生鲜商超定价与日补货量决策的优化策略

摘要

蔬菜类商品具有产量不稳定、易腐烂变质、市场依赖性强等特点，对于生鲜商超而言，能够依据已有数据进行市场需求分析，对商超的补货与定价决策则能够起到很大帮助。本文主要研究蔬菜类商品不同品类或单品之间的销量、定价关系以及商超应对市场时所应采用的日补货策略和定价策略。通过基于季节性指数的时间序列模型指数分析法、集成拟合的一元 (多元) 线性模型、定单点法等方法，预测了各蔬菜品类或单品销量和批发价格的预测值，给出了在一定限制条件下最优的日补货策略和定价策略。

问题一需要分析蔬菜品类和单品在销售量上的分布规律，本文首先对附件一和附件二中的数据进行了预处理，得到了单品名称、品类名称、单品编码、销售量的汇总数据，然后绘制蔬菜品类销量的箱线图、各蔬菜品类销量的热力图，进而对不同品类之间销量相关性关系进行分析。

问题二需要对各蔬菜品类及单品的销售总量和成本加成定价间的关系进行分析，并给出最佳日补货和定价策略。首先建立了季节性指数得时间序列模型，对各品类蔬菜的销售量和批发价格进行预测，利用 excel 中的指数平滑法功能计算得到了 7 月 1 日到 7 月 7 日各蔬菜品类的批发价和销量的预测结果。建立一元线性规划模型，对各品类蔬菜的成本加成定价和销售总量的关系用最小二乘法进行拟合，得到了参数拟合结果。进一步，使用定单点法计算安全库存以及日补货量的最大约束，并统计历史各蔬菜品类的最小销售量作为其最小约束。在此约束条件的基础上，构建以最大化商超收益为目标，以各蔬菜品类的成本利润率为决策变量的多变元线性规划模型，引入季节性指数与加成率动态调整商朝的补货与定价策略。利用遗传算法对模型进行求解，得到了各品类在 2023 年 7 月 1 日到 7 月 3 日的日补货量和定价策略预测结果。

针对问题三：需要在一系列约束条件下，在问题二基础上进一步给出 2023 年 7 月 1 日单品补货量和定价策略，使得收益最大。本文在问题二解决模型的基础上，以问题二中时间序列模型所预测得到的各品类总销售量和批发价格为总数，以各单品历史销售量占所属品类的比例为权重，通过计算加权平均值得到各单品的预测销量和批发价格。然后建立有约束的双变元线性规划回归模型，利用 pyThon 线性规划库 scipy。optimize。linprog 来计算求解得到了满足约束条件的最佳日补货量与定价策略。

针对问题四：为了更好的解决补货和定价问题，本文引入了季节性因素、库存数据、促销活动数据等作为建议补充的数据，并且具体对促销活动数据建立了 0-1 规划模型，并给出了蒙特卡洛法求解的思路，从而能够使商家从更多方面因素进行日补货策略和定价策略的制定。

关键词：生鲜商超补货与定价决策　时间序列模型　指数平滑分析　遗传算法　定单点法

一、 问题重述

1.1 问题背景

蔬菜类商品往往具有保质期较短、易于变质腐烂的特点，其销售价值往往虽销售时间的增加而降低，且大多数当日滞销的蔬菜类商品也不能在隔日进行再售。蔬菜的定价一般采用“成本加成定价”的方法，对于运损以及因变质腐烂而导致品相变差的商品，商超通常需要进行打折处理。因此，蔬菜类商品的商品损耗一般远远高于其它商品，生鲜商超也往往会因商品损耗而付出更多的成本代价。

由于蔬菜进货交易时间大多在凌晨，商家在前一日进行决策时往往不能够确定销售当天具体单品的进货成本与销售价格。同时，季节性生产的蔬菜类商品对市场有较强依赖性。在不同的生产季节，蔬菜产品可能会在销售价格与销量上有着较大的差异。

受制于有限的商超销售空间，生鲜商超需要在不同时间范围内能够选择合理的蔬菜种类销售组合，并依据各销售品类的历史销售以及需求情况进行补货决策和定价决策。

1.2 问题要求

问题一要求依据附件中的相关数据，通过对蔬菜商品各品类及单品销售量分布规律的分析，进一步探求蔬菜类商品不同品类或不同单品之间所存在的关联关系。

问题二属于优化类问题，要求以蔬菜的品类为单位来做补货计划：对各蔬菜品类的销售总量与成本加成定价的关系进行分析，从而给出在 2023 年 7 月 1 日到 7 月 7 日各蔬菜品类的日补货总量和定价策略，使商朝能够获得最大的收益。

问题三要求在问题二的基础上进一步对单品的补货计划进行细化。受限于商超蔬菜类商品的有限销售空间，题目将可进行销售的单品总数限制在了 27-33 个，且每种单品订购量均需要满足最小陈列量 2.5 千克的要求。在此基础之上，题目要求以 2023 年 6 月 24 日到 6 月 30 日的可售品种为根据，确定 7 月 1 日的单品补货量和定价策略，从而使得商超在能够满足市场对各个蔬菜品类商品需求的前提下，尽可能得获取最大收益。

问题四属于开放性问题，在题目已给出的数据以外，题目要求对商超的数据采集策略提供建议，并阐述所建议的数据对制定补货和定价决策所提供的帮助，从而更好地指定补货及定价计划。

二、 问题分析

2.1 问题一的分析

首先对附件一和附件二的数据进行预处理，统计出该时间范围内各蔬菜单品及品类销量和销售量的汇总数据，并整理出各蔬菜品类随时间变化的销量变化情况。对于蔬菜各品类及单品间的分布规律和相关关系，可以通过统计销量的平均值、最值、标准差等统计数据来观测销量的分布统计情况；通过计算蔬菜品类或单品销量之间的相关系数来观测不同品类或单品之间的线性相关性。

2.2 问题二的分析

对于销售总量与成本加成定价间的关系，可以考虑建立集成拟合模型来设出多项式函数来表征其间关系。而对于日补货总量和定价策略，为了使生鲜商超能够达到最大，需要表达出生鲜商超的收益与日补货量和定价的关系。收益的表达可能需要使用预测得到的各品类批发价格的加权平均值和蔬菜品类中各单品销售价的加权平均值 (权重为各单品历史销量占比)，可以考虑利用指数平滑法建立时间序列模型预测得到。对于商超使用的成本加成定价方式，可以考虑使用加价率作为调整定价的方式，对于日补货量的计算则可以考虑使用定单点法。

2.3 问题三的分析

问题三要求在问题二的基础上进一步制定单品的补货计划，对可售单品数、各单品订购量的最小陈列量有一定的要求。考虑在主题上沿用问题二中的模型，将问题三给出的限制作为约束条件，在沿用问题二中多元线性规划模型基础上，建立有约束的双变元线性回归，求解组合优化问题，预测单品的批发价格和销售量，完成单品销售量和单品批发价格的预测，进而求解。

2.4 问题四的分析

三、 模型假设

1. 假设附件中所给出的数据真实且且合理。
2. 假设季节性因素在每年的 4 月至 10 月期间具有明显的影响。
3. 假设商品损耗率不会出现太大的波动，基本与附件中给出的损耗率数据相接近。
4. 假设历史销售数据能够代表未来的销售趋势。
5. 假设附件中所给出的损耗商品均无法出售。
6. 假设未通过数据明确显示出相关性的各个单品或品类之间的销售是独立的。

四、 符号说明

符号	含义
$\phi(t)$	季节性指数
Ω	定单点法中的安全库存
$\theta(t)$	指数平滑法中的平滑值
C_i	某一品类蔬菜的日补货量
$P_i(t)$	以 t 时刻时第 i 个蔬菜品类中各单品成本加成定价为元素的向量
$S_i(t)$	以 t 时刻时第 i 个蔬菜品类中各单品批发价格为元素的向量
K_i	以第 i 个蔬菜品类中各蔬菜单品所占整个品类的销售总量比例为元素的向量
$D_i(t)$	t 时刻第 i 个蔬菜品类的总销售量
T	时间序列，表示历史销售数据的时间点
$\bar{S}_i(t)$	t 时刻第 i 个蔬菜品类的平均加权批发价格
R_i	以第 i 个蔬菜品类中各单品利润为元素的向量
$\bar{P}_i(t)$	t 时刻第 i 个蔬菜品类的平均加权成本加成定价
L_i	以第 i 个蔬菜品类中各单品的损耗率为元素的向量

五、 模型建立与求解

5.1 问题一模型建立与求解

问题一要求通过对附件的数据进行分析，得到蔬菜各品类及单品销售量的分布规律及相互关系。附件一中包含了蔬菜各品类的分类名称、分类编码、单品名称和单品编码，而附件二中包含了某生鲜商超从 2020 年 7 月 1 日至 2023 年 6 月 30 日的销售流水明细数据，其中包括了每一单所对应的单品编码、销量等数据。

对于问题一，首先对附件一和附件二中的数据进行了整合，将附件二销售流水明细

中的每一单销售量依据单品编码与附件一所提供的各蔬菜商品品类和单品名称进行对照，随后按品类进行划分，分别统计各品类销量的总和。图 1体现了数据的初步处理结果。依据处理统计后的数据，利用 python 程序绘制各蔬菜品类销量的箱线图进而观察销量的大致分布情况，绘制大数据量柱图观测各种单品总销量的分布情况。对于相关关系，计算各品类间的相关系数，进而绘制各品类销量间的热力图，观测不同品类间的线性相关关系。

