蔬菜类商品的自动定价与补货决策

摘要

本研究旨在解决生鲜商超中蔬菜类商品的补货和定价决策问题。蔬菜类商品的保鲜期较短且品相会随销售时间的增加而变差，在认定理想情况下当日未销售出去的商品隔日不再进行出售。由于不确定具体商品和进货价格情况，不考虑节假日的影响，需从“售价低，销量高”的生活经验入手，建立售价与销量之间的联系，从而获取最大利润。

问题一旨在分析蔬菜各品类和单品之间的销售量分布规律及相互关系。不论是品类还是单品都可以分为一年四季春、夏、秋、冬来考虑分布情况。接下来通过检验是否满足正态分布选用相应的相关系数来绘制热力图分析相互关系，由于单品数据种类较多，因此进一步对其做层次聚类分析，分成 20 类从而体现单品之间的关系。

问题二考虑商超以品类为单位制定补货计划，在以品类为单位分析各蔬菜品类的销售总量与成本加成定价的关系时，某日品类销售总量为该品类中所有单品的某日销售总量。某日品类平均价格为该品类中所有单品价格的加权平均值，权重为销售量。绘制散点图观察数据的分布后，采用 6 个线性模型分别对不同品类进行拟合，并做了线性假设检验，经检验 6 个品类的定价与销量都满足线性关系，最终以花叶类为例，订购 54.2326kg，定价为 59 元/kg。

问题三商超希望进一步制定单品的补货计划。本问相对于问题二增加了两个约束，统计 2023 年 6 月 24 日至 30 日的 49 种可售品种中有 45 种均可收益，所以我们选择尽可能的订购商品也就是订购 33 种可售商品进行剪枝优化，考虑理想情况下若售货量高于最低订购量（1-损耗率），为了避免浪费，订购量*（1-损耗率）就等于销售量，反之，则选择最低订购量。对这 45 种可收益的单品，建立线性拟合模型，对于呈负相关且相关系数<-0.1 的 14 组数据采用问题 2 的模型，计算收益，对于相关系数>-0.1 即不表现明显线性关系数据采用近一周内（2023 年 6 月 24 日至 30 日）中收益最高的一天的订购量和售价作为 7 月 1 日的单品补货量和定价策略，最后，将这 45 种单品收益进行排序选择前 33 个，例如上海青，补货 9.95kg，单价 5 元/kg。

问题四商超需要采集哪些相关数据以更好地制定蔬菜商品的补货和定价决策，并对解决上述问题有何帮助。由于前面的问题都是在理想情况下和只根据题目所提供的数据建模，对于此问题，建议从实际出发，收集与销售量、销售价格、供应商信息、商品质量和损耗率相关的数据，以便更好地了解市场需求和供应情况，从而支持决策制定过程。

关键词：自动定价与补货决策 相关性分析 层次聚类 线性模型 剪枝优化

一、问题重述

1.1 问题背景

在生鲜商超中，由于蔬菜类的保鲜期较短，随着时间的推移品相变差，大部分品种若当日未售出则隔日无法再售。所以，商超一般会根据各商品的历史销售和需求进行每天的补货。

蔬菜品种和所属产地多，而蔬菜的进货交易时间多为凌晨，所以商家需要早不确切知道具体单品和进货价格的情况下做出当日各蔬菜品类的补货决策。定价一般采用“成本加成定价”方法，对有损坏或品相差的商品一般需要打折出售。在补货决策和定价决策中，可靠的市场需求分析尤为重要。从需求方面来看，蔬菜类的商品销售量与时间存在着关联，从供给侧看，蔬菜的供应在 4~10 月较为丰富，但由于商超销售空间有限，所以合理的销售组合也显得极为重要。

1.2 问题提出

基于背景，根据某商超经销的 6 个蔬菜品类的商品信息、2020.7.1~2023.6.30 各商品的销售流水明细与批发价格的数据及各个商品近期损耗率数据和实际情况，建立数学模型解决问题。

问题一：蔬菜类商品不同品类或不同单品之间可能存在一定的关联关系。根据附件分析出蔬菜各品类及单品销售量的分布规律及相互关系。

问题二：考虑商超以品类为单位做补货计划，通过分析各蔬菜品类的销售总量与成本加成之间的关系，给出各个蔬菜 2023.7 月 1-7 日的补货总量和定价策略，从而使商超的收益最大。

问题三：由于蔬菜类的商品销售空间存在限制，商超希望进一步制定单品的补货计划，需要可售单品总数控制在 27~33 之间，各单品订购量满足最小陈列量 2.5 千克的要求。根据 2023 年六月最后一个星期的可售品种，给出 7.1 的单品补货量及定价策略，在满足市场需求的前提下，商超收益最大。

问题四：分析并提出商超还需要采集那些数据能更好的制定蔬菜商品的补货和定价的决策，数据对于解决上述问题有什么帮助。

二、问题分析

2.1 问题一的分析

根据附件 1 及附件 2 的数据，可以得到每种单品对应的销量、单价及单品对应的分类等信息。为了分析蔬菜类商品中不同品类或者不同单品之间的关系，首先对数据进行清晰，剔除异常值情况。由于存在 3 年的数据，需要考虑对数据进行划分。根据我国所存在的维度信息，将数据划分为四个季度：春季（3-5 月），夏季（6-8 月），秋季（9-12 月），冬季（12-2 月）。可以通过可视化数据分布及计算数据之间的均值方差从而观察四季数据的分布规律[2]。

在此基础上，由于品类类别少，首先判断品类的销售量是否满足正态分布，在满足正态分布的前提下进行相关性分析，从而得出品类之间的相关性。对于单品数据，由于单品种类繁多，考虑各个单品的销售量的分布关系选用合适的相关系数，最后利用聚类分析分成 20 个类对单品之间的关系分析。

2.2 问题二的分析

考虑商超以品类为单位做补货计划，分析各蔬菜品类的销售总量与成本加成定价的关系，所以将品类中的不同单品加在一起当成一个整体来看。以日为单位，分别统计每日各品类的销量和定价，来进行分析。某各品类的销售总量为品类中单品的销售量之和。同理平均价格也是。然后对每个品类的销售量和价格之间的数据分布进行可视化，观察数据的分布类型，采用回归模型对数据进行集合，得到销售量与定价的关系，从而对未来一周补货定价提供策略。

2.3 问题三的分析

根据 2023 年 6 月 24 日至 30 日的可售品种，需要给出 7 月 1 日的单品补货量和定价策略，以最大化商超的收益，并尽量满足市场对各品类蔬菜商品的需求。本问在问题二的基础上考虑统计出在该时间段可收益商品的数量。选择尽可能的订购商品也就是订购 33 种可售商品进行剪枝优化，首先考虑模型是否满足负相关性也就是实际情况下的售价越高销量越低。考虑理想情况下若售货量高于最低订购量（1-损耗率），为了避免浪费，订购量*（1-损耗率）等于销售量，反之，则选择最低订购量。对这 45 种可收益的单品，建立线性拟合模型，对不同的相关性模型采用不同的数学策略，最终建立利润和销量、定价、订货量之间的关系。

2.3 问题四的分析

问题四考虑从实际出发，收集与销售量、销售价格、供应商信息、商品质量和损耗率相关的数据，以便更好地了解市场需求和供应情况，从而支持决策制定过程。

三、模型假设

1、假设该商超统计的数据完全正确，即某日的销量突增，认为是正常现象
2、假设该地区人们的生活习惯和经济水平稳定
3、假设该地区季节划分符合中国自然维度季节划分，没有特殊气候情况
4、假设销量的变化是独立的，即过去销量的变化不会直接影响未来销量的变化
5、假设数据中所给的定价是稳定的，没有经过人为刻意调控
6、假设对于遗漏的某日蔬菜批发价格可以用近段日子的价格代替
7、假设所有蔬菜当日未售出，不会留到次日继续出售
8、假设不存在某一单品是某日必须要出售的
9、假设不存在节假日对于蔬菜出售情况的影响

四、符号说明

表 1 符号说明

变量	解释
W_j	问题一中第 j 个样品的统计量
r_{jk}	问题一中第 j 个和第 k 个品类的皮尔森相关系数
S_{jk}	问题一中第 j 个和第 k 个单品的斯皮尔曼相关系数
d_{jk}	问题一中第 j 个和第 k 个单品的距离
$SALE_i$	问题二中第 i 个单品的销量
u_i	问题二中第 i 个单品的利润
m_i	第 i 个单品的损耗率
S_i	问题二中第 i 个单品的成本
$SALE'_{ik}$	问题三中第 i 个单品第 k 天的销量
X_{ik}	问题三中第 i 个单品第 k 天的售价
$AVER_i$	问题三中第 i 个商品前一个月的平均成本

