基于混合整数规划的生鲜商超补货计划和定价策略方案

摘要

为助力生鲜商超提高盈利能力，本文建立了动态调整规划模型、灰色系统预测模型、线性回归模型、二次规划模型和混合整数规划模型，分析蔬菜类商品销售量的分布规律和相关关系及其与成本加成定价的关系，并设计利润最大的补货计划和定价策略。

针对问题一，首先从天和年两个维度上，将销售数据根据单品加总后，以 20 分钟和一天为最小单位，量化分析蔬菜品类和单品销售量的分布规律，发现：日销售量的分布呈现双峰特征，最高峰和此高峰分别为早上十点和下午六点左右；年销售量分布呈现明显的季节性特征，总体上第三、四季度销量高。针对品类和单品之间的相关关系，首先对所有单品进行 K-means 聚类，所有单品可分为两类；使用**动态时间规整算法**刻画不同品类和单品之间的关联关系，发现花叶类蔬菜与其他五类蔬菜互为替代商品，花菜类和水生根茎类蔬菜互为互补商品；第二类单品之间的关联性普遍较高。

针对问题二，首先使用**灰色系统预测**未来一周所有单品的销售量和成本加成定价，并通过简单线性回归构建销售量与成本加成定价的关系式，得到销售量与成本加成定价之间的关系。其次，以负利润最小化为目标函数，每个单品的进货数量和成本加成定价为决策变量，综合考虑供求关系、价格稳定等约束条件建立二**次规划模型**，求解该模型，得到使得商超利润最大的销售组合，即选择来自六大品类的 51 个单品，给出品类的补货计划和定价策略的详细数据，每天的最大利润达到 1686.28 元，通过单品的最优销售组合来设计品类的最优销售组合，最大程度地使商超利润最大化。

针对问题三，在问题二的基础上，在目标函数中加入是否进货的 0-1 变量 h_i，综合考虑供求关系、价格稳定、单品个数、单品最低陈列量等约束条件，构建**混合整数规划模型**。根据逻辑分析，问题二中的最优解与问题三中的最优解关联密切，问题二中满足最低陈列量的最优解也为问题三中的最优解，问题二中不满足最低陈列量的最优解在问题三中的最优解为 2.5。求解得到，在选择 33 个单品时利润最大，每日的净利润能够达到 1574.09 元，并给出 33 个单品每日的补货计划和定价策略。

针对问题四，从供应端、销售端、管理端、竞争对手和其他五个方面分析需要收集的数据，包括供应商画像数据、库存管理数据、市场价格数据、客户画像数据、销售周期数据、天气数据等。并根据这些数据进一步优化问题二、问题三中的规划模型，帮助商超更好地制定补货计划和销售策略。

关键词：动态调整规划算法　灰色系统预测　二次规划模型　混合整数规划

目录

1 问题重述

1.1 问题背景

蔬菜类商品保鲜期较短，常需尽快销售以避免品相随销售时长增加变差而导致的亏损，因此，生鲜商超通常会根据历史销售和需求数据，每天进行合理的补货。

由于蔬菜品种和产地多元，且进货时间集中在凌晨，商超一般需要在具体单品和进货价格未知的情况下，做出当日各蔬菜的补货计划。对于蔬菜定价，商超常采用成本加成定价法，并对运损和差品相的商品打折销售。为使得利润最大化，合理的补货计划和定价策略对商超而言尤为重要。市场中蔬菜类商品的供求规律往往有迹可循，从需求侧看，蔬菜类商品的销售量与时间往往存在着一定的关联；从供给侧看，蔬菜供应的品种在 4 月到 10 月更多样。商超可以借助可靠的市场供求分析，做出合理的决策。

1.2 问题提出

根据题目背景和附件数据，本题要求建立数学模型，解决以下问题：

1、分析蔬菜各品类及单品销售量的分布规律和相互关系，以寻找蔬菜类商品不同品类和不同单品之间的相关关系。

2、为使商超利润最大化，分析各品类的销售总量与成本加成定价的关系，并参考问题 1，决定 2023 年 7 月 1-7 日的日补货总量，给出定价策略。

3、针对单品制定补货计划，建立模型选取最有利可图的 27-33 个销售单品，在控制单品订购量和尽量满足市场需求的前提下，结合 2023 年 6 月 24-30 日的信息，决定 7 月 1 日的单品补货量和定价策略，使商超获得最大利润。

4、进一步考虑影响补货和定价决策的相关数据，阐明相关数据的具体作用，优化蔬菜商品的补货和定价策略

2 问题分析

2.0.1 问题一的分析

问题一要求分析不同品类和不同单品销售量的分布规律及相关关系，根据背景材料可得，蔬菜销售量在时间分布上可能存在一定规律，将销售数据按日、年重新统计，可视化作图分析销售量的分布规律。对于相关关系，为便于分析，首先对不同单品的销售量进行聚类，选择相关程度可能较高的单品，其次考虑到时间序列的滞后效应，使用动态时间调整算法分析不同品类和不同单品之间的相关关系。

2.0.2 问题二的分析

题目二要求分析销售量与成本加成定价之间的关系，并根据品类为未来一周 7 月 1 日-7 日确定补货计划和定价策略，使得商超的利润最大化。考虑到同一品类之下各单品的销售情况和盈利能力各异，单考虑六大品类的补货计划和定价策略可能不能够真正使得商超的利润最大，因此，本文从各单品出发，从单品的维度上，分析单品的销售量与成本加成定价之间的关系，进一步构

建二次规划模型，然后通过解得单品的最优补货计划和定价策略得到各品类的补货计划和定价策略，并分析各品类的销售数量与成本加成定价之间的关系。

2.0.3 问题三的分析

题目三要求根据 6 月 24-30 日的可售品种，在单品总数和最小陈列订购量的限制条件下，确定使得市商超利润最大的补货计划和定价策略。在第二问的基础上，通过增加 0-1 变量和约束条件，进一步优化二次规划模型，并通过数学分析求解最优的补货计划和定价策略。

2.0.4 问题四的分析

题目四要求对有助于制定补货计划和定价策略的相关数据提出建议和意见，影响决策的变量众多，为帮助商超更好地制定补货计划和定价策略，从供应端、销售端、管理端、竞争对手和其他五个方面就数据收集给出有用建议，并将相应变量加入问题二的二次规划模型中，进一步优化模型。

3 问题假设

模型假设：
1、品类和单品的销量在时间轴上有确定的分布，销量在时间上的分布能够反映商品一定的特征。
2、短时间内，单品的进货价格、销售量、销售价格相对稳定，不存在重大自然灾害、重要政策出台等外生变量。
3、根据需求曲线，单品的销售量和价格呈线性关系。

4 符号说明

符号	说明
t_{ij}	动态时间规整算法的最小累计距离
N_i	蔬菜单品 i 的销售量
s_i	蔬菜单品 i 的进货量
c_i	蔬菜单品 i 的进货价格
d_i	蔬菜单品 i 的折损率
MP_i	蔬菜单品 i 的历史最高价
h_i	0-1 变量，h_i=1 表示进货蔬菜单品 i

5 问题一的模型建立与求解

5.1 数据预处理

问题一需要分析不同品类或不同单品销售量的分布规律和相关关系，根据附件一、附件二，能够得到蔬菜类商品每个单品详尽的销售记录，包括销售价、销售量、销售时间等数据。为便于分析，对数据进行再分类，将 2020 年至 2023 年三年中，同一品类和同一单品的每日的销售数据相继加总，即三年视为一日，得到每一品类和每一单品的三年日销售总量。销售量数据中存在负值，即有退货情况，根据销售时间，在日销售量中减去相应的退货量，修正得到准确的日销售总量。在此基础上，加总计算得同一品类和同一单品的三年月销售总量。

5.1.1 销售量的分布规律

1、日分布规律

根据人们的生活习惯，蔬菜类商品在一天中的销售分布通常有着一定规律，因此，首先考察蔬菜品类的日销售量的分布规律。由于商品并不是每分每秒都被销售，需对日销售量做进一步的划分，为细致刻画分布特征，从 0 点开始，计算每 20 分钟内的销售量。因此，一天一共有 72 个单位，一个单位表示 20 分钟，对六大品类分别计算每 20 分钟内的销售量。在此基础上，对销售量进行可视化分析。

观察图 1 可得，六大品类的日销售量呈现双峰特征，在早上 10 点（单位 30 处）迎来最高峰，其中，花叶类的销售总量最大，高约 10000 千克，即平均而言，商超每天 10 点到 10 点 20 分内的叶菜销售量约 10 千克，其他五类蔬菜的销售量明显低于花叶类，但同样在 10 点达到最高。下午 14 点左右，销售量达到低谷，之后逐步上升。六大品类销售量的次高峰在下午 18 点左右达到，叶菜类蔬菜的销售量同样远高于其他五类，花菜类和水生根茎类蔬菜位列第二、第三，茄类、辣椒类和食用菌三类蔬菜的销售量相对较低。

一般情况下，商超在日常生活中常在 9 点左右开门营业，由于人们大多习惯于早晨买菜，商超开始营业后不久便会迎来购买高峰。高峰过后销售量逐步下降，在中午午餐时间达到最低。下午 18 点左右，正逢人们下班时间，需购买食材准备晚餐，商超销售引来小高峰，之后随着时间逐步下降。图一所示的蔬菜品类销售量的分布规律符合常理，每日的销售量显示双峰特征，在上午 10 点左右达到最高峰，中午 14 点左右迎来低谷，下午 18 点左右达到次高峰。

分析单品的分布规律，同样考虑不同单品的日销售量，采用上述相同方法。分析图 2 发现，从总体分布特征上看，各单品仍主要体现出双峰特征，上午 10 点左右达到最高峰，中午午休之间为低谷，下午下班之后和晚餐之前的时间达到此高峰，对于销售量大的单品，例如叶菜品类的单品，呈现明显的大 “M” 形状，对于销售量较小的单品，例如食用菌类的单品，大体上呈小 “m” 形状，销售量越大的单品双峰分布特征越明显。

从不同单品的分布特征来看，对于日消耗量较少或不常见的蔬菜单品，例如红尖椒（图 2-3），销售量在一天中波动性较大，存在一定的随机性。相较之下，消耗量较大或受众较广的蔬菜单品，如上海青（图 2-2）和茼蒿（图 2-5），销售量存在着明显的大 “M” 分布规律，波动性较小。对于销售分布波动性较大的单品，此类蔬菜单品往往每日的消耗量较小且不固定，因此，人们的购买行为存在随机性，销售分布也存在着较大随机性。

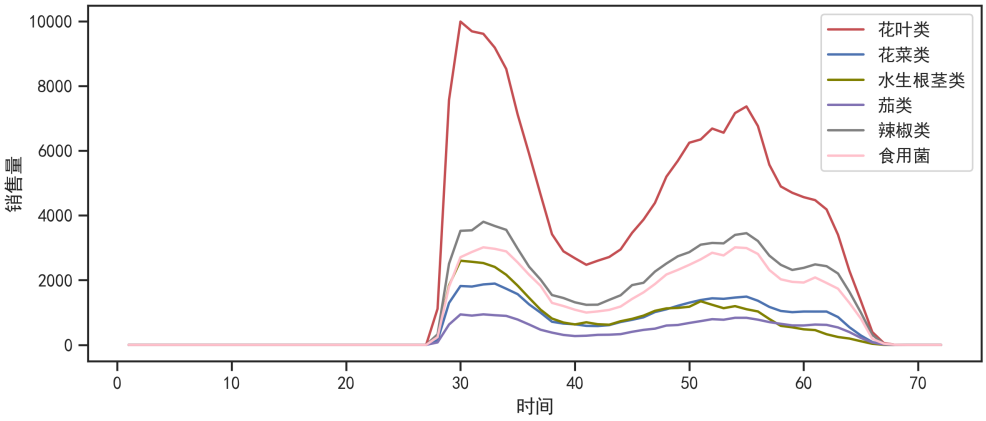

图 1: 各品类销售量随时间的变化

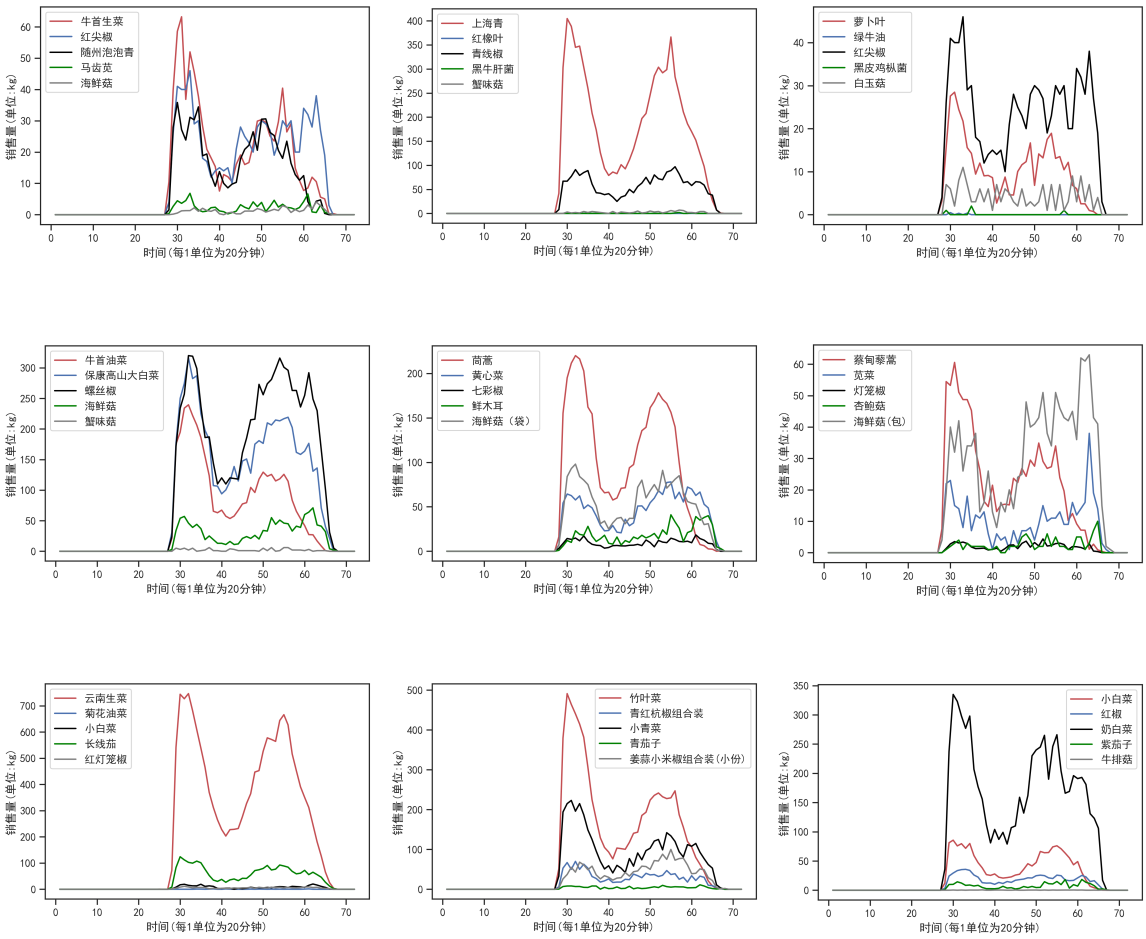

图 2: 不同单品日销售量的分布规律

2、年分布规律

根据背景材料，蔬菜品种在 4 月到 10 月较为丰富，不同品类和不同单品的销售量可能存在着季节性趋势。以天为单位，考察不同品类一年中销售量的分布规律，由图 3 所示，花叶类和辣