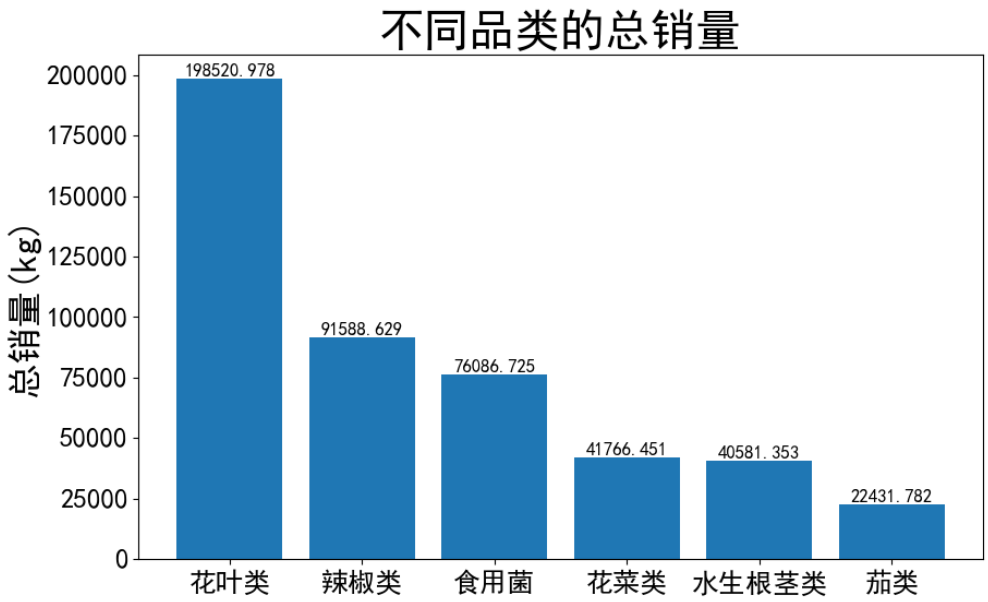

图 **1**　各蔬菜品类销量的柱状图

5.1.1 绘制分析各蔬菜品类销量的箱线图

为了能够较为清晰地通过图表的形式表示出蔬菜类商品各品类的销量分布情况，选择绘制箱线图来进行观测。附件二的销量数据中大多在 0-5kg 的范围内，销量分布于 0-5kg 范围以外的数据基本来自于花叶类，共计有 466 单。为了能够突出展示各品类商品的销量分布情况，考虑到 466 单数据相对于总数据量 878503 单占比 0.05%, 在绘制盒线图的的过程中将分布在 0-5kg 范围以外的数据进行忽略，从而能够得到一个更能清晰表示销量分布情况的盒线图。

图 2是绘制得到的各蔬菜品类销量的箱线图。从盒线图中可以得到，各品类销量的上四分位数都在 1kg 以下，说明大多数蔬菜类商品的销量均集中在 $[0,1)$ 的范围之内。对于分布在上四分位数以上的数据，食用菌、花菜类和茄类主要分布在 3kg 范围以内，花叶类和水生根茎类分布于上四分位数以上的数据的分布相对更为均匀，再结合上文提到的“销量分布于 0-5kg 范围以外的数据基本来自于花叶类”，可以得到花叶类蔬菜商品往往有被大量购买的倾向。而结合日常生活经验，花叶类蔬菜往往作为日常视频中蔬菜

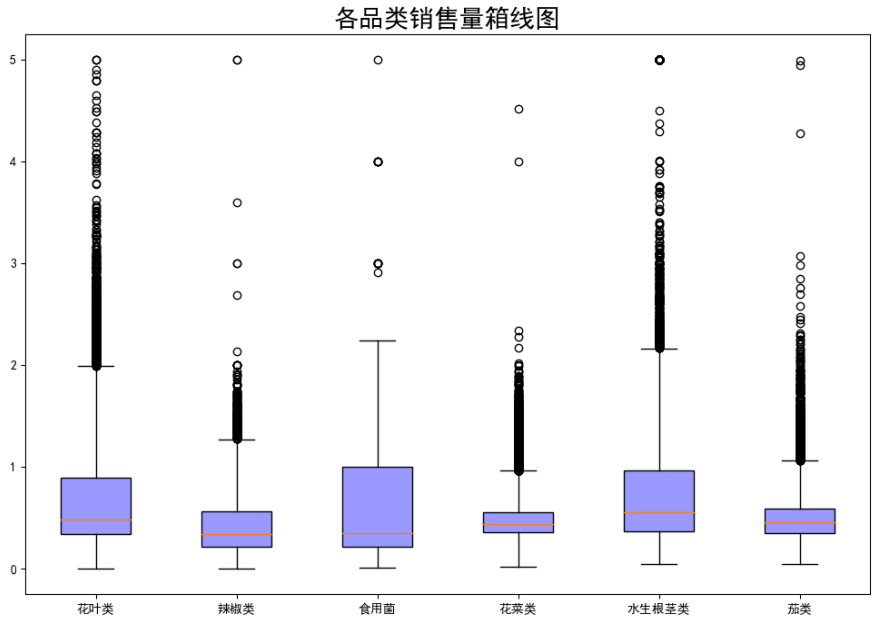

图 **2**　各蔬菜品类销量的箱线图

摄入的主要来源，而水生根茎类蔬菜往往能够在保持完整的情况下储存较长的时间，并在日常食用时也往往需要大量食用，一定程度上解释了两种商品在四分位数以上部分出现较为均匀的分布情况的原因。

5.1.2 绘制分析各蔬菜品类销量的热力图

在完成数据处理的基础之上，对汇总后的数据进行数据透视，以销售日期为索引，以蔬菜品类名称作为列的名称，对销量进行求和从而获取透视表。然后通过 python 中的 corr() 函数获得不同品种间的 Pearson 相关系数，再利用 python 的 seaborn 库，从库中调用 heatmap 函数绘制得到不同蔬菜品类销量的热力图。

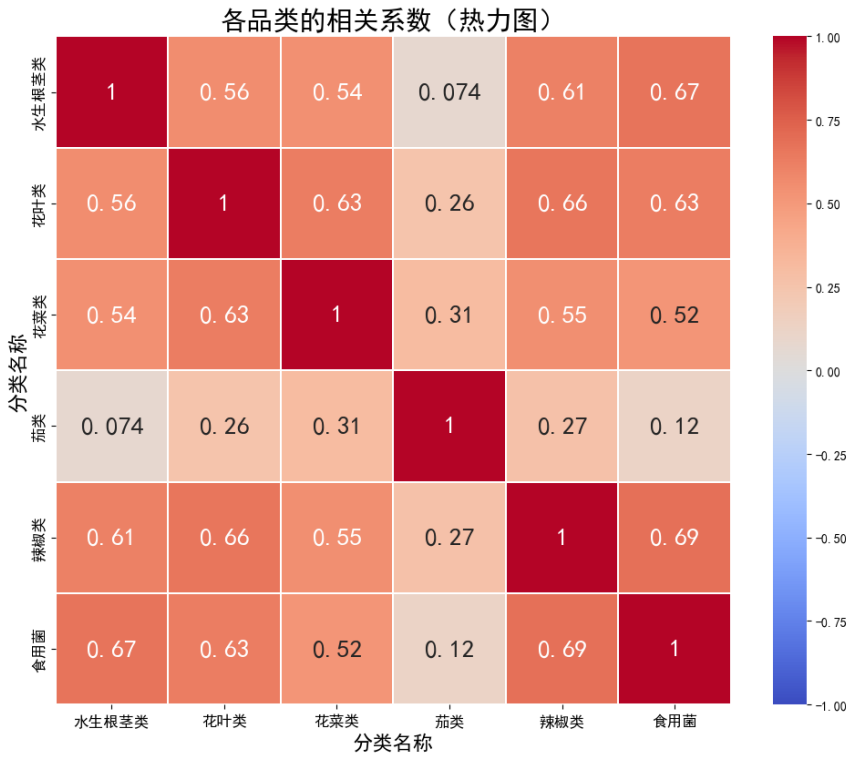

图 **3**　各蔬菜品类相关关系的热力图

图 3 是利用 python 程序所得到的各蔬菜品类销量的热力图。各蔬菜品类销售量在 Pearson 相关系数上整体呈现一个正相关的态势。其中，相较于其它品类，茄类与其它各品类销量的 Pearson 相关系数均没有超过 0.32，说明茄类蔬菜销量与其它蔬菜相关性均不大，剩余各品类之间的销量 Pearson 相关系数大多都维持在 0.5-0.7 之间，不同品类之间的相关性并没有在以 Pearson 相关系数为基础的热力图中体现较大差异，且品类销售量之间也并未体现出非常强的相关性。

5.1.3 绘制分析各蔬菜品类销售量变化趋势的折线图

为了能够清晰描述各蔬菜品类销售量随时间的变化趋势，选用折线图以季度为单位，各品类销售量随时间的变化进行描述。

图 4是绘制得到的各蔬菜品类销售量随月份变化的趋势折线图，从图中可以看出蔬菜类商品的销售随时间变化具有一定的波动性。其中以一月份数据为例，2021 年一月份和 2023 年一月份的花叶类销售量在折线图中均处在一个较高的位置，而 2022 年 1 月份花菜类的销售量则达到了整个折线图的最底端，说明不同年份的同一月份的花叶类蔬菜上品销售量也可能呈现不一样的趋势。而相对于花叶类蔬菜商品在 2022 年上半年呈现一个低谷状态，其它品类的蔬菜商品销售量则在 2020 年九月至 2023 年 5 月整体呈现

一个相对稳定的状态，说明不同类别的蔬菜商品之间的相互影响有限。

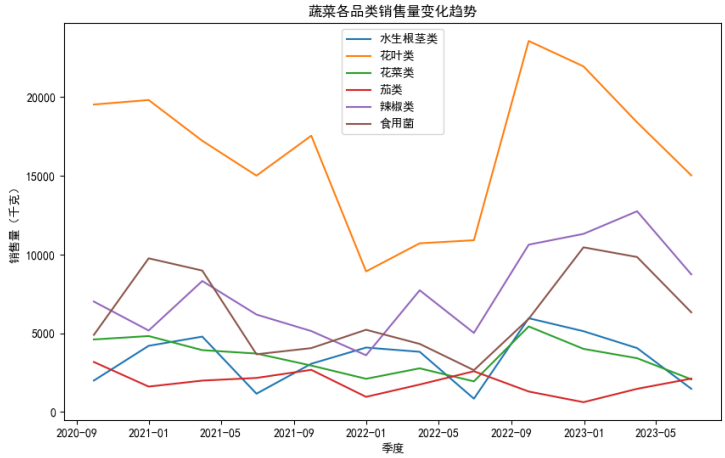

图 **4**　各蔬菜品类销售量随季度变化的折线图

5.2 问题二模型建立与求解

5.2.1 引入季节性指数来体现不同月份各品类销售总量的变化

由问题一的求解过程可得，蔬菜类商品的销量在不同季节时期往往会呈现出一定的波动性。因此，为了能够更为准确的描述蔬菜类商品销量随季节变化而波动的特性，考虑引入季节性参数。首先，引入季节性指数 $\Phi(t)$, 其中，t 表示某一时刻。这个指数用以描述不同月份的季节性质。每年有 12 个月，那么 $\Phi(t)$ 可以从 1 到 12 之间取值为不同的整数，从而用来描述不同月份的不同季节性。