五、问题一模型的建立与求解

5.1 数据预处理

首先对数据进行预处理剔除异常值，对于本问题分析蔬菜各品类及单品销售量的分布规律及相互关系，使用到了附件 1 和附件 2 中的数据，剔除附件 2 中销售类型为“退货”的 461 条数据。

附件 1 给出了某商超经销的 6 个蔬菜品类的商品信息；附件 2 给出了该商超 2020 年 7 月 1 日至 2023 年 6 月 30 日各商品的销售流水明细。如果直接对于数据进行统计或者以月份为整体进行统计，数据量会比较大不利于分析，因此从实际出发，按照中国的维度进行季度划分：第一季度:3-5 月(春季)、第二季度:6-8 月(夏季)、第三季度:9-11 月(秋季)、第四季度:12-2 月(冬季)。为了数据的稳定和普适性。将三年当做一个整体来考虑，不用逐年分析。

5.1 蔬菜各品类销售量的分布规律

利用 python 按照类别对三年数据的春夏秋冬进行统计分析，绘制柱状图如图 1-4 所示：

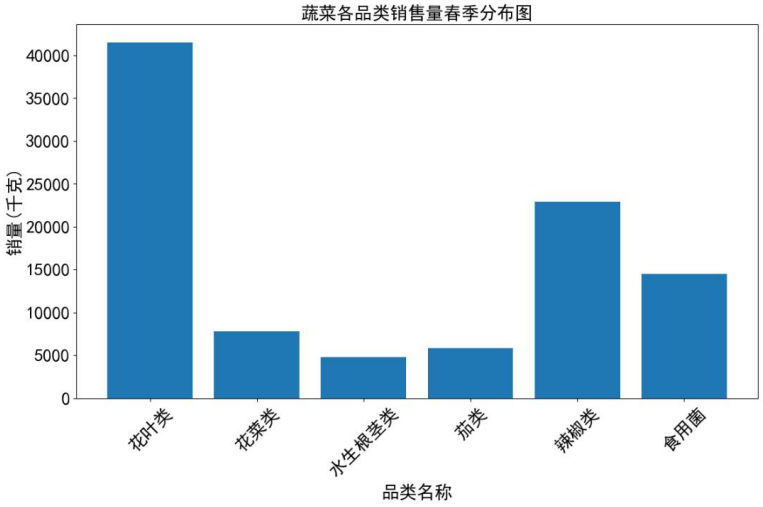


图 1 春季数据分布

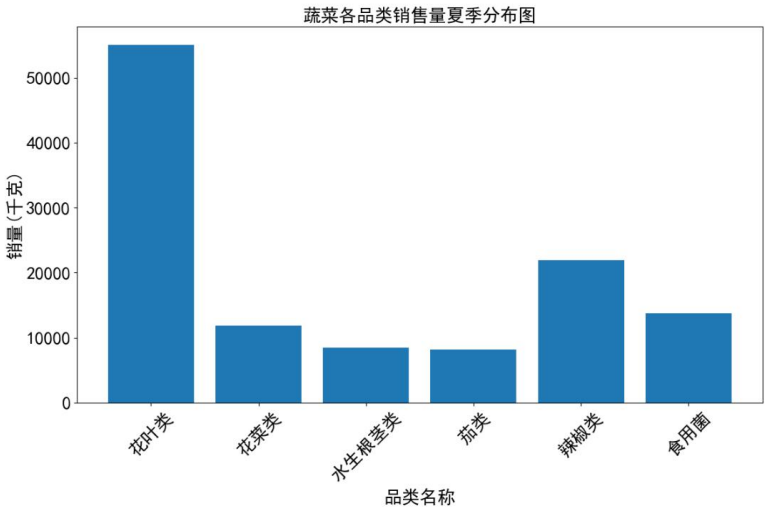


图 2 夏季数据分布

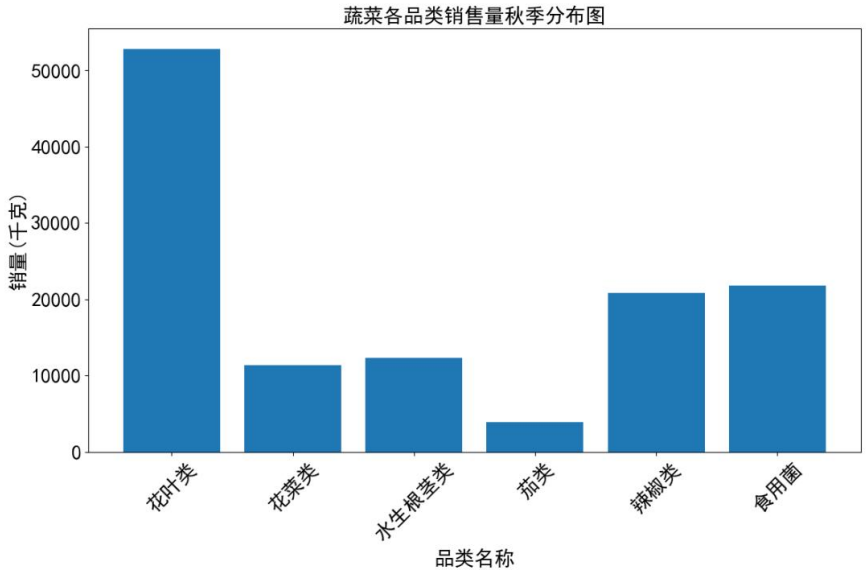


图 3 秋季数据分布

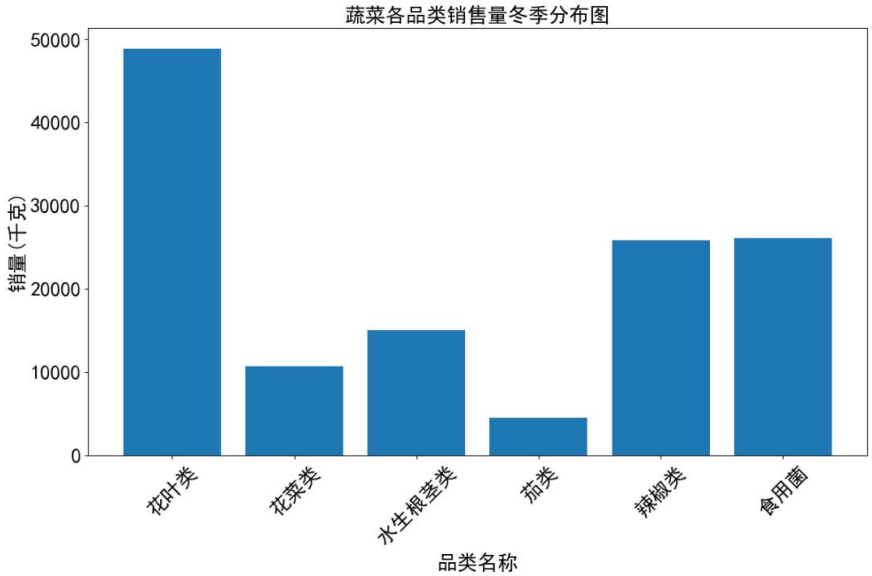


图 4 冬季数据分布

对比四个季度的销售情况，不难发现花叶类蔬菜在四个季节销量都很高，因为它们是许多菜肴的主要成分之一。花叶类蔬菜、花菜类蔬菜和辣椒类蔬菜在四季的分布都比较稳定，受季节因素影响较小，这不仅是因为产量的稳定，也是因为人们日常生活都离不开这些类型的蔬菜，达到供需稳定。水生根茎类蔬菜和食用菌蔬菜在秋冬季的销售量会达到一个增长，这是因为它们是耐寒的蔬菜，能够在寒冷的季节生长并保持较长的储存寿命。而茄类蔬菜在春夏季节的销售量较高，这是因为它们喜欢温暖的气候，并且在这些季节能够获得充足的阳光和水分。此外，该地区的蔬菜销售情况也和该地区人们的饮食习惯有关。