椒类的销售量从 7 月开始有着明显的上升趋势，第三季度的销售量明显增加，销售量波动性也随之增加。花菜类、茄类和水生根茎类的销售量在全年中起伏较小，但在第三季度也有小幅上升。食用菌类的销售量在第四季度最高。此外，六大品类的销售量呈锯齿状，进一步划分时间可推测，蔬菜品类一般在周末的销售量最大，而在周中最低。

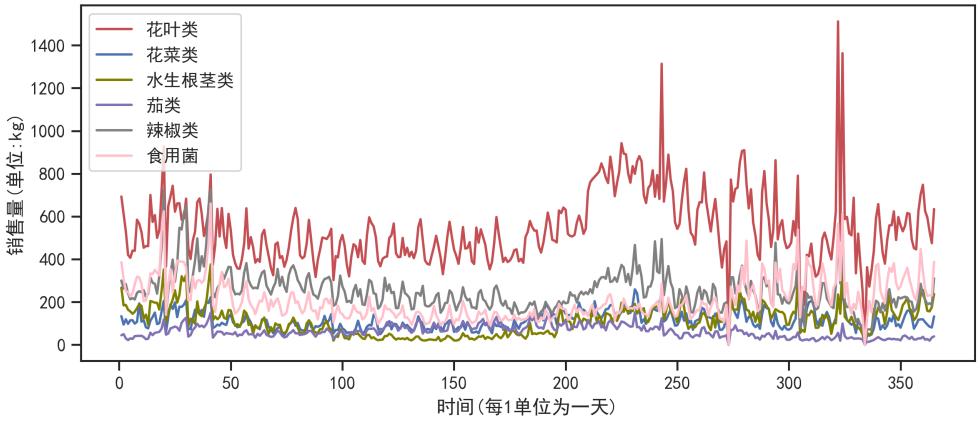

图 3: 不同品类年销售量的分布规律

看不同单品年销售量的分布，以图 4、图 5 为例，不同单品的年销售量也有着明显的季节性趋势。如图 4 所示，竹叶菜在 4 月至 7 月的销售量显著增加，年销售量呈钟形分布；金针菇、黄心菜的销售量在 7 月也有着小幅上升，在第三、第四季度的销售明显更高；而海鲜菇的年销售量主要集中在第一、第二季度。如图 5 所示，西峡香菇、保康高山大白菜、牛首油菜三个单品的销售量从六月左右开始波动上升，第四季度的销售量明显增加。小米椒的销售量在一年中虽起伏不大，但在六月也有小幅上升。总体而言，不同单品的年销售量存在着明显的季节性趋势，第三、第四季度的销售量普遍较高。

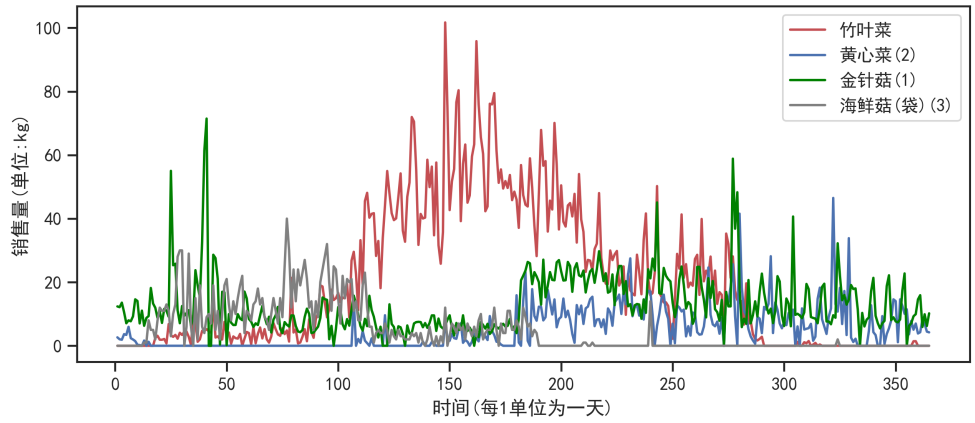

图 4: 不同单品年销售量的分布规律-1

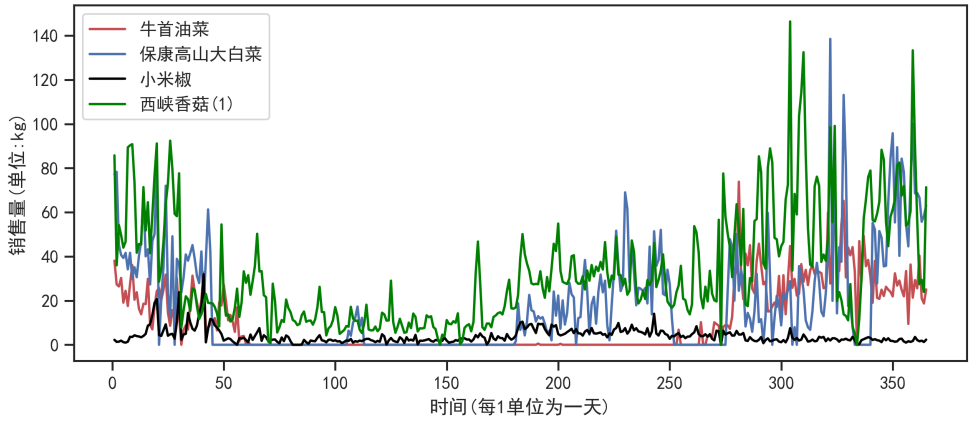

图 5: 不同单品年销售量的分布规律-2

5.1.2 基于 K-means 聚类和动态时间规整的相关关系分析

本题中销售数据均为时间序列，时间序列可能存在着滞后效应，可能会对准确分析品类和单品销售量之间的关系有所影响。此外，通过每条销售记录累加计算而得的销售总量存在离散程度变大的问题，此时一般算法在计算距离，即在计算不同品类和单品销售量之间的相关关系时易受干扰。因此，为应对上述问题，选择动态时间规整（DTW）来分析关联关系。

DTW 算法是一种基于动态规划（DP）的模型匹配方法，可以解决时间序列数据中不同程度和幅度的识别问题 [4]。首先，对 $(\mathrm{a}_1, \mathrm{a}_2, \cdots, \mathrm{a}_\mathrm{n})$ 和 $(\mathrm{b}_1, \mathrm{b}_2, \cdots, \mathrm{b}_\mathrm{m})$ 的两个序列进行最大程度的对齐，构造 n*m 维的矩阵网格，并计算两个序列元素之间的局部距离：

$$d_{ij} = (a_i - b_j)^2, \quad i = 1, n \quad j = 1, m \tag{1}$$

根据局部距离，即图 3 中的矩阵元素（i，j），进一步使用动态规划算法计算两个序列之间的累积距离，最小累积距离即两个序列之间的最佳匹配路径：

$$t_{ij} = d_{ij} + \min\left(t_{i-1,j-1}, t_{i-1,j}, t_{i,j-1}\right) \tag{2}$$

（1）边界条件：路径从左下角 t_{11} 开始，到右上角 t_{nm} 结束。

（2）单调性条件：路径不会返回自身，保证路径上的点随着时间单调进行。

（3）步长条件：路径逐个单位前进，只能与自身相邻的点对齐，保证序列中每个数据都在路径中出现。$\dfrac{1}{1+\sqrt{d_{mn}}}$ 即为两个序列的相似度。

六大品类的 DTW 系数如图 3 所示，观察热力图可得，花菜类蔬菜与水生根茎类存在最强的相关关系，辣椒类蔬菜与食用菌类蔬菜的相关程度也较高。可能由于菜品的搭配以及人们的饮食喜好，花菜类与水生根茎类蔬菜、辣椒类与食用菌类蔬菜之间的关系体现为互补商品，人们常常一同购买西兰花等花菜类蔬菜和莲藕等水生根茎类蔬菜、小米椒等辣椒类蔬菜和杏鲍菇等食用菌。此外，花叶类蔬菜与其他五类蔬菜的相关性均较小，花叶类蔬菜可能与其他蔬菜互为替代商品，人们在购买花叶类蔬菜后往往不会选择购买其他五类蔬菜。

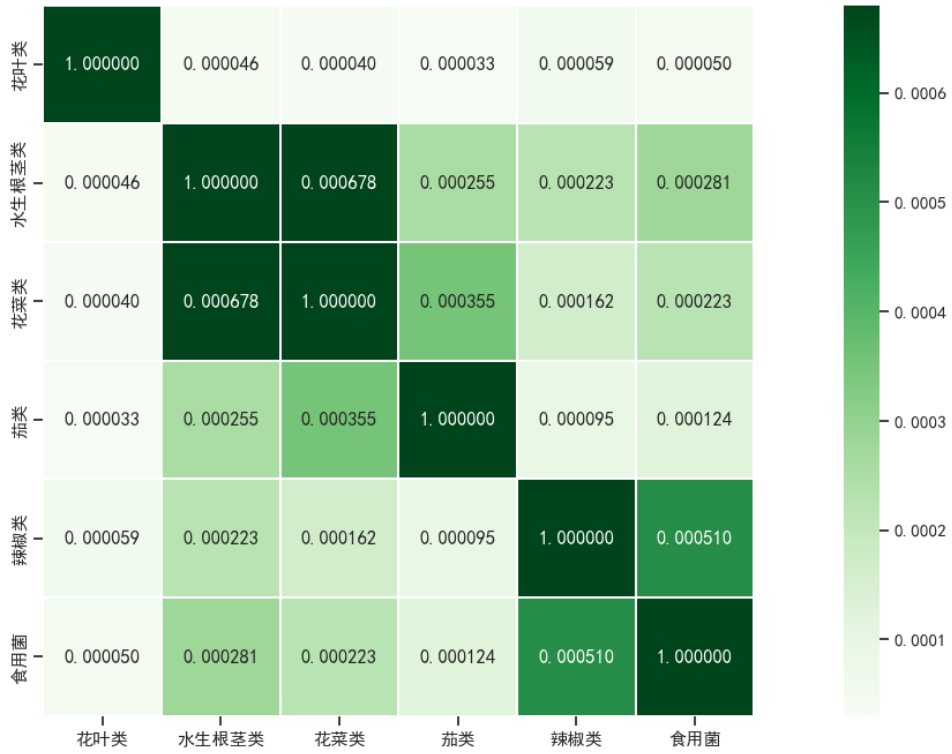

图 6: 各品类之间的相互关系

对于不同单品，由于单品种类众多，首先对所有单品进行 K-means 聚类，将不同单品根据各自特征分类，寻找可能存在关联关系的不同单品。K-means 聚类算法通过选取合适的距离公式，来衡量不同数据对象的相似度，数据之间的距离与相似度负相关，即相似度越小，距离越大。