根据历史数据，对每个月份的季节性指数 $\Phi(t)$ 进行估计。在估计过程中，首先统计每个月的平均销售总量，然后将平均销售总量最高的月份所对应的季节性指数设置为 12，其它月份的季节性指数则依据平均销售总量的相对比例关系进行对应的设置。

	季节性指数					
	水生根茎类	花叶类	花菜类	茄类	辣椒类	食用菌
1月	0.150074814	0.089937437	0.098244258	0.080185159	0.111361226	0.129121893
2月	0.100064061	0.073357139	0.082291742	0.082483193	0.10572961	0.099763132
3月	0.061968658	0.069999867	0.061713958	0.06939948	0.09723884	0.075282804
4月	0.034336115	0.068237861	0.060640632	0.081522636	0.080690388	0.06156413
5月	0.021851933	0.071040487	0.06476284	0.110446241	0.072247189	0.053290768
6月	0.029527108	0.066962984	0.059737156	0.114471289	0.064857713	0.051395996
7月	0.067526999	0.087508676	0.106997264	0.129321915	0.067004268	0.059316891
8月	0.1110633	0.123442526	0.117951966	0.121978227	0.107944994	0.069614667
9月	0.092892196	0.094318823	0.085915631	0.067652672	0.073740639	0.066473159
10月	0.121597523	0.096776604	0.093503803	0.062439578	0.086621441	0.112224228
11月	0.088389143	0.075215739	0.092933369	0.041740598	0.067256504	0.107204115
12月	0.120708149	0.083201857	0.07530738	0.038359012	0.06530719	0.114748217

5.2.2 准备工作——运用指数平滑法建立各品类销量的时间序列模型

1. 建立各品类总销量的时间序列模型

为了描述蔬菜销售总量与时间变化之间的关系，选择使用指数平滑法来建立时间序列模型。假设市场条件（例如消费者购买力、偏好等）保持相对稳定，这使得时间序列分析变得可行。我们使用平滑值 $\Theta(t)$ 来表示某一品类蔬菜销售总量 $D_i(t)$ 的趋势和季节性影响的组合。平滑值的计算方法如下：

$$\Theta(t) = \alpha \cdot \frac{D_i(t)}{\Theta(t-m)} + (1-\alpha) \cdot [\Theta(t-1) + \Gamma(t-1)] \tag{1}$$

其中，t 是某一具体的时刻，α 是平滑系数，在 0 到 1 之间选择，用于平滑实际销售总量和季节性指数。$D_i(t)$ 是第 i 个品类蔬菜商品实际销售总量，$\Theta(t-m)$ 表示 t 时刻所对应的去年同一月份对应的平滑值，$\Theta(t-1)$ 为上一时刻的平滑值 $\Gamma(t-1)$ 是上一时刻的趋势值。

考虑到数据样本量较大，销售量数据的波动情况较为复杂，为了能够将无法由趋势和季节性解释的随机波动加入到模型中进行描述，考虑在平滑值计算公式加上一项 $\xi(t)$ 作为随机误差项。那么平滑值计算公式可以被写成：

$$\Theta(t) = \alpha \cdot \frac{D_i(t)}{\Theta(t-m)} + (1-\alpha) \cdot [\Theta(t-1) + \Gamma(t-1)] + \xi(t) \tag{2}$$

在平滑值的计算方法中涉及到了趋势值 $\Gamma(t)$，趋势值用来表示销售总量的长期趋势，可以用下面的公式进行计算：

$$\Gamma(t) = \beta \cdot [\Theta(t) - \Theta(t-1)] + (1-\beta) \cdot \Gamma(t-1) \tag{3}$$

其中，t 是某一具体的时刻，β 是趋势平滑指数，在 0 到 1 之间选择，用于平滑趋势值。$\Theta(t-1)$ 为上一时刻的平滑值，$\Theta(t)$ 为该时刻的平滑值，$\Gamma(t-1)$ 是上一时刻的平滑值。

在运用指数平滑法的同时可以考虑将季节性指数加入到时间序列模型当中，季节性指数 $\Phi(t)$ 可以用来表示每个月份销售量相对于年度平均销售量的季节性影响，在结合了指数平滑法的前提下，在本模型中，季节性指数可以用以下公式进行计算：

$$\Phi(t) = \frac{D_i(t)}{\Theta(t)} \tag{4}$$

其中，$\Phi(t)$ 是季节性指数，$D_i(t)$ 是第 i 个蔬菜品类在时刻 t 的销售总量，$\Theta(t)$ 为平滑值。

最终依据上述的三个式子可以得到下一时刻某蔬菜品类销量的预测值：

$$D_i(t+1) = \Phi(t+1) \cdot \Theta(t+1) \tag{5}$$

其中，$\Phi(t+1)$ 可以根据历史季节性指数进行估算，或者在实际应用过程中根据实际情况进行调整。

预测销量（千克）	水生根茎类	花叶类	花菜类	茄类	辣椒类	食用菌
2023/7/1	19.8086	246.8522	46.7242	27.0812	55.2246	41.3894
2023/7/2	7.8417	169.1456	32.4624	19.1482	31.3057	49.0403
2023/7/3	5.2483	190.0651	51.1733	25.8096	51.0603	46.3713
2023/7/4	8.7073	188.9506	48.5899	39.0563	57.1481	43.0183
2023/7/5	6.0927	131.0741	32.0820	22.0529	49.3960	37.9541
2023/7/6	4.4337	132.7828	40.5284	18.4298	49.6616	47.8844
2023/7/7	3.3579	158.1254	33.0545	10.9164	50.8275	35.3577

2. 求解时间序列模型预测各品类总销量

(a) 根据历史数据，绘制各品类销售总量随时间变化的曲线图，取数据起始端的的蔬菜品类销售量为初始的平滑值 $\Theta(1)$, 取曲线图初始段的斜率极限值为初始的趋势值 $\Gamma(1)$。

(b) 对于每个时刻 t，根据建模过程中的公式依次计算平滑值 $\Theta(t)$、趋势值 $\Gamma(t)$ 和季节性指数 $\Phi(t)$。

(c) 利用计算得到的平滑值和趋势值，结合一致的历史季节性指数，可以估计未来的销售总量。

3. 预测结果由时间序列模型预测得到的结果如下表所示

预测销量（千克）	水生根茎类	花叶类	花菜类	茄类	辣椒类	食用菌
2023/7/1	19.8086	246.8522	46.7242	27.0812	55.2246	41.3894
2023/7/2	7.8417	169.1456	32.4624	19.1482	31.3057	49.0403
2023/7/3	5.2483	190.0651	51.1733	25.8096	51.0603	46.3713
2023/7/4	8.7073	188.9506	48.5899	39.0563	57.1481	43.0183
2023/7/5	6.0927	131.0741	32.0820	22.0529	49.3960	37.9541
2023/7/6	4.4337	132.7828	40.5284	18.4298	49.6616	47.8844
2023/7/7	3.3579	158.1254	33.0545	10.9164	50.8275	35.3577

5.2.3 准备工作——利用指数平滑法建立批发价格的时间序列模型

1. 建立批发价格的时间序列模型

根据现实生活的经验，由于蔬菜类商品具有季节性生产、产量不稳定等特点，同时蔬菜类商品的销售具有市场依赖性的特点，因此，蔬菜类商品的平均加权批发价格会随时间呈现波动性。为了描述蔬菜类商品批发价格随时间变化的关系，采用指数平滑法建立时间序列模型对各蔬菜品类销售价格进行预测。假设市场条件（例如消费者购买力、偏好等）保持相对稳定，这使得时间序列分析变得可行。

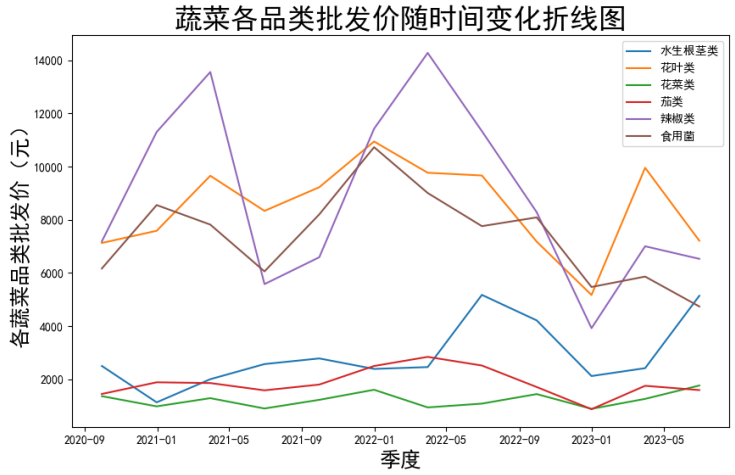

图 **5**　蔬菜各品类批发价随时间折线图

在批发价格的时间序列模型总，平滑值是第 i 各蔬菜类商品平均加权批发价格与季节性影响的总和，平滑值可以使用下面的公式进行计算：

$$\Theta(t)=\alpha\cdot\frac{\bar{S}_i(t)}{\Theta(t-m)}+(1-\alpha)\cdot[\Theta(t-1)+\Gamma(t-1)]+\xi(t) \tag{6}$$

其中，$\bar{S}_i(t)$ 是 t 时刻第 i 种蔬菜品类的加权平均批发价格 (权重为各单品历史销售量所占品类销售量的比例)，t 是某一具体的时刻，α 是平滑系数，在 0 到 1 之间选择，用于平滑实际蔬菜品类批发成本和季节性指数。$\Theta(t-m)$ 表示 t 时刻所对应的去年同一月份对应的平滑值，$\Theta(t-1)$ 为上一时刻的平滑值 $\Gamma(t-1)$ 是上一时刻的趋势值。

平滑值计算公式中的趋势值可以由以下公式进行计算：

$$\Gamma(t)=\beta\cdot[\Theta(t)-\Theta(t-1)]+(1-\beta)\cdot\Gamma(t-1) \tag{7}$$