对数据归一化后计算各个季节品类的均值与方差如下：

表 2 四季数据均值和方差

季度	均值	方差
春	0.39072	0.2490

夏	0.3607	0.2187
秋	0.3877	0.2384
冬	0.4468	0.2859

从上表不难发现，尽管对于不同的品类销售量的差距比较大，在春季各个品类相对销量差距最小，夏季差距最大。

5.1 蔬菜各单品销售量的分布规律

利用 python 按照各个单品对三年数据的春夏秋冬进行统计分析，绘制柱状图如下所示：

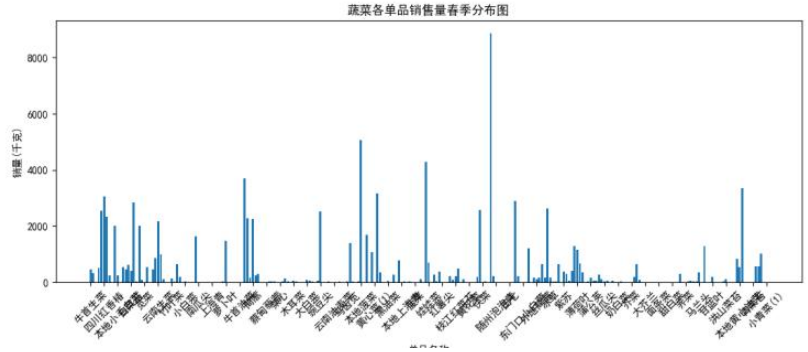


图 5 春季单品分布柱状图

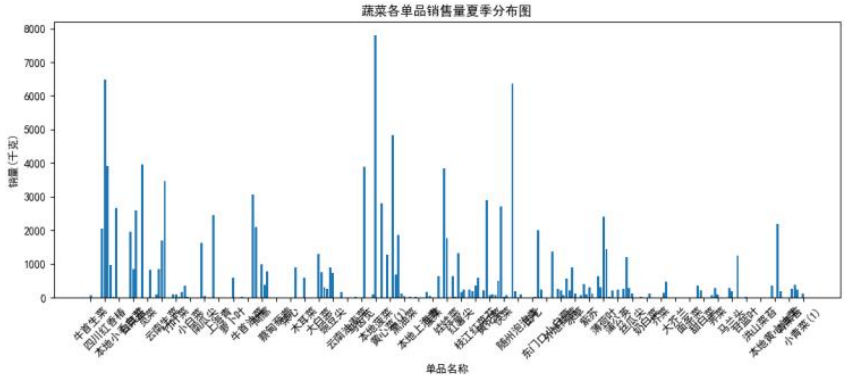


图 6 夏季单品分布柱状图

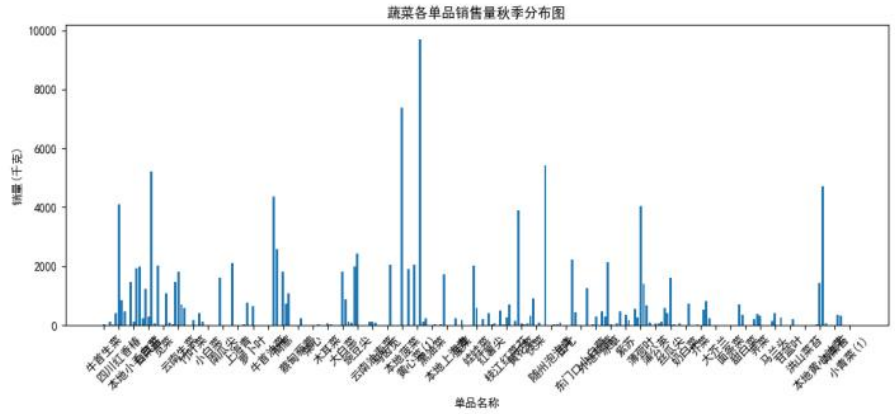


图 7 秋季单品分布柱状图

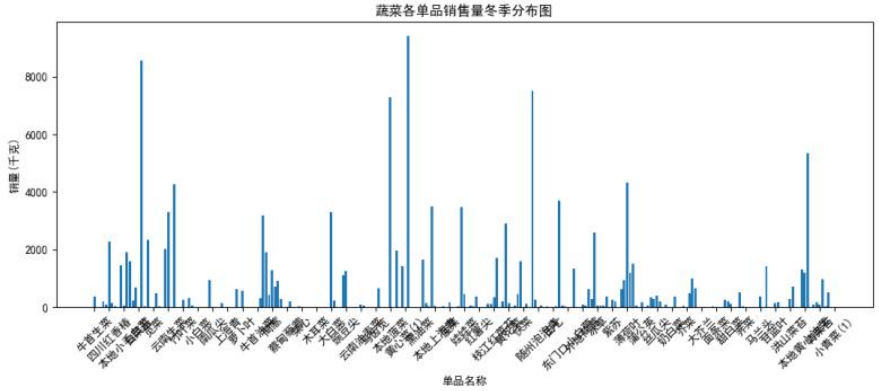


图 8 冬季单品分布柱状图

注：因为横坐标数量较多，画图时候显示完全会彼此覆盖，故每隔 5 个输出一个横坐标

在根茎类蔬菜中，藕和藕尖通常是销售量最高的单品之一，因为它们是许多主食菜肴的主要成分。

在花叶花菜类蔬菜中，菠菜和生菜通常是销售量较高的单品，因为它们广泛用于沙拉和炒菜等菜肴。

在茄类蔬菜中，紫茄子和紫圆茄通常是销售量较高的单品，因为它们可以生食，也是炒菜和烹饪中常用的配料。

在辣椒类蔬菜中，红尖椒和青尖椒通常是销售量较高的单品之一，尤其在一些亚洲地区和健康饮食趋势中。

此外，可以从图上看出，不同季节对于单品的销售量很大，这是由于部分蔬菜的生长周期的原因。

5.2 各品类相互关系模型的建立与求解

5.2.1 正态分布检验模型建立

在检验正态分布方面，本问采用 Shapiro–Wilk test，这是一种在频率上统计检验中检验正态性的方法。

这个统计检验的假设是样本来自于一个正态母体，因此一方面，如果 p 值小于选择的显著度水平（α值通常 0.05），那么在更大概率下应该拒绝零假设，数据的证据显示样本不是来自一个正态分布母体。另一方面，如果 p 值比选择的显著度水平大，那么无证据拒绝零假设，则数据来自于一个正态分布。

检验统计量为：$W_j = \frac{(\sum_{i=1}^{n} a_i^j x_{(i)}^j)^2}{\sum_{i=1}^{n}(x_i^j - \bar{x}^j)^2}$

其中，$x_{(i)}^j$ 表示第 j 个品类样本数据中第 i 阶统计量，即样本中第 i 个最小值。

$\bar{x}^j$ 表示第 j 个品类中的样本均值。

a_i^j 通过公式：$(a_1^j, a_2^j, ...a_n^j) = \frac{m_j^T V_j^{-1}}{(m_j^T V_j^{-1} V_j^{-1} m_j)^{1/2}}$

其中在第 j 类品类中，m_j^T 从一个标准的正态分布随机变量上采样的有序独立同分布的统计量的期望值。V_j^{-1} 是这些有序统计量的协方差。

5.2.2 正态分布检验模型求解

利用 SPSSPRO 对蔬菜各品类进行正态分布检验，结果如表 3 所示：

表 3 正态分布检验 p 值

品类名称	p 值
花叶类	0.064
花菜类	0.486
水生根茎类	0.658
茄类	0.502
辣椒类	0.060
食用菌	0.128

经检验，p 值大于 0.05，拒绝原假设，蔬菜各品类均满足正态分布。

5.2.3 皮尔森相关系模型的建立

由于蔬菜各品类数据均满足正态分布，因此可以采用皮尔森相关系数计算相关性。它衡量的是两个变量之间的线性关系的程度，取值范围在-1 到 1 之间。当相关系数接近 1 时，表示呈现出较强的正相关性；当相关系数接近-1 时，表示呈现出较强的负相关性；当相关系数接近 0 时，表示两个样本数据集之间几乎没有线性关系。计算公式如下所示[6]：

$$r_{jk} = \frac{Cov(X_j.X_k)}{\sqrt{Var(X_j * X_k)}}$$

其中，$Cov(X_j.X_k)$ 表示第 j 个品类和第 k 个品类样本之间的协方差，$Var(X_j)$ 表示第 j 个样本的方差。

5.2.3 各品类相互关系的求解

利用皮尔森相关系数，对春夏秋冬四个季节的各品类之间的相关性进行计算，并绘制热力图，如图 9 所示。

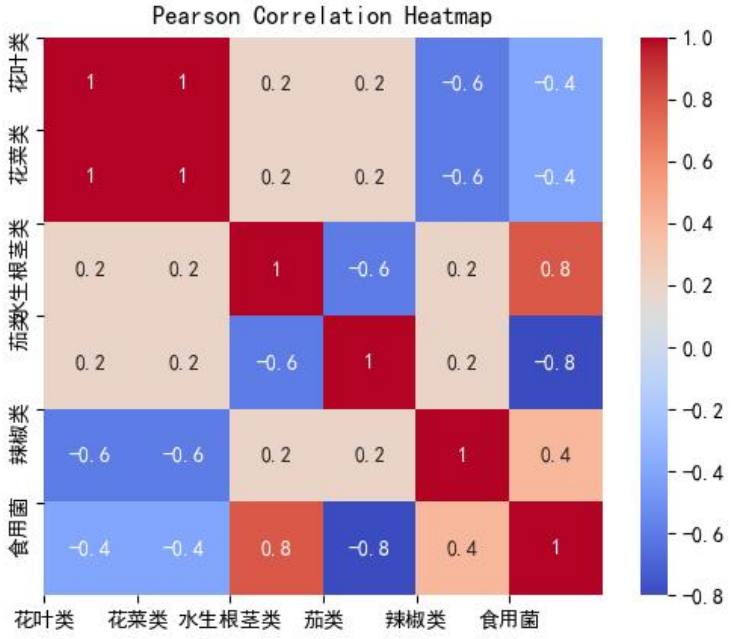


图 9 品类相关性热力图

从上图可以看出花叶类和花菜类蔬菜，水生根茎类和食用菌蔬菜有着很强的正相关性，即表现出一定的共需关系；而茄类和食用菌类蔬菜，茄类和水生根茎类蔬菜有着很强的负相关性，即表现出一定的替代关系。