K-means 聚类算法首先需要从给定的数据对象中随机指定初始聚类数目 k 和对应的初始聚类中心 C，并计算初始聚类中心到其余数据对象的距离，由于销售数据存在一定的波动性，为了更平滑地计算距离，先将数据进行标准化处理，然后选取欧式距离进行计算，聚类中心到空间中其他数据对象的欧氏距离公式为：

$$d\left(x, C_i\right)=\sqrt{\sum_{j=1}^{m}\left(x_j-C_{ij}\right)^2} \tag{3}$$

其中，x 是数据对象，C_i 是第 i 个聚类中心，m 是数据对象的维度，x_j、C_{ij} 为数据对象 x 和聚类中心 C_i 的第 j 个维度的属性值。

根据欧氏距离，衡量相似度，将与聚类中心相似度最高的目标数据分配 C_i 的簇中，分配完毕后，将 k 个簇中的数据对象取平均值，形成新一轮的聚类中心，从而降低数据集的误差平方和，计算公式为：

$$SSE=\sum_{i=1}^{k}\sum_{x\in C_i}\left|d\left(x, C_i\right)\right|^2 \tag{4}$$

SSE 的数值大小作为衡量聚类结果好坏的依据，当其不再变化或收敛时，停止迭代，得到最终结果，流程图如图。

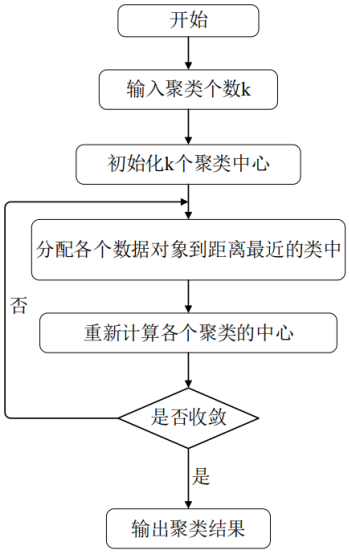

图 7: K-means 算法流程

迭代得到结果后，根据轮廓系数来选择最优的聚类数目 k。轮廓系数是一种评估聚类效果的指标，结合了聚类的凝聚度和分散度，轮廓系数越大，表示聚类效果越好。由图 x 所示，当聚类数目为 2 时，轮廓系数最高，因此将不同单品分为 2 类。

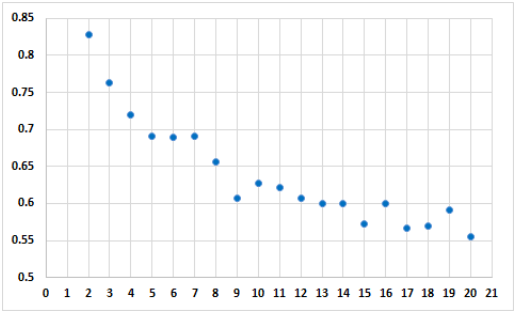

图 8: 轮廓系数

观察两类数据发现，第二类单品数目只有 13 种，主要包括娃娃菜、大白菜、云南生菜等花叶菜，小米椒、螺丝椒等辣椒类蔬菜，以及香菇、金针菇等食用菌。同样使用 DTW 算法分析第二类单品之间的相关关系，由热力图可得，云南油麦菜与泡泡椒、云南油麦菜与西峡香菇、小米椒与芜湖青椒的系数最大。总体上看，云南油麦菜、娃娃菜、紫茄子、泡泡椒和小米椒整体上的与其他单品的相关性较强，可能为人们日常生活中常购买的销售组合。而西兰花和净藕两个单品与其他单品的相关性均较小，可能与其他单品之间互为替代商品关系。

类别	单品
第一类	……
第二类	云南生菜、大白菜、云南油麦菜、娃娃菜、西兰花、净藕、紫茄子、泡泡椒、芜湖青椒、小米椒、螺丝椒、西峡香菇、金针菇

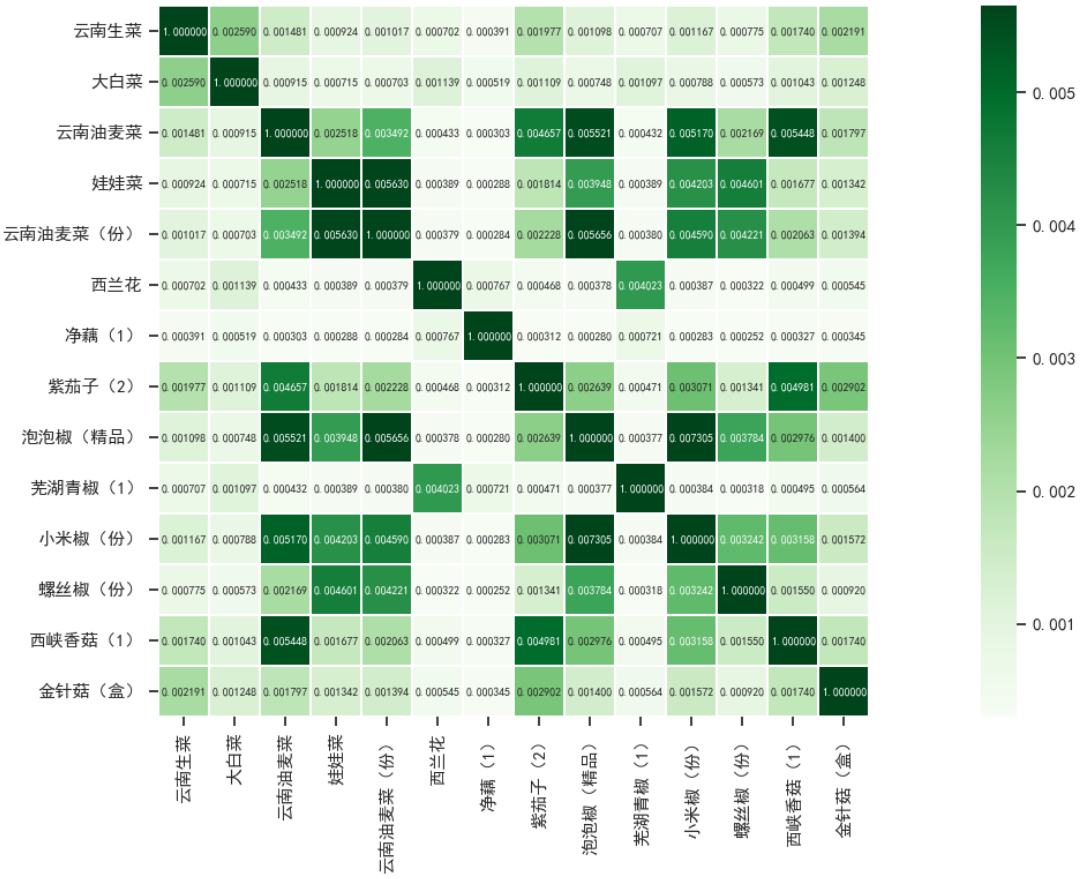

图 9: 第二类单品之间的相互关系

6 问题二的模型建立与求解

题目二要求分析销售量与成本加成定价之间的关系，并根据品类为未来一周 7 月 1 日-7 日确定补货计划和定价策略，使得商超的利润最大化。考虑到同一品类之下各单品的销售情况和盈利能力各异，单考虑六大品类的补货计划和定价策略可能不能够真正使得商超的利润最大，因此，本文从各单品出发，从单品的维度上，分析单品的销售量与成本加成定价之间的关系，进一步构建二次规划模型，然后通过解得单品的最优补货计划和定价策略得到各品类的补货计划和定价策略，并分析各品类的销售数量与成本加成定价之间的关系。

6.1 基于灰色系统预测的销售量与成本加成定价

题目二首先要求分析销售量与成本加成定价之间的关系。根据定义，成本加成定价 = 销售价格 = 单位成本 * (1+ 加成率)，由于此题缺少其他成本，例如租金、工资等数据，将单位成本近似视为进货价格，则销售量 = 进货价格 * (1+ 加成率)。为得到销售量与成本加成定价之间的关系，我们首先使用灰色预测，用过去三个月的销售数据预测未来一周 7 月 1 日-7 日的销售量和销售价格。

灰色预测法是一种基于少量、不完全信息的预测方法，其适用于一些非线性、非平稳的系统，

尤其在短期预测和趋势分析中作用明显。[3] 因此，考虑到本题中所用数据为时间序列，在不同时间不同单品的销售情况各异，选择使用灰色预测，建立 GM（1，1）模型，进行未来一周的短期预测。具体步骤为：

首先,将前三个月每一单品的销售量、价格分别作为原始序列数据 $x=\{x(t_1),x(t_2),\cdots,x(t_n)\}$, 计算序列生成数据 $y=\{y(t_1),y(t_2),\cdots,y(t_n)\}$，其中 $y(t_k)=\sum_{i=1}^{k}x(t_i)$；并构建基本形式：

$$\frac{dy(t)}{dt}+ay(t)=b \tag{5}$$

其次，结合模型均值形式

$$x(t_k)+a[\frac{1}{2}y(t_k)+\frac{1}{2}y(t_{k-1})]=b \tag{6}$$

然后，运用最小二乘法求解参数 a 和 b 的估计值

之后求解时间相应函数

$$(\hat{y}(t_1)+\frac{\hat{b}}{\hat{a}})exp[\hat{a}(t_k-t_1)]-\frac{\hat{b}}{\hat{a}} \tag{7}$$

并求得累加生成序列的模拟值 $\hat{y}(t_1)$=$\hat{y}$=$\{\hat{y}(t_1),\hat{y}(t_2),\cdots,\hat{y}(t_n)\}$, 最终解得销售量、价格的模拟值 $\hat{x}$=$\{\hat{x}(t_1),\hat{x}(t_2),\cdots,\hat{x}(t_n)\}$ ，部分预测结果如图所示。

图 10: 灰色预测模型预测结果展示

对结果进行后验差比检验，一般而言，后验差比值 C 值小于 0.35 则模型精度高，C 值小于 0.5 说明模型精度合格，C 值小于 0.65 说明模型精度基本合格，如果 C 值大于 0.65，则说明模型精度不合格。部分检验结果如图所示，对所有单品的检验结果进行统计，其中精度高占比 71.31%、监督合格占比 11.12%、基本合格占比 8.72%、不合格占比 6.85%，总体上预测的精度较高，预测效果良好。

表 1: 后验差比检验结果

单品	发展系数 a	灰色作用量 b	后验差比 C 值
云南油麦菜 (份)	0	18.024	0.945
小皱皮 (份)	0.001	32.562	0.147
双孢菇 (盒)	0	22.103	0.917
红椒 (2)	-0.002	77.739	0.489
红薯尖	0.006	27.85	0.098
竹叶菜	0.004	48.762	0.258
……	……	……	……

表 2: 精度统计

精度高	<0.35	71.31%
精度合格	<0.5	11.12%
精度基本合格	<0.65	8.72%
精度不合格	>0.65	6.85%

由此，通过灰色预测模型，得到未来一周内预测销售量和销售价格。

表 3: 7 月 1 日-7 日各单品的预测价格

日期	云南油麦菜（份）	小皱皮（份）	双孢菇（盒）	红椒	红薯尖	……
7.1	4.112 元/kg	2.176 元/kg	5.002 元/kg	20.307 元/kg	5.028 元/kg	……
7.2	4.115 元/kg	2.131 元/kg	4.999 元/kg	20.461 元/kg	4.885 元/kg	……
7.3	4.118 元/kg	2.088 元/kg	4.996 元/kg	20.616 元/kg	4.743 元/kg	……
7.4	4.120 元/kg	2.043 元/kg	4.992 元/kg	20.771 元/kg	4.601 元/kg	……
7.5	4.123 元/kg	1.999 元/kg	4.989 元/kg	20.926 元/kg	4.460 元/kg	……
7.6	4.126 元/kg	1.955 元/kg	4.986 元/kg	21.081 元/kg	4.320 元/kg	……
7.7	4.129 元/kg	1.911 元/kg	4.983 元/kg	21.237 元/kg	4.181 元/kg	……

表 4: 7 月 1 日-7 日各单品的预测销量

日期	云南油麦菜（份）	小皱皮（份）	双孢菇（盒）	红椒	红薯尖	……
7.1	2.532kg	1.865kg	6.746kg	8.658kg	6.883kg	……
7.2	2.596kg	1.913kg	6.623kg	8.387kg	6.562kg	……
7.3	2.66kg	1.962kg	6.5kg	8.117kg	6.24kg	……
7.4	2.724kg	2.011kg	6.377kg	7.847kg	5.919kg	……
7.5	2.788kg	2.06kg	6.254kg	7.578kg	5.599kg	……
7.6	2.853kg	2.109kg	6.131kg	7.308	5.279kg	……
7.7	2.917kg	2.158kg	6.009kg	7.039kg	4.96kg	……

6.2 基于线性回归的销售量与成本加成定价的关系分析

根据需求曲线，当价格上升时，需求（即销售量）会下降，销售量与销售价格之间一般意义上呈现线性关系 [1]：

$$N(p) = a + bp \tag{8}$$

其中 N 为销售量，p 为销售价格，a、b 为常数。

为明确销售量与销售价格之间的关系，以未来一周的销售量为因变量，以销售价格为自变量，进行简单线性回归。回归结果如图所示，p 值均，说明系数显著，拟合效果良好，由此得到不同单品之间，销售量与销售价格的关系：

$$N_i(p_i) = a_i + b_i p_i \tag{9}$$

表 5: 后验差比检验结果

单品	拟合方程	p-value
云南生菜	$\mathbf{N}_1 = —0.26811 + 0.50182\mathbf{p}_1$	0.00
菜心)	$\mathbf{N}_2 = —0.37865 + 0.59756\mathbf{p}_2$	0.00
螺丝椒	$\mathbf{N}_3 = 24.352157 + 1.67852\mathbf{p}_3$	0.00
竹叶菜	$\mathbf{N}_4 = 14.80125 + 0.064757\mathbf{p}_4$	0.00
木耳菜（份）	$\mathbf{N}_5 = -0.07849 + 0.36827\mathbf{p}_5$	0.00
圆茄子	$\mathbf{N}_6 = —2.51891 + 0.28113\mathbf{p}_6$	0.00
姜蒜小米椒	$\mathbf{N}_7 = 9.40366 + 0.35221\mathbf{p}_7$	0.00
……	……	……

6.3 基于二次规划模型的补货计划和定价策略

在前两部分基础上，为确定使得商超未来一周的利润最大的销售组合，针对不同单品，建立二次规划模型，以使得商超每日的利润最大为目标函数，从而求解出每种单品的进货数量和定价，和商超每日的最大销售利润。

