x in X]
        sequence0 = sequence0.fillna(sequence0.mean())
        sequence0.replace([np.inf, -np.inf], 0, inplace=True)
        model = RandomForestRegressor(random_state=0, max_depth=5)
        model.fit(X_2d, sequence0)
        pred.append(model.predict([[1086]]))
        Q = [item for sublist in pred for item in sublist]

        # 统计各单品损耗率
        for i in range(n // 2):
        for j in range(data4.shape[0]):
        if data4.iloc[j, 0] == numbers[i]:
        L.append(data4.iloc[j, 2] / 100)


        # 定义销售量变化与定价变动函数
        def Atan(x):
        if x < 1.376:
        return math.tan(x) * (1 / math.pi)
```

```
else:
return -0.3


# 遗传算法
# 目标函数构建即约束不等式的构建
@ea.Problem.single
def evalVars(Vars):
P = Vars[: n // 2]
m = Vars[n // 2:]
F = 0
CV = []
cv = 0
for i in range(n // 2 + 1):
if i < n // 2:
N = (1 + Atan((P[i] - P_pred[i]) / P[i])) * sale_mean[i]
if m[i] >= N:
F += N * P[i] - Q[i] * m[i]
else:
F += m[i] * P[i] * (1 - L[i] + L[i] * discount) - Q[i] * m[i]

if i <= 14:
cv += -m[i]
elif 14 < i <= 16:
if i == 15:
CV.append(cv + sale_pred[0])
cv = 0
cv += -m[i]
elif 16 < i <= 19:
if i == 17:
CV.append(cv + sale_pred[1])
cv = 0
cv += -m[i]
elif 19 < i <= 23:
if i == 20:
CV.append(cv + sale_pred[2])
cv = 0
cv += -m[i]
elif 23 < i <= 28:
if i == 24:
CV.append(cv + sale_pred[3])
cv = 0
cv += -m[i]
elif 28 < i <= 32:
if i == 29:
CV.append(cv + sale_pred[4])
cv = 0
```

```
        cv += -m[i]
    else:
        CV.append(cv + sale_pred[5])
    return F, np.array(CV)


# 约束范围构建
types = [0] * (n // 2) + [0] * (n // 2)
lb = [0.01] * (n // 2) + [2.5] * (n // 2)
ub = [15] * (n // 2) + [150] * (n // 2)

# 问题参数设置
problem = ea.Problem(name='solver',
M=1,
maxormins=[-1],
Dim=n,
varTypes=types,
lb=lb,
ub=ub,
evalVars=evalVars)

# 调用增强精英保留策略的遗传算法
algorithm = ea.soea_SEGA_templet(problem,
ea.Population(Encoding='BG', NIND=300),
MAXGEN=100,
logTras=1,
trappedValue=1e-6,
maxTrappedCount=10)

# 求解
res = ea.optimize(algorithm, seed=1, verbose=True, drawing=1, outputMsg=True,
    drawLog=False, saveFlag=True,
dirName='result/problem3/solver')
```