5.3 各单品相互关系模型的建立与求解

5.3.1 斯皮尔曼相关系数模型的建立

首先我们对蔬菜各单品的销售量进行正态分布检验。根据检验结果，并不符合正态分布。与皮尔逊相关系数不同，斯皮尔曼相关性是基于变量的排名而不是原始值进行计算的，因此不依赖于数据的分布假设。计算公式如下所示：

$$S_{jk} = 1 - \frac{6\sum_{i=1}^{7}(Z_{ij} - Z_{ik})^2}{N(N^2-1)}$$

其中S_{jk}表示第 j 个单品和第 k 个单品之间的斯皮尔曼相关系数，Z_{ij}和Z_{ik}分别表示第 j 个单品和第 k 个单品在第 i 个样本中的表达量的秩次差，秩序差即为排序前后所在排名的差别。

5.3.2 斯皮尔曼相关系数模型的建立

利用 python 计算出各单品之间的斯皮尔曼相关系数，热力图如下所示：

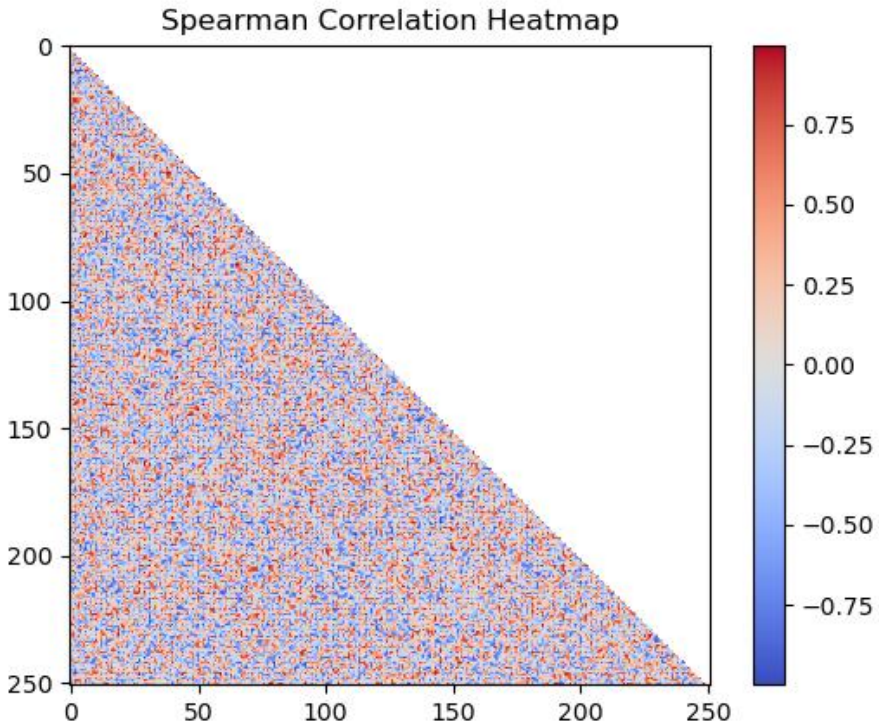


图 10 单品相关性热力图

注：由于横坐标数量太多，并没有显示出具体坐标名称；由于数量较多且满足对称性，为了便于查看，只绘制下半部分热力图。

5.3.4 层次聚类模型的建立

由于单品类型繁多，直接对所有单品进行相关系数分析不能直观的体现出单品之间的相关性，所以本问采用聚类分析的方法，从而更加直观的分析不同单品之间的相互关系。

在层次聚类中，选择合适的距离度量是非常重要的。使用单链层次聚类算法，聚类簇之间的距离被定义为簇中最近两个样本之间的距离。每个样本最初被视为一个独立的聚类簇，然后通过将最近的两个聚类簇合并来逐步形成更大的聚类簇。聚类过程将一直持续，直到所有的样本都被合并为一个聚类簇。具体算法流程如下：

Step1.初始化：将每个单品视为一个单独的簇。

Step2.计算距离矩阵：计算每两个单品之间的距离，并将其保存为距离矩阵。

Step3.合并相近的簇：找到距离矩阵中距离最小的两个簇，并将它们合并为一个新的簇。

Step4.更新距离矩阵：根据新簇的距离更新距离矩阵。

Step5.重复 Step3 和 Step4，直到所有单品之间的距离不满足合并的阈值条件。

在 5.3.1 中计算了S_{jk}，即表示第 j 个单品和第 k 个单品之间的斯皮尔曼相关系数，按照如下公式将该相关系数转化为上述步骤 2 中所需距离度量：d_{jk}表示

第 j 个单品和第 k 个单品之间的距离。

$$d_{jk} = \sqrt{2(1-S_{jk})}$$

5.3.5 层次聚类模型的求解

利用 python 对四季的各类单品进行聚成 20 类[4]，聚类可视化如图 11 所示，20 类具体的聚类结果见附录 1。

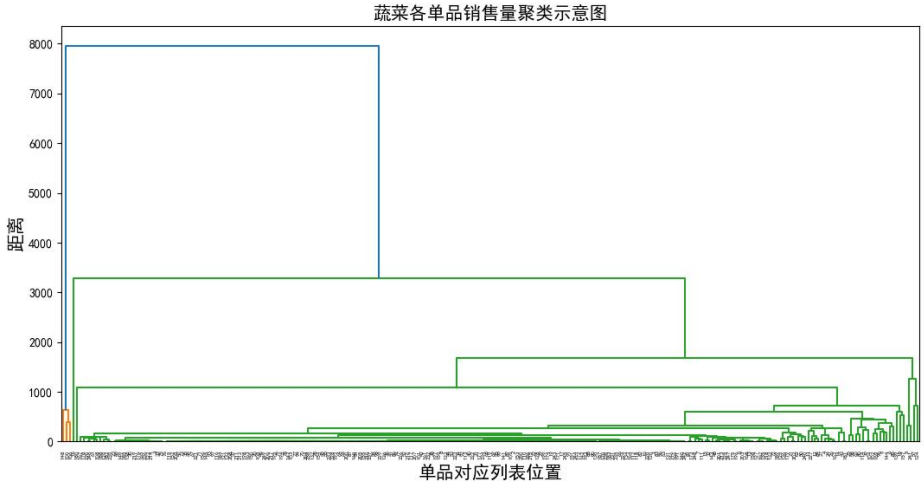


图 11 总的聚类分析图

从图上发现，在模型初期，距离较小时，便有多数单品已经聚成簇，这说明很多单品之间的相关性强。

以一类举例：['四川红香椿', '萝卜叶', '枝江红菜苔(份)', '小青菜(2)', '鱼腥草(份)', '苋菜(份)', '小白菜(份)', '野藕(1)', '高瓜(2)', '净藕(3)', '紫茄子(1)', '红尖椒', '七彩椒(1)', '灯笼椒(1)', '红灯笼椒(1)', '青杭椒(份)', '水果辣椒(份)', '七彩椒(2)', '红灯笼椒(2)', '姬菇(1)', '海鲜菇(1)', '银耳(朵)', '鲜木耳(1)', '西峡香菇(2)', '金针菇(2)', '蟹味菇与白玉菇双拼(盒)', '西峡香菇(份) ', '蟹味菇(袋)', '白玉菇(2)']。可以发现例如像辣椒不同种类的单品，喜欢吃辣椒的都会选择进行购买，喜欢吃菇类的人也会购买各式各样的菇类单品。并且该聚类中的很多单品都会作为主菜的配菜。