其中，β 是趋势平滑指数，在 0 到 1 之间选择，用于平滑趋势值。

在批发价格的时间序列模型中，季节性指数可以用下面的公式进行计算：

$$\Phi(t)=\frac{\bar{S}_i(t)}{\Theta(t)} \tag{8}$$

最终依据上述的三个式子可以得到下一时刻某蔬菜品类批发价格的预测值：

$$\bar{S}_i(t+1)=\Phi(t+1)\cdot\Theta(t+1) \tag{9}$$

其中，$\Phi(t+1)$ 可以根据历史季节性指数进行估算，或者在实际应用过程中根据实际情况进行调整。

假设蔬菜类商品的批发价格会随着季节性蔬菜品类数量的增多而降低，那么对于蔬菜类商品而言，季节性指数越高，其批发价格就越低。而季节性指数由问题一中每

一月份蔬菜品类以及单品种类数决定，蔬菜品类及单品数越多，季节指数越高。依据题目所给出的信息，可以在一年 1 到 12 月份中，4 到 10 月的季节性指数取为相对高一些的值。

2. 求解时间序列模型预测各品类蔬菜商品的批发价格

(a) 首先处理附件三中批发价格的相关数据，先对照附件一将单品名称依据单品编码与具体日期的批发价格相对应，然后绘制蔬菜各品类及单品批发价格随时间的变化曲线图。取数据起始端的的蔬菜品类批发价格为初始的平滑值 $\Theta(1)$，取曲线图初始段的斜率极限值为初始的趋势值 $\Gamma(1)$。

(b) 对于每个时刻 t，根据建模过程中的公式依次计算平滑值 $\Theta(t)$、趋势值 $\Gamma(t)$ 和季节性指数 $\Phi(t)$。

(c) 利用计算得到的平滑值和趋势值，结合一致的历史季节性指数，可以估计未来的批发价格。

预测批发价（元）	水生根茎类	花叶类	花菜类	茄类	辣椒类	食用菌
2023/7/1	12.9378	3.2025	5.4357	4.5490	5.4465	7.5115
2023/7/2	12.9130	3.0732	5.7947	4.4271	7.4974	6.7121
2023/7/3	12.7020	3.1328	5.7725	4.0371	8.9073	8.5300
2023/7/4	10.1526	3.1683	6.0378	4.1568	8.6993	7.9064
2023/7/5	9.7794	3.3406	6.2236	3.7782	8.3622	9.3493
2023/7/6	10.5827	3.0766	5.9431	3.6207	8.1568	7.9064
2023/7/7	9.8911	3.0448	5.8814	3.5489	8.9427	8.7693

5.2.4 模型建立——建立集成拟合的一元线性规划模型

为进一步分析销售总量与成本加成定价之间的关系，我们建立了集成拟合模型对每一蔬菜单品进行多项式函数的拟合，从而表征销量与成本加成定价的关系。

首先依据先前对附件所进行的数据整理工作，以各蔬菜品类中各单品成本加成定价按单品销售量进行加权平均的数值作为蔬菜品类的平均成本加成定价。以蔬菜品类的平均成本加成定价和各蔬菜品类总销量为横纵坐标，以时间为序列，绘制散点图。建立一元线性回归模型，利用散点图中的样本进行拟合，多项式函数的一般形式为：

$$D_i = a_0 + a_1 \cdot \bar{P}_i$$

其中，D_i 是第 i 种蔬菜品类的总销量，$\bar{P}_i$ 是第 i 种蔬菜品类的依据销量进行加权平均后的成本加成定价，a_0, a_1 是多项式系数。

取 2020 到 2023 年的历史销售数据进行拟合，使用最小二乘法对多项式系数 a_0, a_1 进行估算, 最小化实际销量和多项式模型得预测销量之间的残差平方和，从而能够最大程度地拟合销量与成本加成定价之间的关系。定义损失函数为实际销量与多项式模型得

预测销量之间的残差平方和，其具体定义如下：

$$L\left(a_0, a_1, a_2, \ldots, a_n\right)=\sum_{i=1}^{n}\left[D_i-\left(a_0+a_1 \cdot P_i+a_2 \cdot X_{i2}+\cdots+a_n \cdot X_{in}\right)\right]^2 \quad (10)$$

求 L 关于 $a_0, a_1, a_2, \ldots, a_n$ 的偏导数，并令这 n 个偏导数为 0 获得方程组：

$$\begin{cases} \frac{\partial L}{\partial a_0}=0 \\ \frac{\partial L}{\partial a_1}=0 \\ \vdots \\ \frac{\partial L}{\partial a_n}=0 \end{cases}$$

从而能够得到 $a_0, a_1, a_2, \ldots, a_n$ 的估计值，进而得到了多项式模型的系数。

		各蔬菜品类					
		茄类	食用菌	水生根茎类	辣椒类	花菜类	花叶类
多项式模型的系数	a_0	-3834.571	-638.112	-3074.382	-1686.453	-15980.783	-173.567
	a_1	5.228	1.325	5.172	1.84	6.613	0.302

5.2.5 日补货总量和定价策略

假设商超的销售空间在一段时间范围内是固定的，求解某一蔬菜品类日补货总量 C_i 采用定单点法 Order Point Method 来计算，其中安全库存 Ω 和第 i 个蔬菜商品品类的每日平均总销量 D_i 是关键因素。

- 计算平均销量

  首先，估算预测每个蔬菜品类 7 月 1 日到 7 月 7 日销售数据的平均销量，这里可以直接使用 5.2.2 所预测得到的数据。假设 7 月 1 日到 7 月 7 日的预测平均销售量分别为 $D_i(1), D_i(2), \ldots, D_i(7)$，则预测得到的平均销售量

$$D_i=\frac{\sum_{j=1}^{7} D_i(j)}{n}$$

  在此基础之上，结合使用在 5.2.2 中所计算得到的第 7 月的季节性指数，从而完善定单点法对蔬菜类商品季节性销售特点描述的缺失。

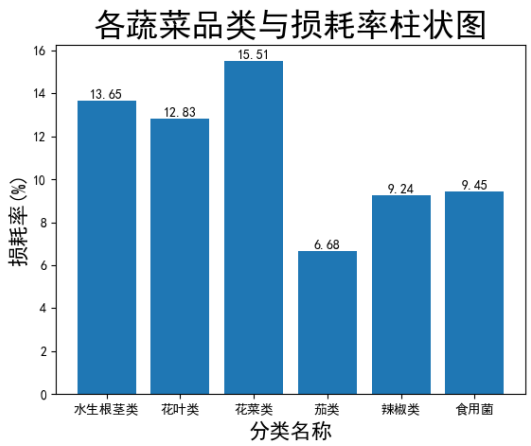

图 **6**　各蔬菜品类损耗率柱状图

- 考虑损耗率

  因为蔬菜类商品往往拥有一定的损耗率，因此在计算平均销量是需要将损耗率考虑在内。假设加权平均损耗率 γ，则考虑加权平均损耗率之后的平均销量为：

$$\bar{D}_i' = D_i \cdot (1 - \gamma) \tag{11}$$

  其中，γ 可以表示为某一品类中各单品损耗率所构成的向量 L_i 和某一品类中各单品销售比例所构成的向量 K_i 的乘积：

$$\gamma = L_i \cdot K_i \tag{12}$$

- 计算安全库存

  为了应对蔬菜类商品市场需求波动和供应侧不确定性的风险，考虑设置安全库存，即额外设置的库存量。结合现实情况判断，蔬菜类商品的销售量不可能能够达到无限大的程度，所以将平均销售量与安全库存量之和设置为最大日补货总量更符合实际情况。蔬菜类商品安全库存量通常取决于销量标准正态分布的临界值 Z 和历史需求的标准偏差 σ。

  计算临界值首先需要选择置信水平，然后依据标准正态分布表查找所选择的置信水平对应的临界值。我们选择 95% 的置信水平，其对应的临界值 Z 通常为 1.645。需求的标准差则可以通过计算销售量的标准差得到。

  安全库存的表达式可以表示为：

$$\omega = Z \cdot \sigma \tag{13}$$

  最终第 i 钟蔬菜品类的最大日补货总量可以表示为：

$$\max C_i = D_i' \cdot [1 + \Phi(7)] + \omega \tag{14}$$

在上述的日最大补货量约束条件之下，构建以最大化商超收益为目标，以各蔬菜品类的成本利润率为决策变量的多变元线性规划模型，再根据实际的销售情况进行加价

率的动态调整。在构建多变元线性规划模型的过程中，主要考虑以下两方面的的约束条件：“商超容量单品类需求约束，市场需求约束”。

各蔬菜品类历史成本（元）	
水生根茎类	11.22元
花叶类	5.44元
花菜类	7.57元
茄类	5.17元
辣椒类	7.61元
食用菌	7.2元

商超容量约束限制商超每日的日补货量不得超过前文中所计算得到的日最大补货量，而市场需求约束则限制商朝每日的日补货量应当不小于历史上该品类售卖的最小量，从而避免市场需求因补货量不足而没有被满足, 即：

$$\min C_i \leq C_i(t) \leq \max C_i(t)$$

假设第 i 个品类的利润率为 $G_i(t)$, 设该蔬菜品类中各单品销售价的加权平均值 $\bar{P}_i(t)$, 该蔬菜品类中各蔬菜单品批发价格的加权平均值为 $S_i(t)$，该蔬菜品类中各单品损耗率的加权平均值为 γ，那么 $G_i(t)$ 可以表示为：

$$G_i(t) = \frac{\bar{P}_i(t) - S_i(t) \cdot (1+\gamma)}{S_i(t) \cdot (1+\gamma)} \times 100\% \tag{15}$$

其中 $\gamma = L_i \cdot K_i$，L_i 为以该品类中各单品损耗率为元素的向量，K_i 为以该品类中各单品销售量所占总销量比例为元素的向量。待定的日补货量为 $C_i(t)$，则收益成本可以表示为

$$\max R_i(t) = [S_i(t) \cdot (1+\gamma)] \cdot G_i(t) \cdot C_i(t)$$

化简得：

$$\max R_i(t) = [P_i(t) - S_i(t) \cdot (1+\gamma)] \cdot C_i(t) \tag{16}$$

各蔬菜品类历史成本	
水生根茎类	11.22元
花叶类	5.44元
花菜类	7.57元
茄类	5.17元
辣椒类	7.61元
食用菌	7.2元

图 **7**　蔬菜各品类历史成本

为了更加贴近于商超成本加成定价的现实定价模式，并使商超能够对定价策略进行动态调整，依据图中信息，按照各蔬菜品类在 7 月 1 日到 7 月 7 日的预期销售总量，对各蔬菜品类确定加价率 w，那么此时收益可以表示为

$$\max R_i(t) = [P_i(t) - S_i(t) \cdot (1+\gamma)] \cdot C_i(t) \cdot (1+w) \tag{17}$$