6.3.1 目标函数的构建

为尽可能地通过不同单品的销售组合，使得商超的销售利润最大，本文以所有单品为研究对象，加总计算所有单品的销售利润，所以，本文构建的目标函数为：

$$\min -\sum_{i=1}^{n} \text{Profit}_i\left(p_i, s_i\right) = N_i\left(p_i\right) \times p_i - c_i \times s_i \tag{10}$$

其中，$Profit_i\left(p_i, s_i\right)$, 即未来一周单品 i 的每日销售利润；$N_i(p_i) = a_i + b_i p_i$，即单品 i 的销售量与价格的关系式；$\mathbf{c}_i$ 为单品 i 的每日进货价格，以前三月各单品不同时间的进货价格的平均数很衡量；$\mathbf{s}_i$ 为单品 i 未来一周的每日进货数量。

6.3.2 约束条件的构建

1、供货量与销售量的关系

由于在每日的销售中，单品 i 的进货数量在经过折损后需满足市场需求，因此存在着供给需求的等式关系，即商超每日的进货数量，必须大于等于每日的销售量。则供货量与销售量的关系式如下：

$$0 \leq N_i\left(p_i\right) \leq s_i \times s_i \tag{11}$$

2、销售价格

由于在销售量与价格的关系式中，可能由于销售量与价格之间的相互作用，即商超也根据历史不断升高的销售量在定价中不断提高价格，部分单品的销售量与价格之间为正线性关系，进而导致不合理，因此，对销售价格设置最高上限，即各单品的销售价格不超过单品所属品类的历史最高价格, 即 MP_i，关系式如下：

$$0 \leq p_i \leq c_i \tag{12}$$

3、利润

为保证商超的利润大于 0，各单品的销售额应大于其进货成本，即：

$$0 \leq p_i \leq MP_i \tag{13}$$

4、非负约束

为有实际意义，进货数量、销售数量和销售价格都应该大于 0。

6.3.3 求解模型

在满足商超的销售需求的情况下，选择各单品不同的销售组合使得商超的利润最大化，结合已知条件和假设，二次规划模型确立为：

$$\min -\sum_{i=1}^{n} \text{Profit}_i\left(p_i, s_i\right) = N_i\left(p_i\right) \times p_i - c_i \times s_i$$

$$\text{s.t.} \begin{cases} 0 \leq N_i\left(p_i\right) \leq s_i \\ 0 \leq p_i \leq MP_i \\ 0 \leq p_i \leq c_i \\ 0 \leq s_i \end{cases} \tag{14}$$

使用 scipy 库中 slsqp 算法求解，得到使得商超利润最大化的单品组合，及相应的补货计划和定价策略，结果如下图所示：

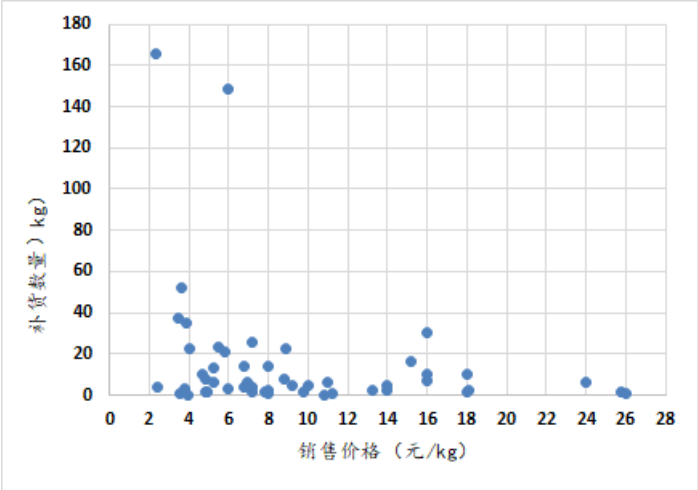

图 11: 各品类的补货计划和定价策略

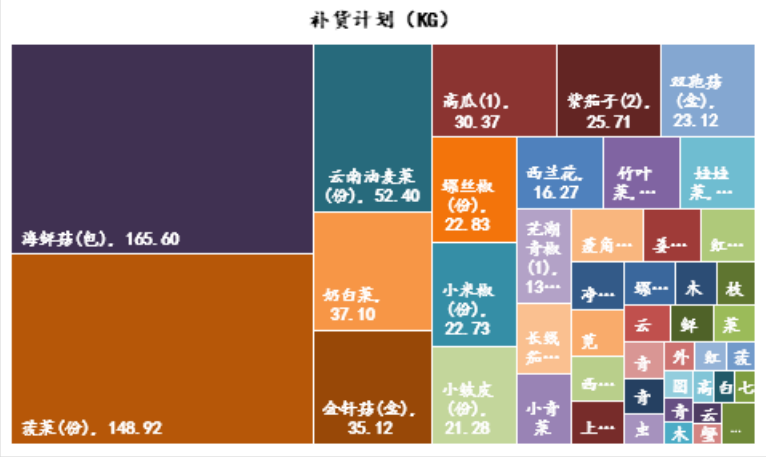

图 12: 各品类的补货计划

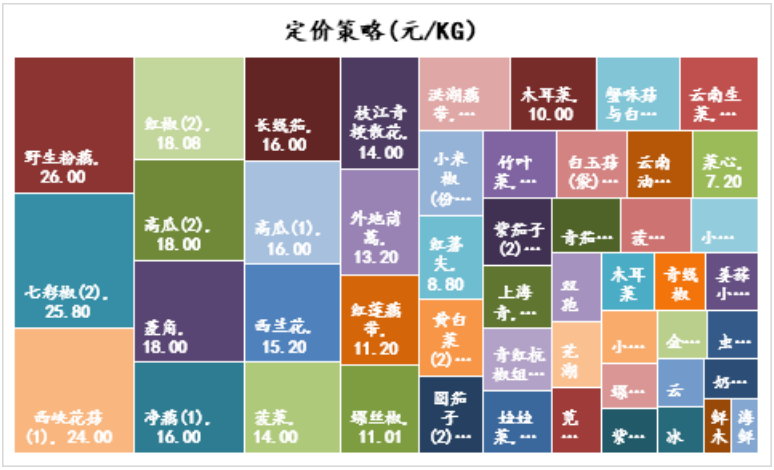

图 13: 各品类的定价策略

6.4 未来一周的最优补货计划和定价策略

最后，根据各单品的补货计划和定价策略，能够得到六大品类的补货计划和定价策略，每日的最大利润可以达到 1686.28 元，补货数量和定价情况的详细数据如下图所示。

花叶类（kg,元/kg)

- 云南生菜 (9.2,4.34)
- 菜心 (7.2,3.92)
- 竹叶菜 (8,14.28)
- 木耳菜（份） (4.9,1.72)
- 黄白菜（2） (8,0.8)
- 奶白菜 (3.49,37.08)
- 木耳菜 (10,4.7)
- 上海青 (6.93,6.04)
- 冰草（盒） (3.5,1.03)
- 云南油麦菜 (7.2,1.56)
- 云南油麦菜（份） (3.63,52.4)
- 菠菜（份） (4.93,72)
- 菠菜 (14,2.19)
- 菜心 (7.2,3.92)
- 娃娃菜 (6.8,14.03)
- 红薯尖 (8.8,7.47)
- 外地茼蒿 (13.2,2.32)
- 小青菜（1） (4.6,10)
- 苋菜 (5.2,6.4)

辣椒类(kg,元/kg)

- 螺丝椒 (11,5.86)
- 芜湖青椒 (5.2,13.4)
- 小米椒(份) (8.9,22.73)
- 小皱皮(份) (5.8,21.27)
- 青线椒(份) (4.8,1.77)
- 螺丝椒(份) (4,22.83)
- 七彩椒(2) (25.8,1.64)
- 姜蒜小米椒组合装(小份) (4.8,7.71)
- 青红杭椒组合装(份) (6.8,3)
- 红椒(2) (18,2.3)

食用菌(kg,元/kg)

- 西峡花菇(1) (24,6)
- 白玉菇(袋) (7.8,1.68)
- 鲜木耳(份) (2.42,4)
- 虫草花(份) (3.8,3.15)
- 双孢菇(盒) (5.5,23.12)
- 蟹味菇与白玉菇双拼(盒) (9.8,1.55)
- 金针菇(盒) (3.8,35.1)
- 海鲜菇(包) (2.6,50)

茄类(kg,元/kg)

- 紫茄子(2) (7.2,25.7)
- 青茄子(1) (6,3.46)
- 长线茄 (16,10.31)
- 紫茄子(1) (3.92,0)
- 圆茄子(2) (8.2.13)

水生根茎类

- 净藕(1) (16,6.78)
- 高瓜(1) (14.4,15.5)
- 菱角 (16,6.8)
- 红莲藕带 (11.2,0.942)
- 野生粉藕 (26,0.76)
- 高瓜(2) (18,1.7)
- 洪湖藕带 (10.8,0)

花菜类(kg,元/kg)

- 西兰花（15.2,16.2）
- 枝江青梗散花（14,14.37）

图 14: 品类的具体补货计划和定价策略

考虑各品类的销售量与成本加成定价之间的关系，从各单品的销售量和成本加成价格关系式出发，将各单品的关系式中的系数根据补货计划中各单品进货数量的占比进行加权平均，得到六大品类的销售量与成本加成定价的关系式，如下表所示。

表 6: 品类的销售量与成本加成定价的关系

品类	拟合方程
花叶类	$\mathbf{N}_1 = 36.421 - 7.6501\mathbf{p}_1$
花菜类	$\mathbf{N}_2 = —9.15 + 1.51\mathbf{p}_2$
水生根茎类	$\mathbf{N}_3 = -59.82 + 4.83\mathbf{p}_3$
茄类	$\mathbf{N}_4 = -29.548 + 6.374\mathbf{p}_4$
辣椒类	$\mathbf{N}_5 = 47.9182 - 7.304\mathbf{p}_5$
食用菌	$\mathbf{N}_6 = 25.1891 - 2.28113\mathbf{p}_6$

7 问题三的模型建立与求解

7.1 混合整数规划模型建立

问题三要求望进一步制定单品的补货计划，要求可售单品总数控制在 27-33 个，且各单品订购量满足最小陈列量 2.5 千克的要求。这相对于在问题二的基础上，增加了两个约束条件：

（1）单品个数限制：进货的单品个数范围为 27-33 个。引入 0-1 变量 h_i，h_i=1 表示商超进货单品 i，则应有：

$$27 \leq \sum_{i}^{n} h_i \leq 33 \tag{15}$$

（2）进货量限制：被进货的单品的进货量应不低于 2.5，即:

$$0 \leq h_i^* \left(s_i - 2.5\right) \tag{16}$$

则该问题的模型表示为：

$$\begin{aligned} \min \quad & -\textstyle\sum_i^n h_i \cdot \mathrm{Profit}_i\left(p_i, s_i\right) \\ \text{其中：} \quad & \mathrm{Profit}_i\left(p_i, s_i\right) = N_i\left(p_i\right) \cdot p_i - c_i \cdot s_i \\ \text{s.t.} \quad & \begin{cases} 0 \leq N_i\left(p_i\right) \leq s_i \\ 0 \leq p_i \leq MP_i \\ 0 \leq s_i \\ 0 \leq h_i^{\cdot}\left(s_i - 2.5\right) \\ 27 \leq \sum_i^n h_i \leq 33 \end{cases} \end{aligned} \tag{17}$$

7.2 模型求解

模型求解可以分为两步，第一步考虑当 h_i 取值给定时，即在进货的单品种类固定时，各单品如何进货和定价能够使得负利润最小。第二步考虑 h_i 的取值不定，即在选择不同的进货单品种类时，对于每一个进货的单品组合，通过步骤一计算其最小的负利润，然后选择所有不同进货单品组合的最小负利润中最小的负利润，即为该优化模型的解。

1、步骤一：当 h_i 取值给定时，优化变量使得目标函数最小

优化目标函数是 $f_i = -h_i \cdot Profit_i(p_i, s_i)$ 之和，由于 h_i 是定值，f_i 只与 p_i,s_i 有关，而 p_i 与 p_j、s_i 与 s_j 之间没有约束关系，可以独立取值而不相互干扰，所以只要分别优化 f_i，得到每个单品的负利润 fi 的最小值后加总。h_i 可能的取值是 0 和 1，我们讨论这两种情况下目标函数的最小值：

（1）对于 h_i=1 的单品，此时这些单品均是商超要进货的单品，考虑这些单品如何进货和数量能够使得负利润最小，此时优化问题转化为一个子问题：

$$\begin{aligned} \min \quad & f_i = -h_i \cdot \text{Profit}_i\left(p_i, s_i\right) \\ \text{s.t.} \quad & \begin{cases} 0 <= N_i\left(p_i\right) <= s_i \\ 0 <= p_i <= MP_i \\ 0 <= s_i - 2.5 \end{cases} \end{aligned}$$

能够发现,这一子问题与第二问的优化模型相比,仅多了单品进货数量的约束,即 $0_i^*\left(s_i - 2.5\right)$。对于第二问中求得的最优解 $\widetilde{p_i}$ 和 $\widetilde{s_i} = N_i\left(\widetilde{p_i}\right)$，若 $\widetilde{s_i} > 2.5$，则已经满足新增的约束条件，因此 $\widetilde{S_l}$ 也是现子问题的最优解；若 $\widetilde{s_i}$<2.5，可以分为两种情况讨论：

a. 由于 $N_i(p_i)$ 为线性函数，f_i 为二次函数，若开口向下，此时 fi 的最小值应该在两端点处取到，由于 $\widetilde{s_i}$>0，因此 s 因为右端点。观察第二问求解的结果能够发现，此时右端点的数值均大于 2.5，因此不可能存在开口向下且 $\widetilde{s_i}$<2.5 的情况。

b. 若 f_i 开口向上，此时 f_i 的最小值应该在对称轴处取到，则此时对称轴 <2.5。若要求 f_i 在 [2.5，右端点] 的区间上取到最小值，此时最优解 =2.5