由于上图坐标过多，看不得不够清晰，我们随机取 20 个单品做一个部分聚类查看效果

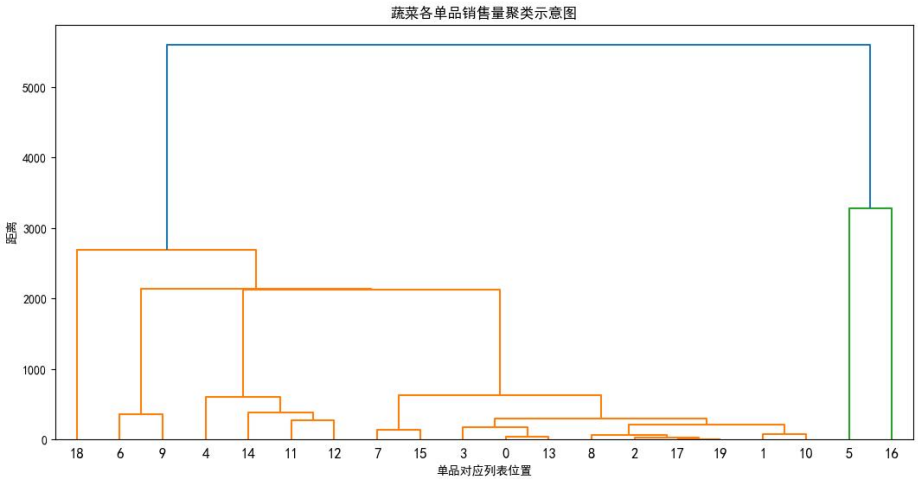


图 12 部分聚类示意图

注：上图中横坐标序号代表在以下列表中的位置

['牛首生菜', '四川红香椿', '本地小毛白菜', '白菜苔', '苋菜', '云南生菜', '竹叶菜', '小白菜', '南瓜尖', '上海青', '萝卜叶', '牛首油菜', '茼蒿', '蔡甸藜蒿', '菜心', '木耳菜', '大白菜', '豌豆尖', '云南油麦菜', '马齿苋']

六、问题二模型的建立与求解

6.1 数据预处理

考虑商超以品类为单位做补货计划，分析各蔬菜品类的销售总量与成本加成定价的关系，显然，本问将品类中的不同单品加在一起当成一个整体来看。以日为单位，分别统计每日各品类的销量和定价，来进行分析。

在以品类为单位分析各蔬菜品类的销售总量与成本加成定价的关系时，计算品类某日销售总量：对于每个品类，计算该品类中所有单品的某日销售总量。计算某日品类平均价格：对于每个品类，计算该品类中所有某日单品价格的平均值。

对每个品类每日的销售量和价格，绘制数据分布散点图，以花叶类为例如图 12 所示。可以看出数据分布较为集中。

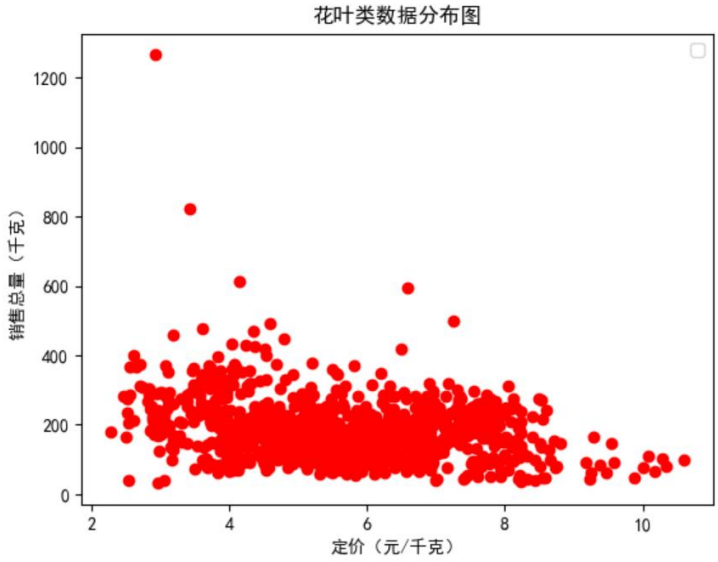


图 13 花叶类数据分布

6.2 一维线性回归模型的建立

根据数据预处理，可以发现每个品类的销售总量和定价之间的关系具有一维线性关系。为了探索它们之间的相关性，采用一维线性回归模型。

首先建立一个全面的蔬菜品类销量与成本加成定价的关系公式[1]，引入与蔬菜销售和市场有关的变量和线性关系[3]。

$$SALE_i = \beta_0^i + \beta_1^i * 定价 + \beta_2^i * 季节因素 + \beta_3^i * 促销因素 + \beta_4^i * 竞争因素 + \varepsilon_i$$

在这个公式中，除了定价作为一个变量外，还引入了其他几个因素来考虑更多的影响因素：

季节因素：考虑到蔬菜销售量可能因季节性变化而有所不同，可以引入一个季节因素的变量，如季节指示变量或季节性指数。

促销因素：考虑到促销活动可能对销售总量产生影响，可以引入一个促销因素的变量，如是否进行促销的二元变量。

竞争因素：考虑到市场竞争对销售总量的影响，可以引入一个竞争因素的变量，如市场份额或竞争对手的数量。

β_0^i，β_1^i，β_2^i，β_3^i 和 β_4^i 是待估计的系数，代表了对于第 i 类单品各个变量对销售总量的影响程度。ε_i 是误差项。

进一步考虑，本问题是针对品类来分析，在问题一中我们分析到各个品类的蔬菜受季节影响仍表现出较稳定的销量，因此无需考虑季节因素影响。对于任何一个单品，显然促销对于销量会有很大的影响，但对于品类而言，我们可以将某一单品促销与否当做是该品类下的两种不同单品，这样便可以忽略促销因素的影响。而竞争因素在本题中并没有提供别的商超的数据，也不予考虑。

因此可以将模型简化为：$SALE_i = \beta_0^i + \beta_1^i$*定价+$\varepsilon_i$ 。

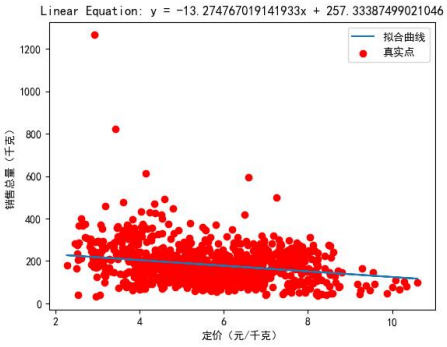
6.3 一维线性回归模型的求解

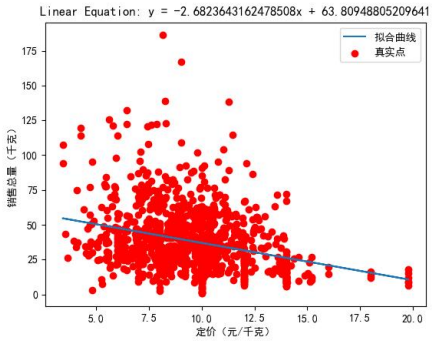
利用 python 对各品类销售量和定价数据进行一维线性回归[5]，结果如表 4 所示：

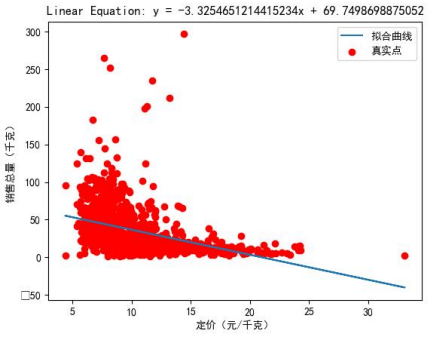
表 4 线性回归结果

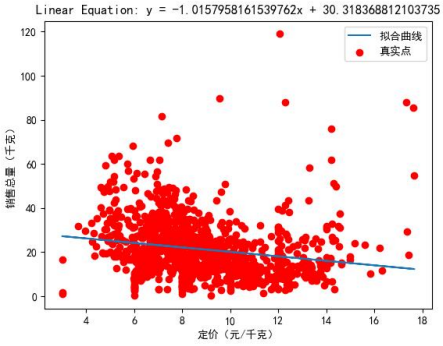
=类别	β_0	β_1	ε
花叶类	257.3338	-13.2748	1.6808
花菜类	63.8095	-2.6824	0.2657
水生根茎类	69.7499	-3.3255	0.2512
茄类	30.3187	-1.0158	0.1600
辣椒类	106.2680	-2.5279	0.4310
食用菌	114.5456	-5.1796	0.5878