加价率的确定主要依赖于各蔬菜类商品品类销量预期值，主要确定策略为：“低销高价，高销低价”。当某蔬菜品类预期总销售量小于 20 千克，加价率 w 取 50%；当某蔬菜品类预期总销售量大于等于 20 千克而小于 50 千克，加价率 w 取 20%；当某蔬菜品类预期总销售量大于 50 千克，加价率 w 取 10%。

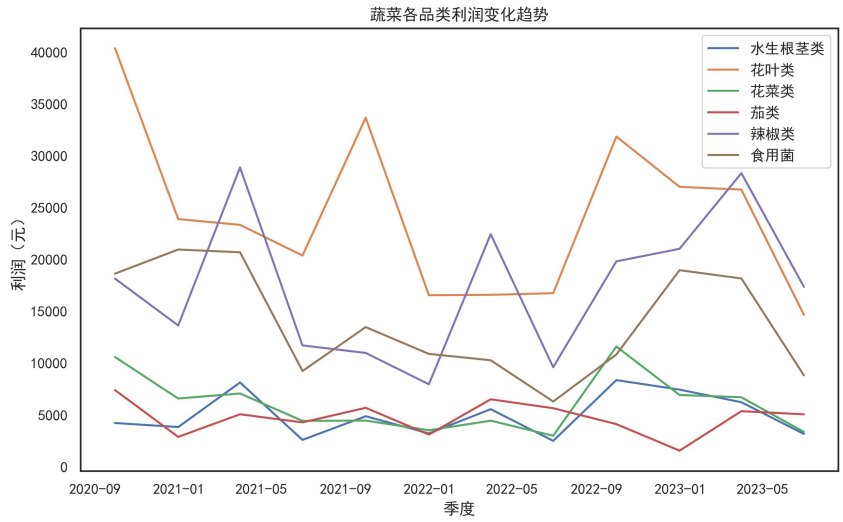

图 **8**　蔬菜各品类利润变化趋势

利用遗传算法对该多元线性规划进行求解，初始化一个包含多个个体的种群，每个个体表示一组可能的解，这些解包括各蔬菜品类的日补货总量 $C_i(t)$ 和定价策略 $P_i(t)$。对于每个个体，计算其适应度 (本模型中指商超的收益)$R_i(t)$, 即总利润。根据适应度选择父代个体，通常选择适应度较高的个体作为父代。从父代中选择两个个体，进行交叉操作，对两个个体的 $C_i(t)$ 和 $P_i(t)$ 进行某种组合，例如，可以计算两个父代个体在交叉点之前的 $C_i(t)$ 的均值，然后将均值分配给子代，并随机分配一个父代的定价策略给

子代，生成新的个体。对一些个体进行变异操作，引入随机性，以增加种群的多样性。计算新生成的个体的 $R_i(t)$，根据 $R_i(t)$ 选择存活个体，通常选择 $R_i(t)$ 较高的个体作为下一代的种群。重复选择父代到选择存货个体的步骤，直到满足停止条件（例如达到最大迭代次数或找到满意解）。得到找到的最优解，即最大化商超收益的蔬菜品类的 $C_i(t)$ 和 $P_i(t)$。

利用遗传算法求解多元线性规划模型的结果如下表所示：

日期	花菜类		花叶类		辣椒类	
	日补货总量（千克）	定价策略（%）	日补货总量（千克）	定价策略（%）	日补货总量（千克）	定价策略（%）
2023-07-01	53.354	50.06%	211.456	33.25%	75.144	75.14%
2023-07-02	20.122	25.25%	260.457	27.43%	108.957	108.95%
2023-07-03	41.145	50.15%	261.424	42.88%	109.551	109.55%
2023-07-04	39.265	87.36%	164.145	20.38%	43.092	43.09%
2023-07-05	50.924	41.12%	262.481	91.23%	110.027	110.02%
2023-07-06	31.45	39.27%	260.473	38.74%	108.927	108.92%
2023-07-07	29.175	52.17%	203.124	55.48	70.087	70.08%
日期	茄类		食用菌		水生根茎类	
	日补货总量（千克）	定价策略（%）	日补货总量（千克）	定价策略（%）	日补货总量（千克）	定价策略（%）
2023-07-01	9.791	96.00%	65.201	33.06%	21.827	84.08%
2023-07-02	8.732	41.80%	66.912	100.03%	11.628	44.21%
2023-07-03	5.622	64.84%	31.853	107.01%	15.661	88.24%
2023-07-04	10.835	105.51%	49.309	86.00%	9.709	32.74%
2023-07-05	3.295	95.34%	46.763	92.50%	18.394	52.91%
2023-07-06	6.624	109.11%	58.331	91.30%	10.224	33.92%
2023-07-07	6.153	84.10%	61.965	62.41%	14.355	38.11%

5.3 问题三求解

5.3.1 模型建立

在问题二建立基础之上，问题三增加了销售单品数量、订单订购量的最小陈列量等限制信息。在继续采用问题二所建立的多元线性规划模型基础之上，将问题三定性为组合优化问题，其模型为有约束的双变元线性回归，组合优化的目标是能够商超获得最大化收益。首先依据问题二中对品类所建立的模型中收益成本的表达形式，从单品角度给出蔬菜类商品总收益的另一种表达形式：

$$\max R_i(t) = \sum_{j=1}^{n} [C_{ij} \cdot (P_{ij} - S_{ij}) \cdot (1 - L_{ij})] \tag{18}$$

其中，i, j 分别表示蔬菜品类和该品类中所含有的单品，P_{ij} 表示第 i 类蔬菜品类中蔬菜的第 j 种单品的成本加成定价，L_{ij} 是第 i 个品类蔬菜商品中第 j 个单品的损耗率，n 为

商超选择的单品种类数量 $(27 \leq n \leq 33)$，设第 i 个蔬菜品类中，第 j 个单品的日补货量为 c_{ij}, 则日补货量满足 $C_{ij} \geq 2.5$ 的条件。

5.3.2 模型求解

问题三加入了限制条件后的模型求解过程如下：

1. 对预期利润的预测

   首先依据 2023 年 6 月 24 日到 6 月 30 日一周甚至 6 月份的可售蔬菜品种的数据对 7 月 1 日各单品销售量进行预测。计算每个单品的预期利润，有附件内容可得，已知各单品损耗率，可以由预测的各蔬菜商品品类批发价格乘以各单品权重，而得到各单品的批发价预测值，可以由预测的各蔬菜品类销量乘以各单品权重得到预测的单品销售量，在这里某单品的权重是指该单品历史销量占所属蔬菜品类历史销量的比例，预测的蔬菜品类批发价格和销售总量可以由时间序列模型预测得到。依据 (18), 可以计算得到对预期利润的预测。

分类名称	单品名称	预测单品销量（千克）	预测单品批发价（元）
花菜类	枝江青梗散花	6.5161	0.75806
花菜类	紫白菜(1)	0.0148	0.00173
花菜类	紫白菜(2)	0.0007	0.00008
花菜类	西兰花	30.7973	3.58286
花菜类	青梗散花	9.3952	1.09301

2. 求解最优化问题由结果图可以得到，收益不一定与商超选择的单品种类数量 n 成正相关关系。可暂时将 n 取为较为适中的 30，即取定前 30 个预期利润最高的单品。进一步将 n 向其临近的整数进行调整，排除比将 n 取定为 30 时收益比将 n 取定为 30 时低的取值。通过 pyThon 进行编程并结合预测和分析，最终将 n 取定为 29，该取值符合题目的基本条件。这些商品的总预期利润约为：60.08。

单品编码	单品名称	日补货总量（千克）	定价策略（%）
102900005115625	本地小毛白菜	3.172	111.79%
102900005115779	云南生菜	10.443	134.23%
102900005115878	茼蒿	9.014	142.23%
102900005115861	牛首油菜	18.215	94.02%
102900005116257	云南油麦菜	2.299	46.21%
102900005116530	云南油麦菜	17.72	47.46%
102900005116714	马齿苋	18.037	60.12%
102900005116899	黑油菜	24.97	128.36%
102900005122654	枝江红菜苔	2.129	59.75%
102900011013274	白玉菇(袋)	2	125.27%
102900011016701	白玉菇(袋)	26.919	58.49%
102900011033944	杏鲍菇(2)	3.151	138.72%
102900011034231	西峡花菇(2)	13	70.94%
102900011034330	双孢菇(盒)	10	53.00%
102900011034439	双孢菇(盒)	5	59.37%
102900011035078	菌菇火锅套餐(份)	2.762	102.07%
102900011035740	蟹味菇与白玉菇双拼(盒)	1	85.37%
102900011021842	牛排菇	7	71.14%
102900011023464	活体银耳	5.317	127.20%
102900011030059	杏鲍菇(250克)	13	98.47%
102900011030097	杏鲍菇(250克)	9	94.31%
102900011030110	杏鲍菇(250克)	15	137.31%
102900011030905	黑皮鸡枞菌(盒)	1	63.44%
102900011031100	鲜木耳(份)	19	118.59%
102900011031216	鲜木耳(份)	13	118.18%
106949711300259	金针菇(盒)	20	74.51%

上表是为 2023 年 7 月 1 日选择的 29 个单品种类选择的补货决策建议

5.4 问题四模型建立与求解

为了更好地解决补货和定价问题，以下是一些建议的数据采集点，为了更好地阐述所采集的数据对于更好指定补货和定价决策能起到一定作用，将相关数剧再放入模型中进行重新建模和求解：

1. 季节性因素与天气数据：
   天气和季节对蔬菜的需求和供应有很大影响，可能会导致不同品种的蔬菜类商品的市场需求呈现波动。为了在建模过程中引入对季节性因素的描述，在问题二建立时间序列模型时将季节性指数代入考虑，同时在利用定单点法对日补货量进行求解的过程中也也加入了季节性指数，通过每个月不同的季节性指数成绩因子反映相关量随季节的波动变化。
2. 库存量数据：
   了解当前库存水平能够帮助商家做出更加准确的补货决策，从而避免过度库存或者货源短缺。在问题二的求解过程中引入定单点法，通过计算日补货量的安全库存来满足各蔬菜品类销售总量与消耗所决定的库存量。
3. 促销活动数据
   在先前建立的模型当中并未深入对促销活动的影响进行分析，在问题四中将其引入。促销活动往往会在很大程度上影响供给需求关系和具体销售策略，记录一定量的促