综上所述，当 h_i=1 时，优化目标函数的解：

$$\hat{s}_i = \begin{cases} \tilde{s}_i \quad \tilde{s}_i > 2.5, h_i = 1 \\ \\ 2.5 \quad \tilde{s}_i \leq 2.5, h_i = 1 \end{cases} \tag{18}$$

（2）对于 $h_i = 0$ 的单品，此时 f_i=0 恒成立，无需优化。

2、步骤二: 找到满足约束条件，且让目标函数可达到的下界最小的 f_i 的取值组合。

若初始时刻 h_i 取值全为 1，则此时使 f_i 的最小值的解应与步骤一中 h_i=1 时的解相同，原始问题的最小值为 $\widetilde{f_i}$；若将某个 h_j 的取值从 1 改为 0，则原始问题的优化函数将会增加

$$\begin{aligned} \left(0 - \widetilde{f_i}\right) &= \widetilde{Profit_i} - \text{loss}_i \\ loss_i &= [2.5 - N_i(\tilde{p}_i)] \times c_i \times 1_{[0,2.5)}(\tilde{p}_i) \end{aligned}$$

若-$\widetilde{f_i}$< 0 时，此时 h_j 变为 0 会减小原始问题的目标优化函数；若-$\widetilde{f_i}$ > 0 时则相反。因此，h_i 满足原始问题的约束的过程可以视为不断将某些 $_j$ 的取值从 1 改为 0，使得每次改变造成的优化函数增大最小，直到满足约束条件。因此，求解原始问题的最优解，只需要：

a. 适用第二问中得到的最优 $\widetilde{p_i}$ 和 $\widetilde{Profit_i}$ 计算-$\widetilde{f_i}$

b. 将-$\widetilde{f_i}$ 从大到小排序

c. 如果大于 0 的-$\widetilde{f_i}$ 个数在 27-33 个之间，则其余的 h_j 取为 0 即满足约束，若个数小于 27，则前 27 个 h_j 取 1，其余的取 0 ，若个数大于 33，则前 33 个 h_j 取 1，其余的取 0。由此得到原始问题的最优解，每日的净利润能够达到 1574.09 元，使得商超利润最大的补货计划和定价策略的具体数据如下：

表 7: 未来一周每日最优的补货计划和定价策略

单品	定价	进货数量	利润	单品	定价	进货数量	利润
小米椒 (份)	8.90	22.73	153.75	海鲜菇 (包)	2.62	50.00	33.29
云南生菜 (份)	4.60	150.00	150.06	木耳菜	10.00	4.71	31.95
西兰花	15.20	16.27	120.03	娃娃菜	6.80	14.04	29.03
长线茄	16.00	10.31	93.00	芜湖青椒 (1)	5.20	13.45	24.42
小皱皮 (份)	5.80	21.28	90.59	七彩椒 (2)	25.80	2.50	11.99
紫茄子 (2)	7.20	25.71	88.76	螺丝椒	11.01	5.87	20.52
金针菇 (盒)	3.85	35.12	84.22	枝江青梗散花	14.00	4.37	19.84
竹叶菜	8.00	14.28	81.15	苋菜	5.20	6.49	18.66
菠菜 (份)	4.93	72.00	60.17	小青菜 (1)	4.67	10.03	18.42
西峡花菇 (1)	24.00	6.08	51.05	姜蒜小米椒	4.80	7.71	18.22
双孢菇 (盒)	5.50	23.12	48.49	螺丝椒 (份)	4.03	22.83	17.20
菱角	16.00	6.81	46.49	上海青	6.94	6.04	17.02
高瓜 (1)	14.42	15.57	44.87	云南生菜	9.20	4.35	15.05
红薯尖	8.80	7.47	41.81	青红杭椒 (份)	6.80	3.83	13.67
云南油麦菜 (份)	3.63	52.41	40.35	红椒 (2)	18.08	2.50	9.49
奶白菜	3.49	37.09	35.72	外地茼蒿	13.20	2.50	9.15
净藕 (1)	16.00	6.78	35.65				

8 问题四的模型建立与求解

8.1 数据收集的分析与意见

题目四要求对有助于制定补货计划和定价策略的相关数据提出建议和意见，影响决策的变量众多，为帮助商超更好地制定补货计划和定价策略，从供应端、销售端、管理端、竞争对手和其他五个方面就数据收集给出有用建议

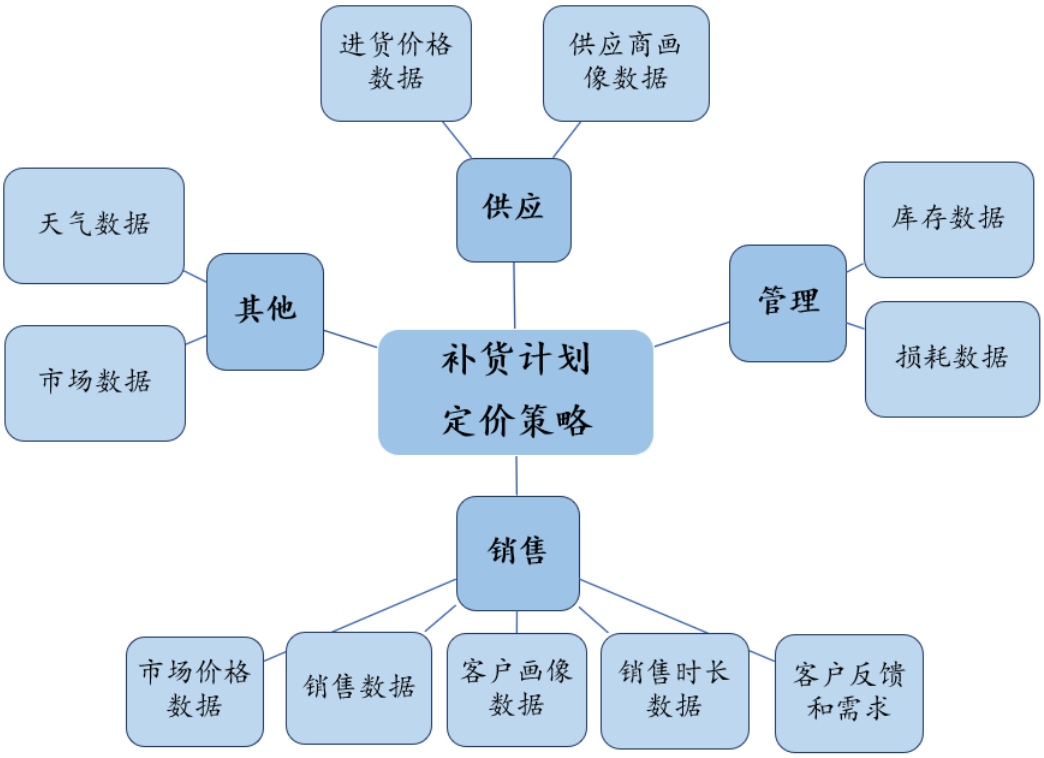

图 15: 补货计划和定价策略有用数据

供应端

1、进价跟踪数据：商超可以持续跟踪蔬菜类商品的进货价格。通过进货价格的记录，商超可以深入了解供应链中的价格变化情况，并进一步分析变动趋势，以便更有效地进行进货决策和成本加成定价。

2、供应商画像数据：商超可以收集供应商相关的信息，包括供货品种、供货质量、供货数量、批发价格、供货时间等数据，对供应商进行画像，全面了解供应商的具体情况和特征，以便商超更好地选择更合适、更具有竞争力的供应商。

销售端

1、销售特征数据：商超可以收集不同时间上蔬菜商品的销售数据，包括不同季节、不同月份、特殊节假日和一天中不同时间的各类蔬菜的销售情况，以了解时间性的销售变化趋势。这有助于商超了解并预测不同时间上的需求变化，并相应地对补货计划和定价策略做出调整。[2]

2、客户画像数据：商超可以收集客户的相关信息，包括购买数量、购买频率、购买种类、距商超远近、购买时间等数据，对客户进行画像，以便商超全面了解客户的购买习惯、偏好、价格敏感度以及忠诚度等情况。在客户画像的基础上，商超也能够针对客户个性化设计补货计划和定价策略，更好地进行销售和利润管理。

3、客户反馈数据：商超可以收集客户的反馈信息，例如购买意向、购买需求、购买要求等数据，以了解消费者对蔬菜商品的需求、喜好等信息，以此及时地调整补货计划和定价策略，更好地满足客户需求，扩大客户群体。

4、销售周期数据：商超可以收集各蔬菜商品的销售时长数据，了解不同商品的销售周期情况，

便于在制定补货计划中结合保鲜期，合理考虑商品的销售时效，避免过度补货或供货不足等情况发生。

管理端

1、库存数据：商超可以持续收集和记录库存数据，包括库存量、库存准确率、库存周转率、库销比等数据，全面准确了解商品的库存情况，以便合理制定和及时调整补货计划，避免过度滞销或缺货等情况，提高库存管理效率，提高库存周转率和利润。

2、损耗率数据：除了附件 4 中提供的近期的商品损耗率数据，商超也可以持续收集和更新损耗率数据，并进一步收集损耗率的变动信息，例如记录不同天气、不同季节、不同供应商下商品损耗率的数据，以便商超评估蔬菜商品的损耗风险，采取有效措施规避风险和减少损耗率，并制定合理的补货计划和定价策略，提高利润率。

竞争端

竞争对手数据：商超可以持续关注并收集竞争对手的数据，包括竞争对手的销售情况、客户分布、定价策略和折扣促销等信息，更好地了解竞争对手情况和市场竞争格局，以便制定差异化的补货计划和定价策略，提高自身的市场竞争力和盈利。

其他方面

1、市场数据：商超可以通过市场调研等手段，收集市场相关数据，包括市场需求变化、饮食观变化、食品安全要求等方面的信息，以便了解全行业发展走向和市场趋势，从而格局市场信息制定和调整补货计划和定价策略，抓住市场机会，谋求发展契机。

2、天气：商超可以收集天气方面的数据，关注了解降水、气温、台风等信息，以便提前了解蔬菜商品的供应情况，更合理地调整补货计划。

8.2 多数据下的模型优化

通过以上分析，蔬菜商品的销售周期、库存积压时间等因素同样对蔬菜的销售量有所影响，同时库存类数据，包括仓库容量、周转率、维护成本等数据，也有助于更好地确定补货计划。若这些数据可得，能够优化第三问的二次规划模型，求解商超最优的补货计划和定价策略。

此时目标函数为：min 销售成本-销售收入 + 折损成本 + 库存维护成本

约束条件包括：1、各单品每日的库存积货量与进货量之和不大于库存总容量

2、各单品每日的库存积货量与进货量之和在减去折损量不大于每日销售量

3、各单品每日的进货量不大于蔬菜供应商的最大供应量

具体的数学表示如下：

$$
\begin{aligned}
&\min -\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{t}\mathrm{N_i}\left(p_{it},d_{ij},\mathrm{j}\right)\cdot p_{it}+\sum_{i=1}^{N}\sum_{t=1}^{T}pc_{it}\cdot s_{it}\\
&+\sum_{i=1}^{N}\sum_{t=1}^{T}sc_{it}\cdot\sum_{j=1}^{t}\left(s_{ij}-\mathrm{N_i}\left(p_{ij},d_{ij},\mathrm{j}\right)\right)\\
&\text{s.t.}\ \sum_{i=1}^{N}\sum_{j=1}^{t}\left(s_{ij}-\mathrm{N_i}\left(p_{i(j-1)},d_{i(j-1)},\mathrm{j}-1\right)\right)<=\mathrm{MS},\mathrm{t}=1,2,\ldots\\
&\sum_{j=1}^{t}\left(s_{ij}-\mathrm{N_i}\left(p_{i(j-1)},d_{i(j-1)},j-1\right)>=\mathrm{N_i}\left(p_{it},d_{it},\mathrm{j}\right),\mathrm{t}=1,2,\ldots\right.\\
&0<=s_{it}<=S_{it},\mathrm{i}=1,2,\ldots\\
&0<=p_{it}<=P_i,\mathrm{i}=1,2,\ldots
\end{aligned}
\tag{19}
$$

其中 p_{it} 为第 i 个单品第 t 天的定价，d_{it} 为第 i 个单品第 t 天的损坏率，s_{it} 为第 i 个单品第 t 天的进货量，sc_{it} 为第 i 个单品储存 t 天的成本，pc_{it} 为第 i 种商品第 t 天的预期购入成本；MS 为库存量,s_{it} 为第 i 种商品第 t 日的预期供应量，P_i 为第 i 种商品的定价上限，N_i (p_{it},d_{ij} , j) 是第 i 种商品，在第 t 日定价为 p_{it}、第 j 日进货、进货时损坏率为 d_{ij} 时的预期需求量，s_{it} 为商品 i 在第 t 天供应商的最大供应量，p_{i0}、$s_{i0}=0$。

9 模型改进方向

9.1 模型的优点

1、问题一分别分析不同品类和单品日销售量和年销售量的分布特征，能够细致刻画商品销售量分别的季节性特征和每日特征。2、通过 DTW 算法计算距离来分析单品之间的相似性，能够考虑到不同单品销售量分布之间的相似特征。3、使用灰色预测来预测未来一周内销量和价格，能够在数据信息不完整的情况下，预测短时间内的销售量和销售价格，同时能够防止先验偏置带来负面影响。4. 问题二从单品维度考虑，通过计算利润最大的销售单品组合，进而得到品类的补货计划和定价策略，能够最大程度地使商超的利润最大化；同时为问题三提供了可用模型，能够较简便地求解答案。

9.2 模型的缺点

1. 销量或价格与其他因素（折扣，折损等）的显式表达式，以来关系不明显。

2. 简化了单个商品的销量对于其他商品的价格的关系。

10 参考文献

参考文献

[1] 徐克. 基于价格分解的鲜活农产品短期价格预测模型. PhD thesis, 中国农业科学院, 2016.