拟合可视化结果如图 13 所示：









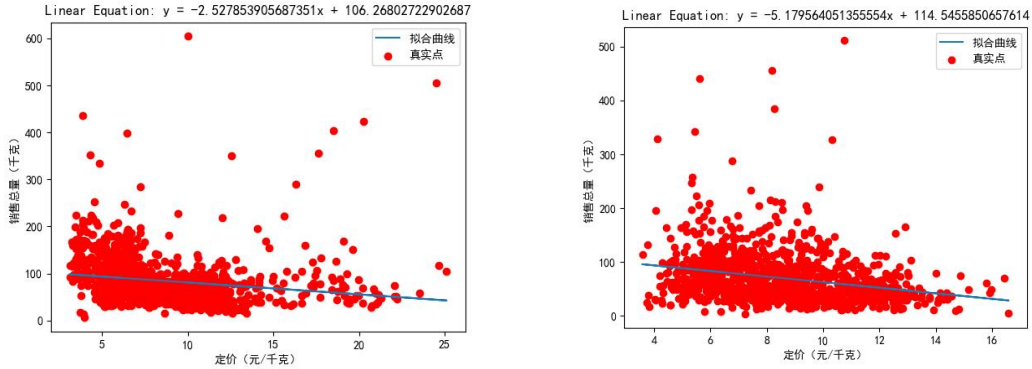


图 14 线性回归可视化结果

下面对回归方程进行显著性检验，p 值如表 5 所示：

表 5 显著性检验 p 值

类别	p 值
花叶类	6.919309807695311e-15
花菜类	5.80445671188664e-23
水生根茎类	3.3542036389938344e-37
茄类	3.2180026075023e-10
辣椒类	5.955220380168485e-09
食用菌	4.836174586147857e-18

根据表格中的 p 值显示，p<0.05，拒绝原假设，符合一维线性关系。

所以六种品类的销售量与定价之间的关系为：

$$\begin{cases} SALE_1 = 257.3338 - 13.2748X_1 + 1.6808 \\ SALE_2 = 63.8095 - 2.6824X_2 + 0.2657 \\ SALE_3 = 69.7499 - 13.2748X_3 + 1.6808 \\ SALE_4 = 30.3187 - 1.0158X_1 + 0.1600 \\ SALE_5 = 106.2680 - 2.5279X_1 + 0.4310 \\ SALE_6 = 114.5456 - 5.1796X_1 + 0.5878 \end{cases}$$

6.3 未来一周补货定价策略

通过 6.2 的分析售价越高，销量越低，符合生活实际。考虑到未来一周的补货定价策略，若能满足每日收益最大，则一周收益最大。

设计利润公式：$u_i = SALE_i\left(X_i - S_i\right) - S_i[SALE_i / \left(1 - m_i\right) - SALE_i]$。公式的前半部分是所销售的量带来的纯利（定价-成本），考虑到蔬菜不能隔夜，当天卖不完的必须扔掉，因此还需要减去浪费的这些钱，即采购价乘上采购的量减去销量即没卖掉的量。

最终，2023 年 7 月（1-7 日）每日的补货总量、定价策略及利润如表 6 所示：

表 6 补货量、定价策略和利润

类别	销量/kg	成本/元	利润/元
花叶类	54.2326	59	1492.4124
花菜类	21.4892	19	246.7938
水生根茎类	24.1267	27	397.4905
茄类	27.3235	24	415.6560
辣椒类	40.0116	76	495.0432
食用菌	38.2336	88	629.8733

七、问题三模型的建立与求解

7.1 数据预处理与分析

首先我们统计 2023 年 6 月 24-30 日的可售品种，一共有 3502 条销售记录且销售类型都是“销售”，统计这些记录的单品共有 49 种。

图 15 数据图

对数据进行预处理，一个最直观的想法，如果这一周有单品在订购量满足最小陈列量 2.5 千克的要求下存在亏钱现象，就不在 7.1 卖。

题目规定各单品订购量满足最小陈列量 2.5 千克的要求，且需考虑各个单品的损耗率，考虑理想情况，所进的物品除了损耗没有别的亏损，故若销售量<=最小陈列量*（1-损耗率），那么进货量即为最小陈列量 2.5 千克；若销售量>最小陈列量*（1-损耗率），则进货量大小等于销售量。

经统计共有 4 种单品亏损，其余 45 种单品均盈利，如下表所示：

表 7 单品利润

单品编码	利润（元）
102900011034330	-1514.5500000000002
106949711300259	-1461.3636363636365
102900011035740	-729.4999999999999
106971533450003	-537.2399999999999
102900011033982	10.006775800711743
……	……
102900005118831	1869.797297297298

注：表格从上到下按照利润从小到大排序

到此，可以将题目中可售单品总数控制在 27-33 件这一约束条件舍去，由于有 45 件单品都获利，直接从中选取 33 个单品进行出售。

7.2 定价补货模型的建立与求解

7.2.1 夏季数据一维拟合模型

考虑到单品不同于品类，对于数据的要求要更为精准，题目要求根据 2023 年 6 月 24-30 日的可售品种，给出 7 月 1 日的单品补货量和定价策略，6-8 月均为夏季，我们只对本年度夏季也就是 2023 年 6 月 1-30 日这 45 种单品的销售情况进行拟合，给出单品编号为 102900011033982（紫茄子）的拟合情况.

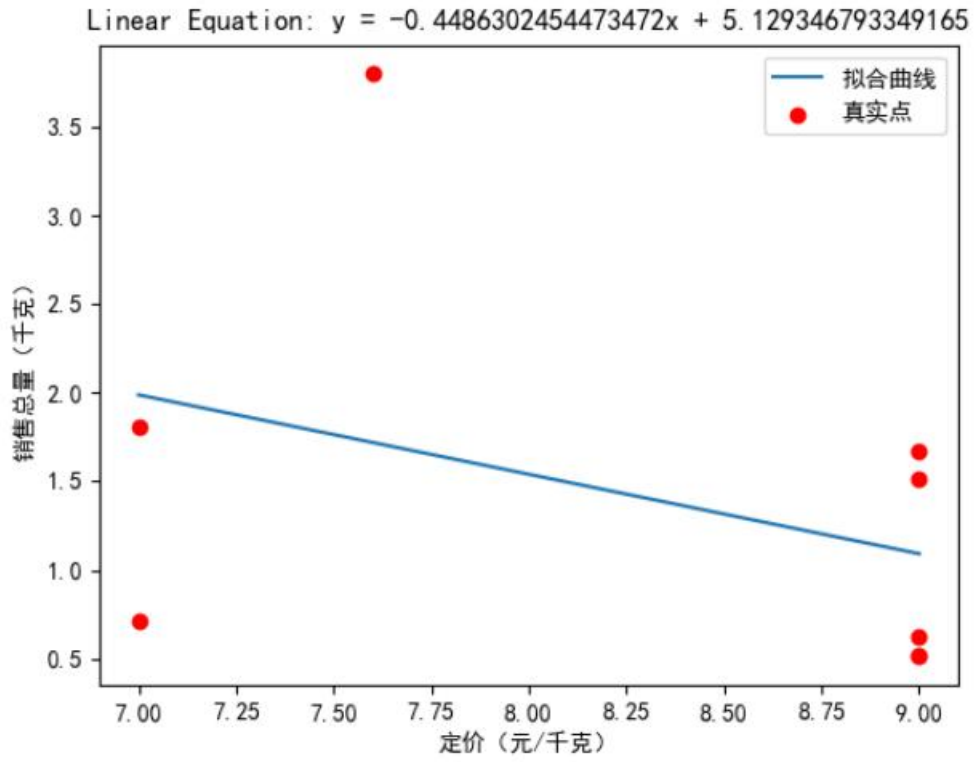


图 16 紫茄子拟合

其余单品的一维线性模型参数对应斜率和截距如下表所示，具体数据见附录 2。

单品编号	斜率	截距	相关系数
102900011033982	-0.010130579796244809	1.6679649383333153	-0.3830512303661113
102900011030929	-0.004910892109244974	1.5397547284760087	-0.8300419152950509
102900011036686	-0.002160968443204692	1.525080149932291	-0.4045493430955938
102900011032237	-0.001445935863985 2108	1.3446767097987022	-0.2730982334 5013695
102900011013274	-0.000835250275026 8047	1.534521424368716	-0.2567860114 8321817
……	……	……	……
102900011030059	-3.044932587060579e-36	8.86145925215844	-0.029652605718661983

单品编号	斜率	截距
102900011033982	0.15808431902506123	0.020919478431030758
102900011030929	1.649484536082474	2.7755575615628914e-17
102900011036686	0.2702049570300109	0.1370810614871094
102900011032237	0.5298438725963999	-0.009097008221421121
102900011013274	0.17190339759208245	0.022780242484848556
……	……	……
102900011030059	2.6195590847497705	-6.28701969027148