销活动相关数据能够帮助商家更好地进行补货和定价决策，从而能够更好地利用因促销活动而带来地需求上涨，获得更高的经济效益。

对附件二商品中打折一栏信息进行相关处理分析，对打折商品信息进行基本数据统计，从而得出某一单品蔬菜打折地概率 P 与折扣的一个普遍值计算关系，如下图所示：

为了能够在促销活动中，在能够给出令客户满意的折扣同时，还能够使得生鲜商超获得最大化的收益，可以考虑建立 0-1 线性规划模型，设 0、1 整数变量 y 作为商场打折的逻辑变元。

$$y=\begin{cases}0, & \text{不进行打折}\\ 1, & \text{进行打折}\end{cases}$$

假设折扣率为 t，$t<1$。

问题三中 (18) 所确定的单品最大化收益的目标函数为

$$\max R_i(t)=\sum_{j=1}^{n}\left[C_{ij}\cdot\left(P'_{ij}-S_{ij}\right)\cdot(1-L_{ij})\right]$$

其中，

$$P'_{ij}=P_{ij}-y\cdot(1-t)\cdot P_{ij}$$

当逻辑变元 $y=0$ 时：

$$P'_{ij}=P_{ij}$$

当逻辑变元 $y=1$ 时：

$$P'_{ij}=t\cdot P_{ij}$$

可以使用蒙特卡洛法 (随机取样法)，应用概率论随机计算找出最优解的 n 和 C_{ij} 的取值。

4. 其它相关数据：
   - 顾客忠实度与满意度了解顾客的喜好与意见可以有效帮助商家更好地满足客户的实际需求，进而提高收益。
   - 蔬菜供应链数据：供应链中任何可能的延迟、中断或提前信息均可以帮助商家提前做出补货决策
   - 竞争商超的价格数据：竞争商超的价格可以帮助商家及时调整定价策略从而能够对顾客产生更大的吸引力。

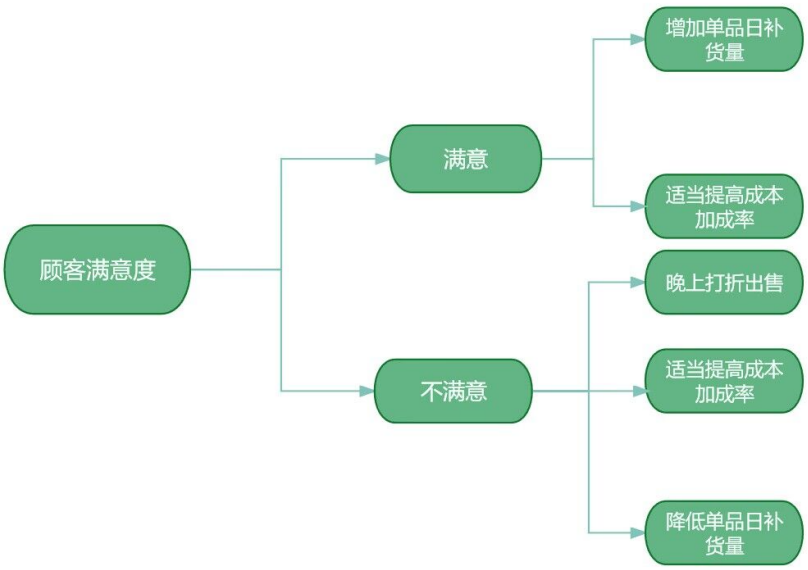

图 **9**　顾客满意度与需求情况

六、 模型分析

6.1 模型检验

1. 对季节性时间序列模型进行模型检验

   用 excel 表格自带的数据分析工具库-VBA，利用其指数平滑法的分析工具对各个蔬菜品类的销售总量随时间的变化趋势进行拟合，实际值与预测值如下图所示：

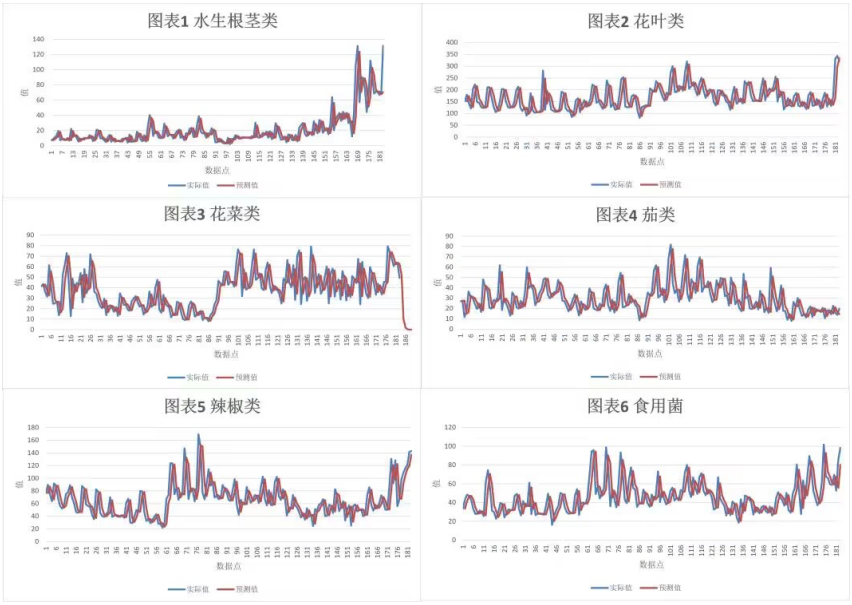

图 **10**　各蔬菜品类时间序列模型的拟合效果图

由图可见实际值与预期值在大部分点重合程度高，孤立点少切浮动水平不高，现预

测的销售量与实际销售量之间的平均绝对误差 (Mean Absolute Error, MAE) 较小，故该利用指数平滑法的季节性时间序列模型较为成功

2. 对各蔬菜商品单品的销售量所占各蔬菜品类销售总量权重参数的灵敏性分析

结合现实经验，每天的单品销量都会有一定程度上的波动，在计算平均加权值得过程中，权值大小会出现波动的情况。此时问题二的日补货量与定价策略将会与实际有所相异，故取花菜类进行分析，定价加成率对权重参数进行灵敏性分析：

取问题二结果中优化出的花菜类 1 7 日成本加成率进行分析，成本加成率的初始平均值为 49.34% , 花菜类单品各权重参数如表所示，从表中数据可以得到的花菜类总收益为：32.18。取最小权重 0.615 的紫白菜（2）的收益：$S(k,w)=\frac{\mathrm{d}k}{\mathrm{d}w}\cdot\frac{w}{k}$，计算结果为-8.2。故权重越小，定价加成率对收益的影响越低。权重较大，可能会影响预测的准确性。

	分类名称	单品名称	销量(千克)	rate	销售金额	加权后销售价
0	花菜类	枝江青梗散花	5821.571	0.13945904	47414.9774	6612.447011
1	花菜类	紫白菜(1)	13.251	0.00031744	180.2526	0.057218524
2	花菜类	紫白菜(2)	0.615	0.00001473	9.024	0.000132948
3	花菜类	西兰花	27514.727	0.65913089	269754.6958	177803.6533
4	花菜类	青梗散花	8393.786	0.20107790	58273.2038	11717.45372

6.2 模型评价

1. 遗传算法

优点：

- 具有强大的全局搜索能力，可以探索解空间中的不同区域，从而有望找到全局最优解。
- 遗传算法容易并行化，可以在多个处理器或计算节点上同时运行，加速问题求解的速度。
- 对于本问题大规模的多元线性规划问题很重要。

缺点：

- 需要大量的计算资源和时间来搜索解空间，计算开销可能会非常昂贵。
- 随机性强，使得同一问题的多次求解可能会得到不同的结果。可能导致不稳定的性能表现。
- 只能提供接近最优解的近似解，没有收敛性保证

参考文献

[1] 卢亚杰. 我国超市优质生鲜蔬菜动态定价问题研究 [D]. 北京交通大学,2010

[2] 乔雪. 考虑销售损失的生鲜产品的联合补货定价策略 [D]. 东南大学

附录 A　本文中使用的 python 程序代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['font.sans-serif'] = ['Simhei']
matplotlib.rcParams['axes.unicode_minus'] =False

df1 = pd.read_excel("附录1.xlsx" )
df2 = pd.read_excel("附录2.xlsx")
df3 = pd.read_excel("附录3.xlsx")
df4 = pd.read_excel("附录4.xlsx")
df44 = pd.read_excel("附录4拷贝.xlsx")

# 问题一
#汇总附件1和附件2 包含各种单品的流水信息
merge_data = pd.merge(df1, df2, on='单品编码', how='left')
#获取按品类计算的销售量
sale_catagory =
    merge_data.groupby("分类名称")['销量(千克)'].sum().sort_values(ascending=False)

#绘制箱线图
data1= merge_data[(merge_data.分类名称=='花叶类')& (merge_data['销量(千克)']<=5)
    &(merge_data['销量(千克)']>=0)]['销量(千克)']
data2= merge_data[(merge_data.分类名称=='辣椒类')&
    (merge_data['销量(千克)']<=5)&(merge_data['销量(千克)']>=0)]['销量(千克)']
data3= merge_data[(merge_data.分类名称=='食用菌')&
    (merge_data['销量(千克)']<=5)&(merge_data['销量(千克)']>=0)]['销量(千克)']
data4= merge_data[(merge_data.分类名称=='花菜类')&
    (merge_data['销量(千克)']<=5)&(merge_data['销量(千克)']>=0)]['销量(千克)']
data5= merge_data[(merge_data.分类名称=='水生根茎类')&
    (merge_data['销量(千克)']<=5)&(merge_data['销量(千克)']>=0)]['销量(千克)']
data6= merge_data[(merge_data.分类名称=='茄类')&
    (merge_data['销量(千克)']<=5)&(merge_data['销量(千克)']>=0)]['销量(千克)']
```