[2] 沈辰 and 穆月英. 我国蔬菜价格的时间序列变动分析. 统计与决策, (16):3, 2011.

[3] 谢乃明. 灰色预测: 思想, 方法与应用. 南京航空航天大学学报: 社会科学版, 24(4):11–18, 2022.

[4] 郝建园. 基于 dtw 算法的非语音类信号研究. 2009.

.ipynb	.py、.R	.png	.png、.pptx	.xlsx、.csv
2 散点图.ipynb	Add_Name.py	2.png	output 加大.png	2_named_.xls
大类热力图.ipynb	DTW.py	3.png	单品密度.png	3_named_.xlsx
单品密度（年）.ipynb	Make_Cost.py	4.png	单品密度年 1.png	2023 _6_price.xlsx
单品密度.ipynb	Make_Sales_624-630.py	5.png	单品密度年 2.png	2023_6_quantity_new
第二类热力图.ipynb	Make_Sales_day.py	6.png	单品热力图.png	Class_Sim.xlsx
聚类.ipynb	Make_Sales_year.py	7.png	第二类热力图.png	Class_years.xlsx
密度.ipynb	optim_2.py	8.png	密度图.png	Type_Sim.xlsx
密度图（年）.ipynb	optim_3.py	9.png	品类密度（年）.png	Type_years.xlsx
热力图.ipynb	R1.R	2 散点图.png	品类热力图.png	Type 聚类.xlsx
		output.png	大类图.pptx	价格预测.xlsx
				数量预测.xlsx
				系数矩阵 (更新).csv

11 附录

11.1 问题 1 代码

11.1.1 .py 脚本 1

```
import pandas as pd
import re
import numpy as np

base = "D:\\MathModel\\"


def Clean_Type(src_file:pd.DataFrame,tgt_file:pd.DataFrame,Type_Code):
    idx = src_file['单品编码'] == int(Type_Code)
    type_file = src_file.loc[idx,["扫码销售时间",'销量(千克)']]
    type_file = type_file.reset_index(drop=True)
    # print(type_file)
    for i in range(type_file.shape[0]):
        if not type_file.empty:
            h = int(re.search('(\d+):.*',
            type_file.loc[i,"扫码销售时间"]).group(1))
            m = int(re.search('\d+:(\d\d).*',
            type_file.loc[i,"扫码销售时间"]).group(1))
            if m >=0 and m <20:
                m = 0
```

```
            if m >= 20 and m <40:
                m = 1
            if m >= 40 and m < 60:
                m = 2
            if type_file.loc[i,'销量(千克)'] >=0:
                tgt_file.loc[h*3+m,Type_Code]
                += type_file.loc[i,'销量(千克)']
def Clean_Type_():
    name_file = pd.read_excel(base+"附件1.xlsx")
    print("file1 load done.")
    print(name_file.shape[0])
    src_file = pd.read_excel(base + "附件2.xlsx")
    print("file2 load done.")
    type_num = 251
    type_columns = []
    for i in range(type_num):
        code = str(name_file.loc[i,"单品编码"])
        type_columns.append(code)
        print(code)
    print("type columns load done.")
    tgt_file = pd.DataFrame
    (np.zeros((72,type_num)),columns=type_columns)
    for i in type_columns:
        Clean_Type(src_file,tgt_file,Type_Code=i)
        print(i+" done.")
    # tgt_file = tgt_file/tgt_file.sum(axis=0)
    tgt_file.to_excel(base+"Type.xlsx",index=False)

# Clean_Type_()

def Clean_Class
    (src_file:pd.DataFrame,tgt_file:pd.DataFrame,Type_Code,Class_Code):
    idx = src_file['单品编码'] == int(Type_Code)
    type_file = src_file.loc[idx,["扫码销售时间",'销量(千克)']]
    type_file = type_file.reset_index(drop=True)
    # print(type_file)
    for i in range(type_file.shape[0]):
        if not type_file.empty:
```

```
            h = int(re.search('(\d+):.*',
            type_file.loc[i,"扫码销售时间"]).group(1))

            m = int(re.search('\d+:(\d\d).*',
            type_file.loc[i,"扫码销售时间"]).group(1))
            if m >=0 and m <20:
                m = 0
            if m >= 20 and m <40:
                m = 1
            if m >= 40 and m < 60:
                m = 2
            if type_file.loc[i,'销量(千克)'] >=0:
                tgt_file.loc[h*3+m,Class_Code]
                += type_file.loc[i,'销量(千克)']


def Clean_Class_():
    name_file = pd.read_excel(base+"附件1.xlsx")
    print("file1 load done.")
    src_file = pd.read_excel(base + "附件2.xlsx")
    print("file2 load done.")
    type_num = 251
    class_num = 6
    type_columns = []
    for i in range(type_num):
        code = (str(name_file.loc[i,"单品编码"]),
        str(name_file.loc[i,"分类编码"]))
        type_columns.append(code)
        print(code)
    print("type columns load done.")
    tgt_file = pd.DataFrame
    (np.zeros((72,class_num)),
    columns=['1011010101','1011010402','1011010201',
    '1011010501','1011010504','1011010801'])
    for type_code,class_code in type_columns:
        Clean_Class(src_file,tgt_file,
        Type_Code=type_code,Class_Code=class_code)
        print(type_code," done.")
    tgt_file.to_excel(base+"Class.xlsx",index=False)
```

```
Clean_Class_()

def Standization(name,alpha=0.001):
    df = pd.read_excel(base+name+".xlsx")
    df = df + alpha
    df = df/df.sum(axis=0)
    df.to_excel(base+name+"_std.xlsx",index=False)
```

11.1.2 .py 脚本 2

```
import pandas as pd
import numpy as np
import re

base = "D:\\MathModel\\"

def date_transform(m:int,d:int):
    n = d
    if m == 1:
        n += 0
    if m == 2:
        n += 31
    if m == 3:
        n += 59
    if m == 4:
        n += 90
    if m == 5:
        n +=120
    if m == 6:
        n += 151
    if m == 7:
        n += 181
    if m == 8:
        n += 212
    if m == 9:
        n += 242
    if m == 10:
        n += 273
    if m == 11:
```

```
        n += 303
    if m == 12:
        n += 334
    return n - 1

def Clean(src_file:pd.DataFrame,
Type_file:pd.DataFrame,Class_file:pd.DataFrame,Type_Code,Class_Code):
    type_file = src_file[src_file['单品编码'] == int(Type_Code)]
    type_file = type_file.reset_index(drop=True)
    if not type_file.empty:
        for i in range(type_file.shape[0]):
            if type_file.loc[i,'销量(千克)'] >=0:
                date = type_file.loc[i, "销售日期"].strftime("%Y/%m/%d")
                m = int(re.search('\d+/(\d+)/.*', date).group(1))
                d = int(re.search('\d+/\d+/(\d+).*',date).group(1))

                Type_file.loc[date_transform(m,d),Type_Code]
                += type_file.loc[i,'销量(千克)']
                Class_file.loc[date_transform(m,d),Class_Code]
                += type_file.loc[i, '销量(千克)']

def Clean_():
    name_file = pd.read_excel(base+"附件1.xlsx")
    print("file1 load done.")
    print(name_file.shape[0])
    src_file = pd.read_excel(base + "附件2.xlsx")
    print("file2 load done.")
    type_num = 251
    type_columns = []
    type_tuples = []
    for i in range(type_num):
        code = (str(name_file.loc[i,"单品编码"]),
        str(name_file.loc[i,"分类编码"]))
        type_tuples.append(code)
        type_columns.append(str(name_file.loc[i,"单品编码"]))
    Type_file = pd.DataFrame(np.zeros((365,
    len(type_columns))),columns=type_columns)
    Class_file = pd.DataFrame(np.zeros((365,6)),
                columns=['1011010101',
```

```
                '1011010402', '1011010201', '1011010501', '1011010504',
                '1011010801'])
    for type_code,class_code in type_tuples:
        Clean(src_file,Type_file=Type_file,Type_Code=type_code,
        Class_Code=class_code,Class_file=Class_file)
        print(type_code+"␣done.")
    # tgt_file = tgt_file/tgt_file.sum(axis=0)
    Type_file.to_excel(base+"Type_years.xlsx",index=False)
    Class_file.to_excel(base + "Class_years.xlsx", index=False)

Clean_()
```

11.1.3 .py 脚本 3

```
import pandas as pd
import numpy as np
import re

base = "D:\\MathModel\\"

def date_transform(m:int,d:int):
    n = d
    if m == 1:
        n += 0
    if m == 2:
        n += 31
    if m == 3:
        n += 59
    if m == 4:
        n += 90
    if m == 5:
        n +=120
    if m == 6:
        n += 151
    if m == 7:
        n += 181
    if m == 8:
        n += 212
    if m == 9:
        n += 242
```

```
    if m == 10:
        n += 273
    if m == 11:
        n += 303
    if m == 12:
        n += 334
    return n - 1

def Clean(src_file:pd.DataFrame,
Type_file:pd.DataFrame,Class_file:pd.DataFrame,Type_Code,Class_Code):
    type_file = src_file[src_file['单品编码'] == int(Type_Code)]
    type_file = type_file.reset_index(drop=True)
    if not type_file.empty:
        for i in range(type_file.shape[0]):
            if type_file.loc[i,'销量(千克)'] >=0:
                date = type_file.loc[i, "销售日期"].strftime("%Y/%m/%d")
                m = int(re.search('\d+/(\d+)/.*', date).group(1))
                d = int(re.search('\d+/\d+/(\d+).*',date).group(1))

                Type_file.loc[date_transform(m,d),Type_Code]
                += type_file.loc[i,'销量(千克)']
                Class_file.loc[date_transform(m,d),Class_Code]
                += type_file.loc[i, '销量(千克)']

def Clean_():
    name_file = pd.read_excel(base+"附件1.xlsx")
    print("file1 load done.")
    print(name_file.shape[0])
    src_file = pd.read_excel(base + "附件2.xlsx")
    print("file2 load done.")
    type_num = 251
    type_columns = []
    type_tuples = []
    for i in range(type_num):
        code = (str(name_file.loc[i,"单品编码"]),
        str(name_file.loc[i,"分类编码"]))
        type_tuples.append(code)
        type_columns.append(str(name_file.loc[i,"单品编码"]))
    Type_file = pd.DataFrame
```

```
    (np.zeros((365,len(type_columns))),columns=type_columns)
    Class_file = pd.DataFrame(np.zeros((365,6)),
    columns=['1011010101', '1011010402', '1011010201',
    '1011010501', '1011010504','1011010801'])
    for type_code,class_code in type_tuples:
        Clean(src_file,Type_file=Type_file,
        Type_Code=type_code,Class_Code=class_code,Class_file=Class_file)
        print(type_code+"␣done.")
    # tgt_file = tgt_file/tgt_file.sum(axis=0)
    Type_file.to_excel(base+"Type_years.xlsx",index=False)
    Class_file.to_excel(base + "Class_years.xlsx", index=False)

Clean_()
```

11.1.4 .ipynb 脚本 1

```
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
df1=pd.read_excel(r"C:\Users\1\Desktop\Type_years.xlsx",sheet_name='Sheet2')
df2=pd.read_excel(r"C:\Users\1\Desktop\Type_years.xlsx",sheet_name='Sheet3')

df1
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(10,4))
plt.xlabel('时间(每1单位为一天)')
plt.ylabel('销售量(单位:kg)')
p1=sns.lineplot(data=df1,y=df1['牛首油菜']  ,x=df1['时间'], color="r")
p1=sns.lineplot(data=df1,y=df1['保康高山大白菜'],x=df1['时间'],color="b")
p1=sns.lineplot(data=df1,y=df1['小米椒'], x=df1['时间'],  color="black")
p1=sns.lineplot(data=df1,y=df1['西峡香菇(1)'], x=df1['时间'],  color="green")
#p1=sns.lineplot(data=df1,y=df1['金针菇(盒)'],x=df1['时间'],   color="grey")

plt.legend(['牛首油菜','保康高山大白菜','小米椒','西峡香菇(1)'])

plt.show()
df2
rc = {'font.sans-serif': 'SimHei',
```

```
        'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(10,4))
plt.xlabel('时间(每1单位为一天)')
plt.ylabel('销售量(单位:kg)')
p1=sns.lineplot(data=df2,y=df2['竹叶菜']   ,x=df2['时间'], color="r")
p1=sns.lineplot(data=df2,y=df2['黄心菜(2)'],x=df2['时间'],color="b")
#p1=sns.lineplot(data=df2,y=df2['螺丝椒'], x=df2['时间'],  color="black")
p1=sns.lineplot(data=df2,y=df2['金针菇(1)'], x=df2['时间'],  color="green")
p1=sns.lineplot(data=df2,y=df2['海鲜菇(袋)(3)'],x=df2['时间'],    color="grey")

plt.legend(['竹叶菜','黄心菜(2)','金针菇(1)','海鲜菇(袋)(3)'])

plt.show()
```

11.1.5 .ipynb 脚本 2

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_excel(r"C:\Users\1\Desktop\Class_Sim.xlsx")

df
dfn=df.rename(index={0:'花叶类',1:'水生根茎类',2:'花菜类',3:'茄类',4:'辣椒类',5:'食
dfnsns.set(font_scale=1)
rc = {'font.sans-serif': 'SimHei',
        'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(figsize = (20,8))
sns.heatmap(dfn,square=True,fmt='f',annot=True,linewidths=0.3,cmap='Greens',vmin=0.0
            ,vmax=0.00068)
```

11.1.6 .ipynb 脚本 3