注：单品编号顺序与表 x 顺序一致，即按照近 7 天利润从小到大排序

7.2.2 7.1 日定价补货数学模型

经统计筛选出，可以发现有 45 个单品编号的斜率小于 0，很显然符合定价越高销量越低，然而再考虑线性相关性的时候，以-0.1 为阈值，相关系数小于-0.1 则认为定价与销量并不存在线性关系，经统计只有 14 组数据符合线性关系，而另外 31 个单品编号的相关系数并未达到阈值，故我们直接用最近 7 天内的最高收益代表 7 月 1 日可能得最高收益，为了利润最高进货量为 $\min\{2.5*(2-m_i), SALE'_{ik}\}$，其中 k 为受益最多的那天。

对于相关系数小于-0.1 的单品，即负相关的产品，参考第二问所使用的方法来计算利润，对于每天的利润考虑如下所示：

① $SALE'_{ik} \le 2.5*(1-m_i), J_{ik} = 2.5, i = 1..45, k = 1..7$

对于筛选出来的某个单品，当销量比最少进货量要小，那么直接进最少进货量 2.5kg。

此时利润为：$u'_{ik} = SALE'_{ik} * (X_{ik} - S_i) - AVER_i * (2.5 - SALE'_{ik}) i = 1..45, k = 1..7$

② $SALE'_{ik} \ge 2.5*(1-m_i), J_{ik} = SALE'_{ik}, i = 1..45, k = 1..7$

当销量比最少进货量要大，那么理想情况下进货量就等于销量。

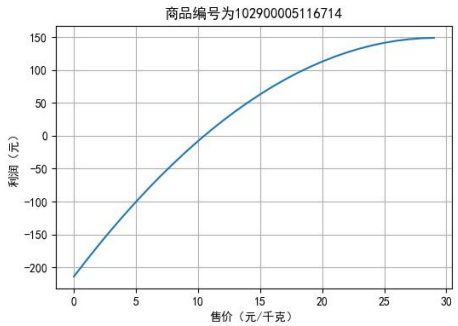
此时利润为：

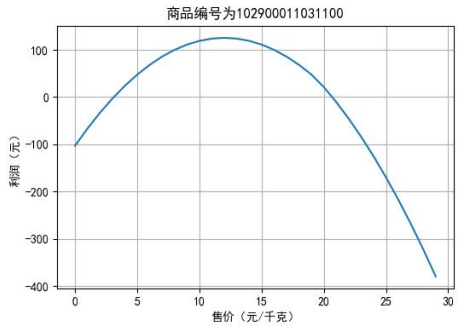
$$u'_{ik} = SALE'_{ik} * (X_{ik} - S_{ik}) - AVER_{ik} * (SALE'_{ik} / (1-m_i) - SALE'_{ik}), i = 1..45 k = 1..7$$