```
plt.figure(figsize=(12,8))
# plt.boxplot(data1,labels=['花叶类'],
#           whis=2,flierprops={'marker':'o'})

plt.boxplot([data1,data2,data3,data4,data5,data6],labels=['花叶类','辣椒类','食用菌','花菜类','水生根茎类','茄
          whis=2,flierprops={'marker':'o'},patch_artist=True,boxprops={'color':'k','facecolor':'#9999ff'})
plt.title('各品类销售量箱线图',fontsize=20)
plt.show()

#每一类的热力图及六类之间的热力图

pivot_cata = pd.pivot_table(merge_data,index="销售日期",
    columns="分类名称",values="销量(千克)",aggfunc=np.sum).fillna(0)

cor_cata=pivot_cata.corr()


plt.figure(figsize=(12,8))
sns.heatmap(cor_cata,cmap='coolwarm',annot=True,annot_kws={'size':20,'weight':'bold'},linewidths=.3,vmin=-1,
plt.title("各品类的相关系数（热力图）",fontsize=21)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('分类名称',fontsize=16)
plt.ylabel('分类名称',fontsize=16)
plt.show()


# 六个类各自单品热力图
cata1 = sale_catagory.index[0]
cata2 = sale_catagory.index[1]
cata3 = sale_catagory.index[2]
cata4 = sale_catagory.index[3]
cata5 = sale_catagory.index[4]
cata6 = sale_catagory.index[5]

cata1_data = merge_data[merge_data.分类名称==cata1]
cata2_data = merge_data[merge_data.分类名称==cata2]
cata3_data = merge_data[merge_data.分类名称==cata3]
cata4_data = merge_data[merge_data.分类名称==cata4]
cata5_data = merge_data[merge_data.分类名称==cata5]
cata6_data = merge_data[merge_data.分类名称==cata6]

pivot_cata1 = pd.pivot_table(cata1_data,index="销售日期",
    columns='单品名称',values="销量(千克)",aggfunc=np.sum).fillna(0)
pivot_cata2 = pd.pivot_table(cata2_data,index="销售日期",
```

```
    columns="单品名称",values="销量(千克)",aggfunc=np.sum).fillna(0)
pivot_cata3 = pd.pivot_table(cata3_data,index="销售日期",
    columns="单品名称",values="销量(千克)",aggfunc=np.sum).fillna(0)
pivot_cata4 = pd.pivot_table(cata4_data,index="销售日期",
    columns="单品名称",values="销量(千克)",aggfunc=np.sum).fillna(0)
pivot_cata5 = pd.pivot_table(cata5_data,index="销售日期",
    columns="单品名称",values="销量(千克)",aggfunc=np.sum).fillna(0)
pivot_cata6 = pd.pivot_table(cata6_data,index="销售日期",
    columns="单品名称",values="销量(千克)",aggfunc=np.sum).fillna(0)

cor_cata1=pivot_cata1.corr()
cor_cata2=pivot_cata2.corr()
cor_cata3=pivot_cata3.corr()
cor_cata4=pivot_cata4.corr()
cor_cata5=pivot_cata5.corr()
cor_cata6=pivot_cata6.corr()


fig,axs =plt.subplots(2,3,figsize=(12,8))

sns.heatmap(cor_cata1,cmap='coolwarm',linewidths=.3,vmin=-1,vmax=1,ax=axs[0,0],annot=False)
axs[0,0].set_title('花叶类')
sns.heatmap(cor_cata2,cmap='coolwarm',linewidths=.3,vmin=-1,vmax=1,ax=axs[0,1],annot=False)
axs[0,1].set_title('辣椒类')
sns.heatmap(cor_cata3,cmap='coolwarm',linewidths=.3,vmin=-1,vmax=1,ax=axs[0,2],annot=False)
axs[0,2].set_title('食用菌')
sns.heatmap(cor_cata4,cmap='coolwarm',linewidths=.3,vmin=-1,vmax=1,ax=axs[1,0],annot=False)
axs[1,0].set_title('花菜类')
sns.heatmap(cor_cata5,cmap='coolwarm',linewidths=.3,vmin=-1,vmax=1,ax=axs[1,1],annot=False)
axs[1,1].set_title('水生根茎类')
sns.heatmap(cor_cata6,cmap='coolwarm',linewidths=.3,vmin=-1,vmax=1,ax=axs[1,2],annot=False)
axs[1,2].set_title('茄类')

plt.tight_layout()
plt.show()


# 绘制批发价格随时间变化趋势
#汇总附件1和附件2 包含各种单品的流水信息
merge_data = pd.merge(df1, df2, on='单品编码', how='left')
#获取按品类计算的销售量
sale_catagory =
    merge_data.groupby("分类名称")['销量(千克)'].sum().sort_values(ascending=False)
# Merge attachment 3 (wholesale prices) with attachment 1 (product and category info)
pricing_data = pd.merge(df3, df1, on="单品编码", how="left")
cost_data = pd.merge(pricing_data,df44,on='单品编码',how='left')
```

```
# # Average wholesale price for each category in the past
# avg_wholesale_price = pricing_data.groupby('分类名称')['批发价格(元/千克)'].mean()
#
# Merge the average wholesale price with the loss rate from attachment 4
# cost_data = pd.merge(avg_wholesale_price, df4, left_on='分类名称', right_on='小分类名称',
    how="left")
# Calculate the cost per kg considering the loss rate
cost_data['cost_per_kg'] = cost_data['批发价格(元/千克)'] * (1 + cost_data['损耗率(%)'] /
    100)

# # Extract the relevant columns
# cost_data1=cost_data[['小分类名称','平均损耗率(%)_小分类编码_不同值']]
# cost_data = cost_data[['小分类名称', 'cost_per_kg']]
# 按季节可视化
quarterly_sales = cost_data.resample('Q', on='日期')['cost_per_kg'].sum()
sales_quarter_catagory = cost_data[merge_data['销售类型'] == '销售'].groupby(['分类名称',
    pd.Grouper(key='日期', freq='Q')])['cost_per_kg'].sum()
fig, ax = plt.subplots(figsize=(10, 6))
for category in sales_quarter_catagory.index.levels[0]:
    ax.plot(sales_quarter_catagory.loc[category].index,
        sales_quarter_catagory.loc[category].values, label=category)
ax.legend()
ax.set_xlabel('季度',fontsize=18)
ax.set_ylabel('各蔬菜品类批发价（元）',fontsize=18)
ax.set_title('蔬菜各品类批发价随时间变化折线图',fontsize=24)
plt.show()


# 获取各单品加权销售额
#汇总
merge_data = pd.merge(df1, df2, on='单品编码', how='left')

merge_price = pd.merge(merge_data, df3,left_on=["单品编码", "销售日期"],
    right_on=["单品编码", "日期"], how="left")
# 计算利润
merge_price["销售金额"] = merge_price["销售单价(元/千克)"] * merge_price["销量(千克)"]
merge_price["利润"] = merge_price["销售金额"] - (merge_price["销量(千克)"] *
    merge_price["批发价格(元/千克)"])

df = merge_price[ merge_price['销量(千克)'] < 10]
df = df.rename(columns={'分类名称': 'class', '销售日期': 'Date', '单品名称'
    :'categories','销量(千克)': 'Sales', '销售单价(元/千克)':
    'Price','销售金额':'total','利润':'profits','损耗率(%)':'loss'})
df4 =
    df4.rename(columns={'小分类编码':'分类编码','小分类名称':'分类名称','平均损耗率(%)_小分类编码_不同值':'平均
```

```
category_price_sum = df.groupby(['class','categories'])['Sales'].sum()
class_price_sum = df.groupby('class')['Sales'].sum()
category_total_sum=df.groupby(['class','categories'])['total'].sum()

category_price_sum = category_price_sum.reset_index()
class_price_sum = class_price_sum.reset_index()
category_total_sum = category_total_sum.reset_index()

classes = df['class'].unique()
class_data = {}
for class_ in classes:
    categories = category_price_sum.loc[category_price_sum['class'] == class_,
        'categories'].values
    sales_datas = category_price_sum.loc[category_price_sum['class'] == class_,
        'Sales'].values
    rate = sales_datas / class_price_sum.loc[class_price_sum['class'] ==
        class_,'Sales'].values
    price = category_total_sum.loc[category_total_sum['class']==class_,'total'].values
    total = rate * price

    data =
        pd.DataFrame({'catagories':class_,'products':categories,'kg':sales_datas,'rate':rate,'单品金额':price
    class_data[class_] = data

# rate_df = pd.DataFrame(class_data)

for class_ in classes:
    print(class_,'\n',class_data[class_].groupby('catagories')['加权后销售价'].sum())



#对时间重构数据
ts_data = merge_data.groupby(['销售日期', '分类名称'])['销量(千克)'].sum().reset_index()
ts_data = ts_data.pivot(index='销售日期', columns='分类名称', values='销量(千克)').fillna(0)


df1 = pd.read_excel("附录1.xlsx" )
df2 = pd.read_excel("附录2.xlsx")
df3 = pd.read_excel("附录3.xlsx")
df4 = pd.read_excel("附录4.xlsx")
df44 = pd.read_excel("附录4拷贝.xlsx")

#汇总附件1和附件2 包含各种单品的流水信息
merge_data = pd.merge(df1, df2, on='单品编码', how='left')
#获取按品类计算的销售量
sale_catagory =
    merge_data.groupby("分类名称")['销量(千克)'].sum().sort_values(ascending=False)
```

```
#按季节可视化
quarterly_sales = merge_data.resample('Q', on='销售日期')['销量(千克)'].sum()
sales_quarter_catagory = merge_data[merge_data['销售类型'] == '销售'].groupby(['分类名称',
    pd.Grouper(key='销售日期', freq='Q')])['销量(千克)'].sum()
fig, ax = plt.subplots(figsize=(10, 6))
for category in sales_quarter_catagory.index.levels[0]:
    ax.plot(sales_quarter_catagory.loc[category].index,
        sales_quarter_catagory.loc[category].values, label=category)
ax.legend()
ax.set_xlabel('季度')
ax.set_ylabel('销售量（千克）')
ax.set_title('蔬菜各品类销售量变化趋势')
plt.show()


#汇总附件1和附件2 包含各种单品的流水信息
merge_data = pd.merge(df1, df2, on='单品编码', how='left')
#获取按品类计算的销售量
sale_catagory =
    merge_data.groupby("分类名称")['销量(千克)'].sum().sort_values(ascending=False)