```
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
df=pd.read_excel(r"C:\Users\1\Desktop\Type.xlsx")
print(df)
```

```
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p1=sns.lineplot(data=df,y=df['牛首生菜']  ,x=df['时间'], color="r")
p1=sns.lineplot(data=df,y=df['红尖椒'],x=df['时间'],color="b")
p1=sns.lineplot(data=df,y=df['随州泡泡青'], x=df['时间'],  color="black")
p1=sns.lineplot(data=df,y=df['马齿苋'], x=df['时间'],  color="green")
p1=sns.lineplot(data=df,y=df['海鲜菇'],x=df['时间'],   color="grey")

plt.legend(['牛首生菜','红尖椒','随州泡泡青','马齿苋','海鲜菇'])

plt.show()
df2=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet2')
df3=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet3')
df4=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet4')
df5=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet5')
df6=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet6')
df2
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p2=sns.lineplot(data=df2,y=df2['上海青']  ,x=df2['时间'], color="r")
p2=sns.lineplot(data=df2,y=df2['红橡叶'],x=df2['时间'],color="b")
p2=sns.lineplot(data=df2,y=df2['青线椒'], x=df2['时间'],  color="black")
p2=sns.lineplot(data=df2,y=df2['黑牛肝菌'], x=df2['时间'],  color="green")
p2=sns.lineplot(data=df2,y=df2['蟹味菇'],x=df2['时间'],   color="grey")

plt.legend(['上海青','红橡叶','青线椒','黑牛肝菌','蟹味菇'])

plt.show()
df3
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
```

```
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p3=sns.lineplot(data=df3,y=df3['萝卜叶']  ,x=df3['时间'], color="r")
p3=sns.lineplot(data=df3,y=df3['绿牛油'],x=df3['时间'],color="b")
p3=sns.lineplot(data=df3,y=df3['红尖椒'], x=df3['时间'],color="black")
p3=sns.lineplot(data=df3,y=df3['黑皮鸡枞菌'], x=df3['时间'],color="green")
p3=sns.lineplot(data=df3,y=df3['白玉菇'],x=df3['时间'],color="grey")

plt.legend(['萝卜叶','绿牛油','红尖椒','黑皮鸡枞菌','白玉菇'])

plt.show()
df4
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p4=sns.lineplot(data=df4,y=df4['牛首油菜']  ,x=df4['时间'], color="r")
p4=sns.lineplot(data=df4,y=df4['保康高山大白菜'],x=df4['时间'],color="b")
p4=sns.lineplot(data=df4,y=df4['螺丝椒'], x=df4['时间'],color="black")
p4=sns.lineplot(data=df4,y=df4['海鲜菇'], x=df4['时间'],color="green")
p4=sns.lineplot(data=df4,y=df4['蟹味菇'],x=df4['时间'],color="grey")

plt.legend(['牛首油菜','保康高山大白菜','螺丝椒','海鲜菇','蟹味菇'])

plt.show()
df5
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p5=sns.lineplot(data=df5,y=df5['茼蒿']  ,x=df5['时间'], color="r")
p5=sns.lineplot(data=df5,y=df5['黄心菜'],x=df5['时间'],color="b")
p5=sns.lineplot(data=df5,y=df5['七彩椒'], x=df5['时间'],color="black")
```

```
p5=sns.lineplot(data=df5,y=df5['鲜木耳'], x=df5['时间'],color="green")
p5=sns.lineplot(data=df5,y=df5['海鲜菇（袋）'],x=df5['时间'],color="grey")

plt.legend(['茼蒿','黄心菜','七彩椒','鲜木耳','海鲜菇（袋）'])

plt.show()
df6
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p6=sns.lineplot(data=df6,y=df6['蔡甸藜蒿']  ,x=df6['时间'], color="r")
p6=sns.lineplot(data=df6,y=df6['苋菜'],x=df6['时间'],color="b")
p6=sns.lineplot(data=df6,y=df6['灯笼椒'], x=df6['时间'],color="black")
p6=sns.lineplot(data=df6,y=df6['杏鲍菇'], x=df6['时间'],color="green")
p6=sns.lineplot(data=df6,y=df6['海鲜菇(包)'],x=df6['时间'],color="grey")

plt.legend(['蔡甸藜蒿','苋菜','灯笼椒','杏鲍菇','海鲜菇(包)'])

plt.show()
df7=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet7')
df8=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet8')
df9=pd.read_excel(r"C:\Users\1\Desktop\Typezong.xlsx",sheet_name='Sheet9')
df7
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p7=sns.lineplot(data=df7,y=df7['云南生菜']  ,x=df7['时间'], color="r")
p7=sns.lineplot(data=df7,y=df7['菊花油菜'],x=df7['时间'],color="b")
p7=sns.lineplot(data=df7,y=df7['小白菜'], x=df7['时间'],color="black")
p7=sns.lineplot(data=df7,y=df7['长线茄'], x=df7['时间'],color="green")
p7=sns.lineplot(data=df7,y=df7['红灯笼椒'],x=df7['时间'],color="grey")

plt.legend(['云南生菜','菊花油菜','小白菜','长线茄','红灯笼椒'])
```

```
plt.show()
df8
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p8=sns.lineplot(data=df8,y=df8['竹叶菜']  ,x=df8['时间'], color="r")
p8=sns.lineplot(data=df8,y=df8['青红杭椒组合装'],x=df8['时间'],color="b")
p8=sns.lineplot(data=df8,y=df8['小青菜'], x=df8['时间'],color="black")
p8=sns.lineplot(data=df8,y=df8['青茄子'], x=df8['时间'],color="green")
p8=sns.lineplot(data=df8,y=df8['姜蒜小米椒组合装(小份)'],x=df8['时间'],color="grey")

plt.legend(['竹叶菜','青红杭椒组合装','小青菜','青茄子','姜蒜小米椒组合装(小份)'])

plt.show()
df9
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p9=sns.lineplot(data=df9,y=df9['小白菜']  ,x=df9['时间'], color="r")
p9=sns.lineplot(data=df9,y=df9['红椒'],x=df9['时间'],color="b")
p9=sns.lineplot(data=df9,y=df9['奶白菜'], x=df9['时间'],color="black")
p9=sns.lineplot(data=df9,y=df9['紫茄子'], x=df9['时间'],color="green")
p9=sns.lineplot(data=df9,y=df9['牛排菇'],x=df9['时间'],color="grey")

plt.legend(['小白菜','红椒','奶白菜','紫茄子','牛排菇'])

plt.show()
```

11.1.7 .ipynb 脚本 4

```
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
```

```
df=pd.read_excel(r"C:\Users\1\Desktop\Class.xlsx",sheet_name='Sheet1')
print(df)
##p1=sns.kdeplot(df['花菜类'], shade=True, color="b")
#p1=sns.kdeplot(df['水生根茎类'], shade=True, color="olive")
#p1=sns.kdeplot(df['茄类'], shade=True, color="m")
#p1=sns.kdeplot(df['辣椒类'], shade=True, color="grey")
#p1=sns.kdeplot(df['食用菌'], shade=True, color="pink")
#plt.legend(loc='best')
#plt.show()
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(10,4))
plt.xlabel('时间(每1单位为20分钟)')
plt.ylabel('销售量(单位:kg)')
p1=sns.lineplot(data=df,y=df['花叶类']  ,x=df['时间'], color="r")
p1=sns.lineplot(data=df,y=df['花菜类'],x=df['时间'],color="b")
p1=sns.lineplot(data=df,y=df['水生根茎类'], x=df['时间'],  color="olive")
p1=sns.lineplot(data=df,y=df['茄类'], x=df['时间'],  color="m")
p1=sns.lineplot(data=df,y=df['辣椒类'],x=df['时间'],   color="grey")
p1=sns.lineplot(data=df,y=df['食用菌'],x=df['时间'],  color="pink")
plt.legend(['花叶类','花菜类','水生根茎类','茄类','辣椒类','食用菌'])

plt.show()
```

11.1.8 .ipynb 脚本 5

```
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
df=pd.read_excel(r"C:\Users\1\Desktop\Class_years.xlsx")
print(df)
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(10,4))
plt.xlabel('时间(每1单位为一天)')
plt.ylabel('销售量(单位:kg)')
p1=sns.lineplot(data=df,y=df['花叶类']  ,x=df['时间'], color="r")
p1=sns.lineplot(data=df,y=df['花菜类'],x=df['时间'],color="b")
```

```
p1=sns.lineplot(data=df,y=df['水生根茎类'], x=df['时间'],  color="olive")
p1=sns.lineplot(data=df,y=df['茄类'], x=df['时间'],  color="m")
p1=sns.lineplot(data=df,y=df['辣椒类'],x=df['时间'],   color="grey")
p1=sns.lineplot(data=df,y=df['食用菌'],x=df['时间'],  color="pink")
plt.legend(['花叶类','花菜类','水生根茎类','茄类','辣椒类','食用菌'])

plt.show()
```

11.1.9 .ipynb 脚本 6

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_excel(r"C:\Users\1\Desktop\Type_Sim.xlsx")
sns.set(font_scale=1)
dfn=df.rename(index={0:'牛首生菜',1:'四川红香椿',2:'本地小毛白菜',3:'白菜苔',4:'苋菜
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(figsize = (20,8))
sns.heatmap(dfn,square=True,annot=True,linewidths=0.3,cmap='Greens',vmin=0
            ,vmax=0.042)
```

11.1.10 .ipynb 脚本 7

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import numpy as np
  #轮廓系数 和 不同商品的分类标签
df=pd.read_excel(r"C:\Users\1\Desktop\Type聚类.xlsx")
# 将DataFrame转换成numpy数组
data = df.values[:,1:-1]


# 对特征进行标准化处理
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(data)

# 选择要用于聚类的特征列
features = X_scaled  # 从第二列开始选择所有列
matrix = np.zeros((20, 2))

for i in range(2,21):

    # 定义聚类算法和聚类数量
    kmeans = KMeans(n_clusters=i)  # 假设要将商品分为3个类别

    # 训练聚类模型
    kmeans.fit(features)



    # 计算轮廓系数
    score = silhouette_score(features, kmeans.labels_, metric='euclidean')

    matrix[i-2,0]=i
    matrix[i-2,1]=score
scorescof = pd.DataFrame(matrix)


scorescof.to_excel("轮廓系数与聚类个数.xlsx", index=False, header=False)
print(matrix)


cluster = KMeans(n_clusters=2)
cluster.fit(features)
labels=pd.DataFrame(cluster.labels_)
labels.to_excel('商品分类.xlsx')

center=pd.DataFrame(cluster.cluster_centers_)
center.to_excel('聚类中心点坐标.xlsx')
```

11.1.11 .ipynb 脚本 8

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt
df=pd.read_excel(r"C:\Users\1\Desktop\Type_Sim全.xlsx")

df
dfn=df.rename(index={0:'云南生菜',1:'大白菜',2:'云南油麦菜',3:'娃娃菜',4:'云南油麦菜
                     ,6:'净藕（1）',7:'紫茄子（2）',8:'泡泡椒（精品）',9:'芜湖青椒（
                     ,11:'螺丝椒（份）',12:'西峡香菇（1）',13:'金针菇（盒）'})
dfn
sns.set(font_scale=1.5)
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(figsize = (25,8))
sns.heatmap(dfn,square=True,fmt='f',annot=True,linewidths=0.3,cmap='Greens',vmin=0.0
            ,vmax=0.005656       ,annot_kws={"fontsize":7})
```

11.2 问题 2 代码

11.2.1 .py 脚本 4

```
from cvxopt import matrix,solvers
import numpy as np
import pandas as pd
from scipy.optimize import minimize

base = "D:\\MathModel\\"

def Make_Cost(src_file:pd.DataFrame,strat_idx,end_idx,columns):
    src_file_ = pd.DataFrame(data=None, columns=src_file.columns)
    for j in range(len(strat_idx)):
        src_file_ = src_file_.append
        (src_file[strat_idx[j]:end_idx[j]].reset_index(drop=True),
        ignore_index=True
    tgt_file = pd.DataFrame(np.zeros(columns.shape),index=columns,
    columns=['批发价格(元/千克)'])
    for i in columns:
        tgt_file.loc[i,'批发价格(元/千克)'] =
        src_file_.loc[src_file_["单品编码"]==int(i),'批发价格(元/千克)'].mean()

    tgt_file.to_excel(base+"R\\可用商品进价1.xlsx",index=False)
```

```
def Make_Cost2(src_file:pd.DataFrame,columns):
    tgt_file = pd.DataFrame(np.zeros(columns.shape),
    index=columns,columns=['批发价格(元/千克)'])
    for i in columns:
        tgt_file.loc[i,'批发价格(元/千克)'] =
        src_file.loc[src_file[src_file.columns[0]] == int(i),'批发价格(元/千克)'].m

    tgt_file.to_excel(base+"R\\可用商品均进价.xlsx",index=False)

# src_file = pd.read_excel(base+"附件3.xlsx")
# name_file = pd.read_excel(base+"R\\price_dense.xlsx")
# columns = name_file.columns[1:]
# Make_Cost(src_file=src_file,strat_idx=[0,15191,35969,54473],
end_idx=[1039,18265,39429,55982],columns=columns)

# src_file = pd.read_excel(base+"可用商品均进价_前一周.xlsx")
# name = pd.read_excel(base+"R\\2023_6_price.xlsx").columns
# Make_Cost2(src_file,name[1:])

# def Solve():
#     params = pd.read_csv(base+"系数矩阵.csv",encoding='gbk')
#     costs = pd.read_excel(base+"R\\可用商品进价.xlsx")
#     print("load done.")
#     P = matrix(0.,(63,63))
#     k = params['价格系数'].to_numpy()
#     b = params['截距'].to_numpy()
#     c = costs['批发价格(元/千克)'].to_numpy()
#     P[::64] = -2*k
#     q=matrix(c-b)
#     A=np.vstack((np.diag(-1*k),np.diag(-1*np.ones(63))))
#     # print(A.shape)
#     A=matrix(A)
#     b=matrix(np.hstack((b,np.zeros(63))))
#     print("establish done.")
#
#     s=solvers.qp(P,q,A,b,None,None)
#     print("最优解:",s['x'])
#     print("最优值:",s['primal objective'])
```

```
def Solve():
    params = pd.read_csv(base+"R\\系数矩阵(更新).csv",encoding='gbk')
    costs = pd.read_excel(base+"R\\可用商品均进价.xlsx")
    price = pd.read_excel(base+"R\\2023_6_price.xlsx")
    ub = price[price.columns[1:]].max(axis=0).to_numpy()
    print("load done.")
    k = params['系数'].to_numpy()
    b = params['截距'].to_numpy()
    c = costs['批发价格(元/千克)'].to_numpy()
    f = lambda x: np.dot(-1*k,x**2)+np.dot(c*k-b,x)+np.dot(c,b)
    quantity = pd.read_excel(base+"R\\2023_6_quantity_new.xlsx")
    ub2 = quantity[quantity.columns[1:]].max(axis=0).to_numpy()



    cons=({'type':'ineq','fun':lambda x:k*x+b},
    {'type':'ineq','fun':lambda x:2*ub2-k*x-b})



    bound = list(zip(np.zeros(ub.shape[0]),ub))

    print("establish done.")
    res = minimize(f,np.ones(k.shape[0])*8.5,
    constraints=cons,bounds=bound)



    print("最优解:",res)
    return -1*res.fun,res.x,k*res.x+b

code = pd.read_excel(base+"R\\2023_6_quantity_new.xlsx")
profit,price,expect_sale = Solve()
df = pd.DataFrame({"price":price,
"expect_sale":expect_sale,"profit":profit})
df.set_index(code.columns[1:],inplace=True)
df.to_excel(base+"R\\2___.xlsx")
```