# #绘制品类与销量的柱状图
plt.figure(figsize=(10, 6))
x = sale_catagory.index
y = sale_catagory.values
plt.bar(x,y)
# figure1 = sns.barplot(x = sale_catagory.index,y = sale_catagory.values,data=cata_sales )

for a,b in zip(x,y):
    plt.text(a,b+10,b,ha='center',va='bottom',fontsize=12)
plt.title("不同品类的总销量",fontsize = 30)
plt.ylabel("总销量(kg)",fontsize = 24)
plt.tick_params(labelsize=18)
plt.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['font.sans-serif'] = ['Simhei']
matplotlib.rcParams['axes.unicode_minus'] =False
```

```
df1 = pd.read_excel("附录1.xlsx" )
df2 = pd.read_excel("附录2.xlsx")
df3 = pd.read_excel("附录3.xlsx")
df4 = pd.read_excel("附录4.xlsx")
df44 = pd.read_excel("附录4拷贝.xlsx")

#汇总
merge_data = pd.merge(df1, df2, on='单品编码', how='left')

merge_price = pd.merge(merge_data, df3,left_on=["单品编码", "销售日期"],
    right_on=["单品编码", "日期"], how="left")
# 计算利润
merge_price["销售金额"] = merge_price["销售单价(元/千克)"] * merge_price["销量(千克)"]
merge_price["利润"] = merge_price["销售金额"] - (merge_price["销量(千克)"] *
    merge_price["批发价格(元/千克)"])

df = merge_price[ merge_price['销量(千克)'] < 10]
df = df.rename(columns={'分类名称': 'class', '销售日期': 'Date', '单品名称'
    :'categories','销量(千克)': 'Sales', '销售单价(元/千克)':
    'Price','销售金额':'total','利润':'profits','损耗率(%)':'loss'})
df4 =
    df4.rename(columns={'小分类编码':'分类编码','小分类名称':'分类名称','平均损耗率(%)_小分类编码_不同值':'平均



category_price_sum = df.groupby(['class','categories'])['Sales'].sum()
class_price_sum = df.groupby('class')['Sales'].sum()
category_total_sum=df.groupby(['class','categories'])['total'].sum()

category_price_sum = category_price_sum.reset_index()
class_price_sum = class_price_sum.reset_index()
category_total_sum = category_total_sum.reset_index()

classes = df['class'].unique()
class_data = {}
for class_ in classes:
    categories = category_price_sum.loc[category_price_sum['class'] == class_,
        'categories'].values
    sales_datas = category_price_sum.loc[category_price_sum['class'] == class_,
        'Sales'].values
    rate = sales_datas / class_price_sum.loc[class_price_sum['class'] ==
        class_,'Sales'].values
    price = category_total_sum.loc[category_total_sum['class']==class_,'total'].values
    total = rate * price

    data =
        pd.DataFrame({'分类名称':class_,'单品名称':categories,'销量(千克)':sales_datas,'rate':rate,'销售金额'
```

```
    class_data[class_] = data

class_data['花叶类'].to_excel('花叶类信息表.xlsx')
class_data['辣椒类'].to_excel('辣椒类信息表.xlsx')
class_data['茄类'].to_excel('茄类信息表.xlsx')
class_data['水生根茎类'].to_excel('水生根茎类信息表.xlsx')
class_data['花菜类'].to_excel('花菜类信息表.xlsx')
class_data['食用菌'].to_excel('食用菌信息表.xlsx')

# 问题二

#对时间重构数据
ts_data = merge_data.groupby(['销售日期', '分类名称'])['销量(千克)'].sum().reset_index()
ts_data2 =merge_price.groupby(['销售日期', '分类名称'])['销售金额'].sum().reset_index()
pivot_ts=pd.pivot_table(ts_data,index='销售日期', columns='分类名称',
    values='销量(千克)').fillna(0)
pivot_ts2=pd.pivot_table(ts_data2,index='销售日期', columns='分类名称',
    values='销售金额').fillna(0)

pivot_ts2.to_excel('时间销量金额表.xlsx')
x=pivot_ts['花叶类'].values
y=pivot_ts2['花叶类'].values
plt.figure(figsize=(10,8))
plt.scatter(x,y)
plt.legend()
plt.title('销售量与加权后销售价散点图',fontsize=20)
plt.xlabel('销售量(kg)',fontsize=16)
plt.ylabel('加权后销售价(元)',fontsize=16)
plt.show()

# 线性回归
def forecast_sales(category):
    category_data = daily_sales[daily_sales['分类名称'] == category].reset_index(drop=True)
    category_data['day_num'] = np.arange(len(category_data))

    X = category_data[['day_num']]
    y = category_data['销量(千克)']

    # Split the data into training and testing sets (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
        random_state=42, shuffle=False)

    # Train a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict the sales for the test set
```

```
    y_pred = model.predict(X_test)

    # Calculate the mean squared error of the prediction
    mse = mean_squared_error(y_test, y_pred)

    # Predict the sales for the next 7 days
    next_7_days = np.array([max(category_data['day_num']) + i for i in range(1,
        8)]).reshape(-1, 1)
    future_forecast = model.predict(next_7_days)

    return future_forecast, mse

forecasts = {}
errors = {} #均方误差
for category in daily_sales['分类名称'].unique():
    forecasts[category], errors[category] = forecast_sales(category)

# Re-generating the day_num column for the entire daily_sales DataFrame 生成新的一列
daily_sales['day_num'] = daily_sales.groupby('分类名称').cumcount()

# Visualization based on forecasts and errors

# Plotting sales forecast vs actual sales for each category 实际销售与预测销售
plt.figure(figsize=(16, 8))

for category in daily_sales['分类名称'].unique():
    category_data = daily_sales[daily_sales['分类名称'] == category].reset_index(drop=True)
    X = category_data[['day_num']]
    y = category_data['销量(千克)']
    _, X_test, _, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
        shuffle=False)

    # Plotting the actual test data
    plt.scatter(X_test.day_num, y_test, label=f"Actual {category}", marker='x')

    # Plotting the forecasted data
    next_7_days = np.array([max(category_data['day_num']) + i for i in range(1, 8)])
    plt.plot(next_7_days, forecasts[category], label=f"Forecast {category}")

plt.title('销售预测与实际销售')
plt.xlabel('时间')
plt.ylabel('销量 (kg)')
plt.legend(loc='upper left')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

```
# Plotting the Mean Squared Error for each category
plt.figure(figsize=(10, 6))
plt.bar(errors.keys(), errors.values(), color='skyblue')
plt.title('每个类别的MSE')
plt.xlabel('类别')
plt.ylabel('MSE')
plt.grid(axis='y', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()

# 按品类计算利润和销售量
category_profit = merge_price.groupby("分类名称")["利润"].sum()
category_sales_volume = merge_price.groupby("分类名称")["销量(千克)"].sum()

# 绘制各品类销售量和利润关系图
plt.figure(figsize=(12, 6))
sns.scatterplot(x=category_sales_volume, y=category_profit, hue=category_profit.index,
    s=100)
plt.title("绘制蔬菜各品类销售量和利润散点图",fontsize=20)
plt.xlabel("各品类总销售量（kg）",fontsize=16)
plt.ylabel("总利润 (元)",fontsize=16)

plt.legend(title="蔬菜品类", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

# Group by '单品名称' and get the average sales for the entire dataset
avg_sales_overall = merged_data.groupby('单品名称')['销量(千克)'].mean().reset_index()

# Use this as the forecasted sales for July 1st
forecasted_sales_july1_overall = avg_sales_overall.copy()
forecasted_sales_july1_overall.rename(columns={'销量(千克)': '预测销量_7月1日'},
    inplace=True)

forecasted_sales_july1_overall.head()

# Merge forecasted sales with pricing and loss rate data
product_cost_data = pd.merge(forecasted_sales_july1_overall, pricing_data, on="单品名称",
    how="left")
product_cost_data = pd.merge(product_cost_data, attachment_4, left_on='分类名称',
    right_on='小分类名称', how="left")

# Calculate the cost per kg considering the loss rate
product_cost_data['cost_per_kg'] = product_cost_data['批发价格(元/千克)'] * (1 +
    product_cost_data['平均损耗率(%)_小分类编码_不同值'] / 100)

product_cost_data[['单品名称', '预测销量_7月1日', '批发价格(元/千克)',
```

```
    '平均损耗率(%)_小分类编码_不同值', 'cost_per_kg']].head()

# 问题三

# Define a function to set the markup rate based on forecasted sales volume for individual
    products
def determine_product_markup_rate(sales_forecast):
    if sales_forecast < 0.5:
        return 1.5 # 150% markup for low volume items
    elif sales_forecast < 1:
        return 1.3 # 130% markup for medium volume items
    else:
        return 1.2 # 120% markup for high volume items

# Calculate the expected profit for each product
product_cost_data['markup_rate'] =
    product_cost_data['预测销量_7月1日'].apply(determine_product_markup_rate)
product_cost_data['expected_price'] = product_cost_data['cost_per_kg'] *
    product_cost_data['markup_rate']
product_cost_data['expected_profit_per_kg'] = product_cost_data['expected_price'] -
    product_cost_data['cost_per_kg']
product_cost_data['total_expected_profit'] = product_cost_data['expected_profit_per_kg'] *
    product_cost_data['预测销量_7月1日']

# Sort the products based on the expected profit
sorted_products = product_cost_data.sort_values(by='total_expected_profit',
    ascending=False).drop_duplicates(subset='单品名称')

sorted_products[['单品名称', '预测销量_7月1日', 'cost_per_kg', 'expected_price',
    'total_expected_profit']].head()

# Select top 27-33 products based on the expected profit
selected_products = sorted_products.head(33)

# Calculate the total expected profit for these selected products
total_expected_profit = selected_products['total_expected_profit'].sum()

selected_products[['单品名称', '预测销量_7月1日', 'cost_per_kg', 'expected_price',
    'total_expected_profit']], total_expected_profit


# Set the replenishment volume for each selected product
min_display_volume = 2.5
selected_products['replenishment_volume'] =
    selected_products['预测销量_7月1日'].apply(lambda x: max(x, min_display_volume))

# Display the selected products, their expected price and replenishment volume
```

```
selected_products_final = selected_products[['单品名称', 'expected_price',
    'replenishment_volume']]
selected_products_final
selected_products_final.to_excel(r"C:\Users\86136\Desktop\selected_products_final.xlsx",
    index=False)
```