11.2.2 .py 脚本 5

```
import pandas as pd
import numpy as np
import pickle

base = "D:\\MathModel\\"

"""323139 374056"""
"""575205 618701"""
def Clean(src_file:pd.DataFrame,
price_file:pd.DataFrame,quantity_file:pd.DataFrame,Type_Code,date,start_idx,end_idx)

    type_file = src_file[start_idx:end_idx]
    type_file = type_file[type_file['单品编码'] ==
    int(Type_Code)].reset_index(drop=True)
    type_file = type_file[type_file['销售类型'] ==
    "销售"].reset_index(drop=True)
    type_file = type_file[type_file['销售日期'] ==
    date].reset_index(drop=True)
    # print(type_file)
    if not type_file.empty:
        price_file.loc[date,Type_Code] =
        type_file['销售单价(元/千克)'].mean()
        quantity_file.loc[date,Type_Code] =
        type_file['销量(千克)'].sum()



def Clean_1(goods_name):
    src_file = pd.read_excel(base + "附件2.xlsx")
    print("file1 load done.")
    type_columns = []
    for i in goods_name:
        type_columns.append(str(i))

    type_num = len(type_columns)
    print("type columns load done.Num type:" + str(type_num))
    year = "2023"
    date_idx = [year+"/6/"+str(i) for i in range(1, 31)]
    price_file = pd.DataFrame(np.zeros((30, type_num)),
```

```
    index=date_idx, columns=type_columns)
    quantity_file = pd.DataFrame(np.zeros((30, type_num)),
    index=date_idx, columns=type_columns)
    for j in date_idx:
        for i in range(len(type_columns)):
            Clean(src_file=src_file,
                  price_file=price_file,
                  quantity_file=quantity_file,
                  Type_Code=type_columns[i],
                  date=j,
                  start_idx=860679,
                  end_idx=878503)
        print(j+" done.")
    # tgt_file = tgt_file/tgt_file.sum(axis=0)
    price_file.to_excel(base+"R\\"+year+"_6_price.xlsx")
    quantity_file.to_excel(base+"R\\"+year+"_6_quantity.xlsx")

def Clean_2():
    type_file = pd.read_excel(base + "R\\2022_rate.xlsx")
    print("file2 load done.")
    type_columns = type_file.columns
    for i in type_columns[1:]:
        mean = type_file.loc[type_file[i] != 0, i].mean()
        type_file.loc[type_file[i] == 0, i] = mean

    type_file.to_excel(base+"R\\2022_rate_dense.xlsx",
    index=False)

# def Clean_3():
#     src_file = pd.read_excel(base + "附件6.xlsx")
#     print("file1 load done.")
#     type_file = pd.read_excel(base + "R\\2022_quantity_dense.xlsx")
#     print("file2 load done.")
#     type_columns = type_file.columns.drop(['Unnamed: 0'])
#
#
#
#     type_num = len(type_columns)
#     for year in ['2021','2022']:
```

```
#         date_idx =
#            [year+"/"+date_transform(i) for i in range(1, 62)]
#         damage_file = pd.DataFrame(np.zeros((61, type_num)),
#           index=date_idx, columns=type_columns)
#         for j in date_idx:
#             for i in range(len(type_columns)):
#                 # print(src_file.loc[2,src_file.columns[0]])
#                 idx = src_file[src_file.columns[0]] == int(type_columns[i])
#                 src_file_ = src_file[idx].reset_index(drop=True)
#                 # print(type_file)
#                 if not src_file_.empty:
#                     rate = src_file_[src_file.columns[2]].mean()
#                     damage_file.loc[j,type_columns[i]] = rate
#                     # print(rate)
#
#             print(j+" done.")
#             # tgt_file = tgt_file/tgt_file.sum(axis=0)
#         damage_file.to_excel(base+"R\\"+year+"_damage.xlsx")

def Clean_4():
    type_file = pd.read_excel(base + "R\\2022_rate_dense.xlsx")
    print("file2 load done.")
    type_columns = type_file.columns

    tgt_file_1 = pd.read_excel(base + "R\\2022_cost.xlsx")
    tgt_file_1 = tgt_file_1[type_columns]
    tgt_file_2 = pd.read_excel(base + "R\\2022_quantity.xlsx")
    tgt_file_2 = tgt_file_2[type_columns]
    for i in type_columns[1:]:
        mean1 = tgt_file_1.loc[tgt_file_1[i] != 0, i].mean()
        tgt_file_1.loc[tgt_file_1[i] == 0, i] = mean1
        mean2 = tgt_file_2.loc[tgt_file_2[i] != 0, i].mean()
        tgt_file_2.loc[tgt_file_2[i] == 0, i] = mean2

    tgt_file_1.to_excel(base + "R\\2022_cost_dense.xlsx", index=False)
    tgt_file_2.to_excel(base+"R\\2022_quantity_dense.xlsx",index=False)
with open(base + "可用商品名称", "rb") as file:
    goods_name = pickle.load(file)
```

11.2.3 .py 脚本 6

```
import pandas as pd
import numpy as np
import pickle

base = "D:\\MathModel\\"

def Make_Cost1(src_file:pd.DataFrame,strat_idx,end_idx):
    goods = set()
    tgt_file = pd.DataFrame(data=None,
    columns=["单品编码", "批发价格(元/千克)"])
    df2 = src_file[strat_idx:end_idx].reset_index(drop=True)
    tgt_file = tgt_file.append(df2,ignore_index=True)
    for i in range(df2.shape[0]):
        goods.add(df2.loc[i,"单品编码"])
    pickle.dump(goods, open(base + "可用商品名称", "wb"))
    tgt_file.to_excel(base+"可用商品日进价_前一周.xlsx",index=False)

def Make_Cost2(src_file:pd.DataFrame,columns):
    tgt_file = pd.DataFrame(np.zeros(len(columns)),
    index=columns,columns=['批发价格(元/千克)'])
    for i in columns:
        tgt_file.loc[i,'批发价格(元/千克)'] =
        src_file.loc[src_file["单品编码"]==
        i,'批发价格(元/千克)'].mean()

    tgt_file.to_excel(base+"可用商品均进价_前一周.xlsx")

# src_file = pd.read_excel(base+"附件3.xlsx")
# Make_Cost1(src_file,strat_idx=55628,end_idx=55982)
# with open(base+"可用商品名称","rb") as file:
#     names = pickle.load(file)
#
# columns = []
# for i in names:
#     columns.append(i)
# src_file = pd.read_excel(base+"可用商品日进价_前一周.xlsx")
```

```
# Make_Cost2(src_file,columns)
```

11.2.4 .ipynb 脚本 9

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pylab as plt
df=pd.read_excel(r"C:\Users\1\Desktop\2_named.xlsx")
df
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', style='ticks',rc=rc)
plt.figure(dpi=300,figsize=(6,4))
plt.xlabel('价格')
plt.ylabel('预计销售量(单位:kg)')
plt.xticks(np.arange(0, 50, 3))
p1=sns.scatterplot(data=df,y=df['期望销量']  ,x=df['价格'], color="blue")
plt.show()
```

11.2.5 .R 脚本 1

```
library(readxl)

quan23 <- read_excel("C:/Users/1/Desktop/数量预测.xlsx")
price23 <- read_excel("C:/Users/1/Desktop/价格预测.xlsx")

price<-as.matrix(price23)
quan<- as.matrix(quan23 )



ncol(price)
price[,i]
matrc<-c(5,5,5)
for(i in 1:51){
  p<-as.vector(price[,i])
  q<-as.vector(quan[,i])

  p<-as.numeric(p)
  q<-as.numeric(q)
```

```
  fit<-lm(q~1+p)
  cof<-fit$coefficients

  pVal <- anova(fit)$'Pr(>F)'[1]
  matrc=rbind(matrc,c(cof,pVal))
  print(c(cof,pVal))
  #print(summary(fit))
  #print(-(fit$coefficients[3]/2*fit$coefficients[2]))


}
matrc
dim(matrc)


write.csv(matrc,'系数矩阵(更新).csv')
```

11.3 问题 3 代码

11.3.1 .py 脚本 7

```
import numpy as np
import pandas as pd

from scipy.optimize import minimize

base = "D:\\MathModel\\"

# def Make_Quantity(src_file:pd.DataFrame,columns):
#     src_file = src_file[columns]
#     src_file.to_excel(base+"R\\2023_6_quantity_new.xlsx")
#
# src_file = pd.read_excel(base+"R\\2023_6_quantity.xlsx")
# name = pd.read_excel(base+"R\\2023_6_price.xlsx").columns
# Make_Quantity(src_file,name[1:])


# def Solve(num_x):
#     params = pd.read_csv(base + "R\\系数矩阵(更新).csv",
```

```
encoding='gbk')
#       print(params.shape[0])
#       costs = pd.read_excel(base+"R\\可用商品均进价.xlsx")
#       price = pd.read_excel(base+"R\\2023_6_price.xlsx")
#       print("load done.")
#       ub1 = price[price.columns[1:]].max(axis=0).to_numpy()
#
#       quantity = pd.read_excel(base+"R\\2023_6_quantity_new.xlsx")
#       ub2 = quantity[quantity.columns[1:]].max(axis=0).to_numpy()
#       print("load done.")
#       k = params['系数'].to_numpy()
#       b = params['截距'].to_numpy()
#       c = costs['批发价格(元/千克)'].to_numpy()
#       f = lambda x: np.dot(-1*k,x[:num_x]**2)-
#         np.dot(b,x[:num_x])+np.dot(c,x[num_x:])
#
#
#       bound = list(zip(np.zeros(ub1.shape[0]),ub1))
#       bound += list(zip(2.5*np.ones(ub2.shape[0]),5*ub2))
#       cons = ({'type':'ineq','fun':lambda x:k*x[:num_x]+b},
#               {'type':'ineq','fun':lambda x:x[num_x:]-k*x[:num_x]-b})
#
#
#       print("establish done.")
#       res = minimize(f,np.hstack([ub1*0.93,ub2*0.88]),constraints=cons,bounds=bound)
#
#
#       print("最优解:",res)
#       return (k*res.x[:num_x]+b)*res.x[:num_x]-c*res.x[num_x:],res.x[:num_x],k*res.x

# profit, price, expect_sale,supply = Solve(51)
# df = pd.DataFrame({"price": price, "expect_sale": expect_sale,"supply":supply,"pr
# df.to_excel(base + "R\\3.xlsx", index=False)

def select(columns):
    costs = pd.read_excel(base+"R\\可用商品均进价.xlsx")
    df = pd.read_excel(base+"R\\2__.xlsx")
    df['loss'] = 0
    df['supply'] = df['expect_sale']
```

```
    df['name'] = columns
    for i in range(df.shape[0]):
        if df.loc[i,'supply'] <= 2.5:
            df.loc[i, 'supply'] = 2.5
        df.loc[i,'loss'] = (df.loc[i,'loss']-
        df.loc[i,'expect_sale'])*costs.loc[i,'批发价格(元/千克)'] +
        df.loc[i,'price']*df.loc[i,'expect_sale']
        df.loc[i,'profit'] =
        df.loc[i,'expect_sale']*df.loc[i,'price'] -
        df.loc[i,'supply']*costs.loc[i,'批发价格(元/千克)']
    num = df[df['loss']>0].shape[0]
    df = df.sort_values('loss',ascending=False)
    if num >=27 and num <= 33:
        df = df[:num]
    elif num < 27:
        df = df[:27]
    else:
        df = df[:33]

    df.set_index('name', inplace=True)
    df.to_excel(base + "R\\3__.xlsx")

df = pd.read_excel(base+"R\\2023_6_quantity_new.xlsx")
select(df.columns[1:])
```