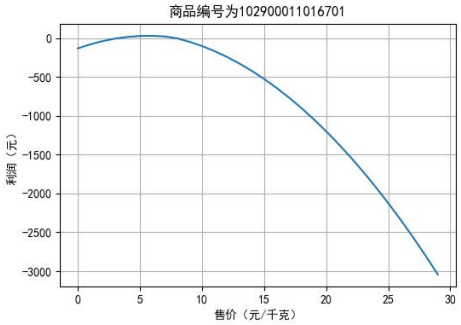
最终从 45 个单品中选出前 33 个最近 7 天的最高利润从而满足利润最大。

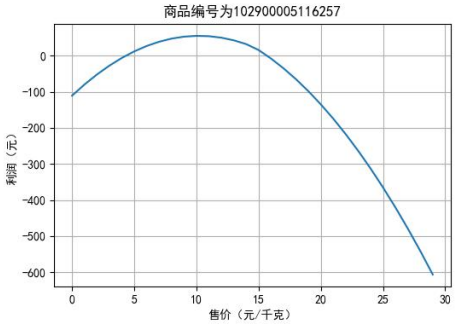
7.2.3 模型的结果

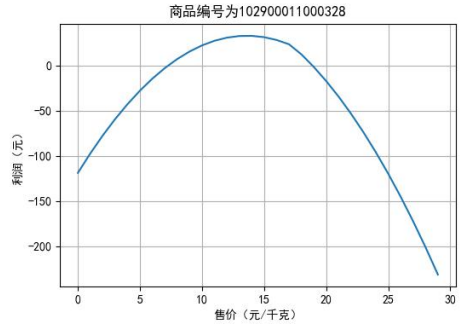
14 种销量与定价呈负相关的利润曲线图如下所示。

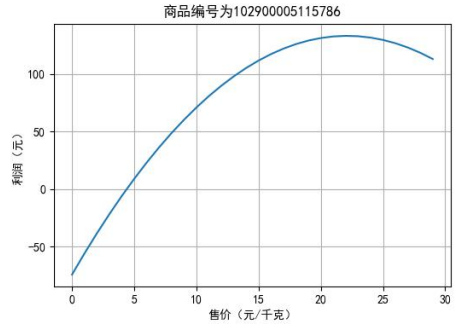


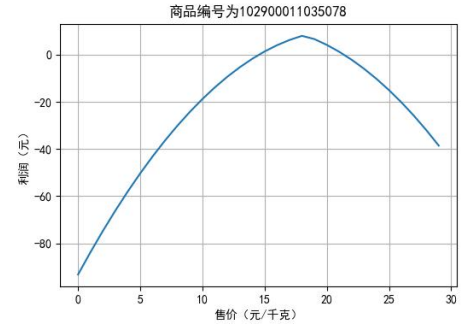


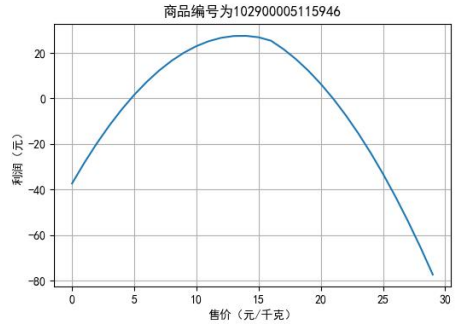

商品编号为102900011016701
利润（元）
售价（元/千克）

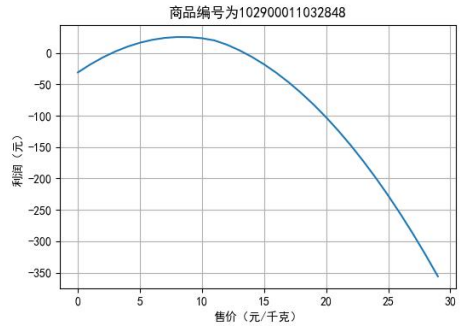

商品编号为102900005116257
利润（元）
售价（元/千克）

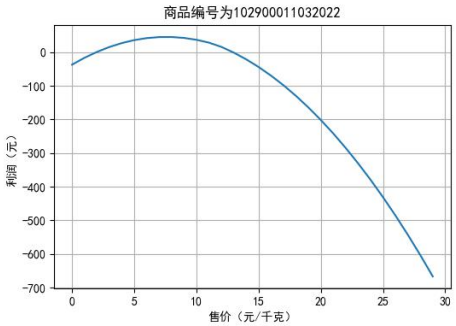

商品编号为102900011000328
利润（元）
售价（元/千克）


商品编号为102900005115786
利润（元）
售价（元/千克）


商品编号为102900011035078
利润（元）
售价（元/千克）


商品编号为102900005115946
利润（元）
售价（元/千克）


商品编号为102900011032848
利润（元）
售价（元/千克）


商品编号为102900011032022
利润（元）
售价（元/千克）

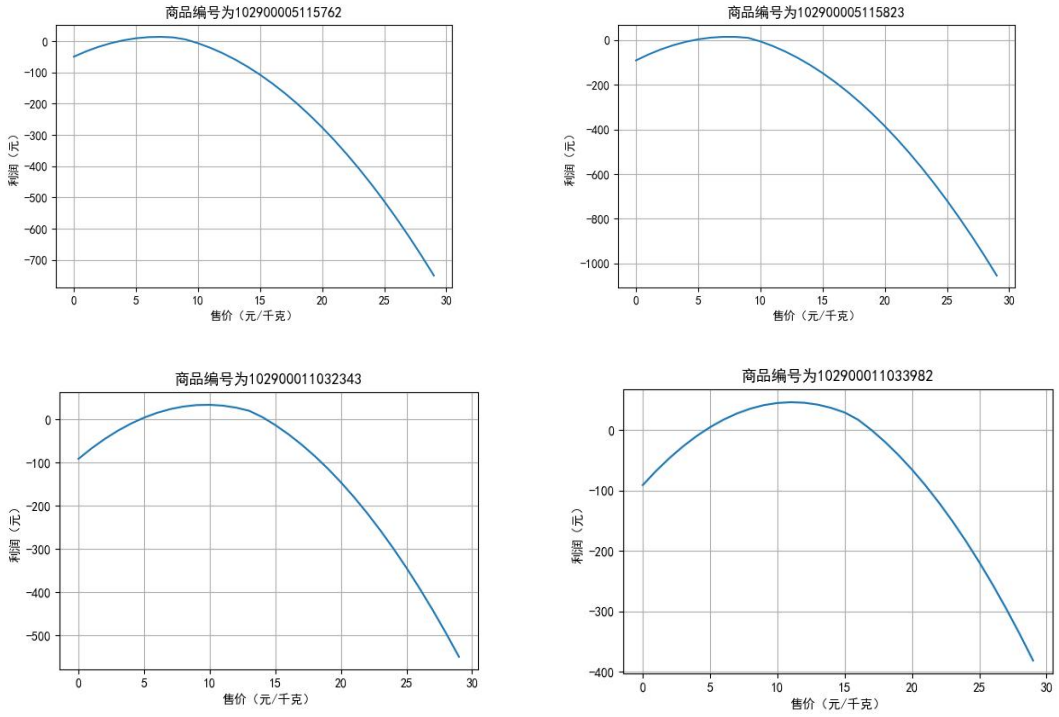


图 17 利润曲线

注：利润最高点处即为最高利润。

由上述函数及最近 7 天内部分单品的最高收益整理可得，7 月 1 日的单品补货量和定价策略如下：

表 8 单品补货定价及利润

单品编号	补货量（千克）	定价策略（元/千克）	利润（元）
102900005115823	9.950153338509322	5.0	52.70642405554685
102900005115762	4.174769274831027	7.0	35.569021564723386
102900011032022	7.4234417052767245	8.0	44.7187844185449
102900011032848	4.835772374833674	8.0	25.384289951705025
102900005115946	2.9721988916563	14.0	27.52884350929366
102900011033982	4.85447941392972	13.0	31.729533013142
102900005115786	7.565446410784713	22.0	133.12896839118295
102900011000328	4.868296972860126	14.0	32.7035518535331
102900005116257	9.68415208333348	10.0	54.92463131971883
102900011016701	10.574126373626363	6.0	28.963911371237433
102900011031100	14.098070379584648	12.0	125.21344595863701
102900005116714	8.051793657192663	29.0	148.94368931116827
102900011032343	3.584635215122623	9.8	18.873893152426228
102900005118831	12.0	6.8	70.979
102900011030059	35.9	4.5	157.457
102900005116899	6.389	16.0	76.945
102900011031100	33.0	5.8	107.414

102900011016701	14.749	5.2	58.871
102900011000328	7.427	12.0	76.789
102900011032251	19.0	5.9	97.762
102900005115786	10.155	3.4	20.232
102900011034026	9.151	14.0	92.668
102900011030110	10.0	6.8	42.844
102900005118824	3.426	16.0	30.768
102900011032022	8.0	2.8	13.756
102900005116509	3.9	6.0	12.565
102900011023464	7.552	5.2	21.424
102900011001691	4.942	14.0	46.667
102900005115779	3.168	9.2	18.822
102900011032732	2.5	18.0	9.768
102900005118817	2.845	14.0	23.376
102900011034439	4.0	5.8	12.236
102900011006948	3.876	12.0	28.956

八、问题四的求解

为了更好地制定蔬菜商品的补货和定价决策，商超可以考虑采集以下相关数据：

销售数据：收集更多地区的每种蔬菜商品的销售数据，包括销售数量、销售额、销售地点/门店、销售时间等。这些数据能够提供蔬菜商品的需求和销售趋势，帮助判断哪些蔬菜商品受欢迎，哪些可能需要调整补货策略或定价，以此数据建模可以更好地展现出蔬菜品类或单品的相关性。

库存数据：在本题中题目说大部分蔬菜品种今日若未售出第二天不再出售，按理想情况下规定了所有蔬菜今日未售出第二天均不再出售。但在实际情况下应该跟踪蔬菜商品的库存情况，包括每种蔬菜商品的库存量、库存周转率等。库存数据可以帮助判断是否存在库存积压或缺货情况，以便及时调整补货策略。

供应商数据：不同供应商的实时进货价格肯定是不一样的。收集关于蔬菜供应商的数据，包括供应商的稳定性、供应能力、交货准时率等信息。这些数据可以帮助商超评估供应商的可靠性，确保蔬菜商品的供应充足和及时。

价格数据：蔬菜市场是存在竞争的，在本题中不存在竞争所以不需要采取价格措施。但在实际情况下监测蔬菜商品的市场价格和竞争对手的定价策略是很必要的。了解市场价格趋势和竞争对手的定价情况，可以帮助商超制定合理的蔬菜商品定价策略，以保持竞争力并最大化利润。

消费者偏好和反馈数据：在实际情况下，了解消费者的偏好和反馈对于市场营销的重要性不言而喻。了解消费者对不同蔬菜商品的偏好、购买习惯和反馈意见。这些数据可以帮助商超更好地了解消费者需求，进行产品调整和定位，提供更符合消费者期望的蔬菜商品，从而挖掘长久客户稳定利润。

以上数据可以通过销售记录、POS 系统、库存管理系统、供应商合作信息、市场调研、消费者调查等方式进行收集。这些数据将提供商超在制定补货和定价决策时的重要参考，帮助优化蔬菜商品的供应链管理、库存控制、定价策略以及市场定位，从而更好地满足消费者需求、提高销售效益和竞争力。

九、模型的评价

9.1 模型的优点

1.精确性：本模型利用数学方法对问题进行抽象和描述，精确地定义问题的变量、关系和约束条件。并没有引入一些不可控的变量，从而保障问题的求解具有精确性。

2.包容性：部分特殊点或异常点并不会影响模型的整体拟合趋势，从而保障模型在遇到存在异常点的数据时也能使用。

3.高效性：对于问题 2 的求解，只关注销售总量和定价本身，根据数据分布散点图构建线性关系，高效求解。对于问题 3 的求解，将复杂的问题分解，将可售单品总数控制在 27-33 个这一约束条件通过盈利情况简化为定量，从而只考虑各单品订购量满足最小陈列量 2.5 千克这一约束条件，将模型与问题 2 模型合并。

9.2 模型的缺点

1.线性模型的相关性较低：虽然用线性模型拟合销量和定价的关系得到的误差较小，且模型趋势与散点图大致一致，但是模型所表现出的相关性较低。

2.对于问题 2 考虑品类的销量和定价的关系，简单的将品类中的所有单品价格做了个加权平均，并没有太好的处理方法。

3.并没有考虑一些特殊情况对于蔬菜销售的影响，即理想化的模型放在现实中运用存在一定的问题。

十、参考文献

[1] 章斌权. 销售利润率计算公式应规范统一[J]. 工业会计, 2000 (9): 23-24.

[2] 罗超平, 王钊. 波动频率, 季节性上涨与蔬菜价格演进机理: 1978~ 2010 年[J]. 改革, 2012 (5): 94-100.

[3] 张川, 刘保政, 戢守峰, 等. 基于随机补货间隔和非线性变质率的易腐商品补货模型[J]. 东北大学学报 (自然科学版), 2012, 33(1): 141.

[4] 王骏, 王士同, 邓赵红. 聚类分析研究中的若干问题[D]. , 2012.

[5] 丁克良, 沈云中, 欧吉坤. 整体最小二乘法直线拟合[J]. 辽宁工程技术大学学报: 自然科学版, 2010 (1): 44-47.

[6] 樊嵘, 孟大志, 徐大舜. 统计相关性分析方法研究进展[J]. 数学建模及其应用, 2014, 3(1): 1-12.

[7] 高辰颖, 郭晓燕. 蔬菜销售的障碍因素及其对策分析[J]. 商情, 2013 (25): 81-81.

附录

附录 1：

Cluster 8: ['牛首生菜', '白菜苔', '蔡甸藜蒿', '本地上海青', '随州泡泡青', '外地茼蒿', '茼蒿(份)', '红薯尖(份)', '洪湖藕带', '红杭椒', '组合椒系列', '青尖椒(份)', '红杭椒(份)', '红尖椒(份)', '红椒(2)', '双孢菇', '虫草花(袋)', '鲜木耳(份)', '姬菇(份)', '海鲜菇(袋)(2)']

Cluster 16: ['四川红香椿', '萝卜叶', '枝江红菜苔(份)', '小青菜(2)', '鱼腥草(份)', '苋菜(份)', '小白菜(份)', '野藕(1)', '高瓜(2)', '净藕(3)', '紫茄子(1)', '红尖椒', '七彩椒(1)', '灯笼椒(1)', '红灯笼椒(1)', '青杭椒(份)', '水果辣椒(份)', '七彩椒(2)', '红灯笼椒(2)', '姬菇(1)', '海鲜菇(1)', '银耳(朵)', '鲜木耳(1)', '西峡香菇(2)', '金针菇(2)', '蟹味菇与白玉菇双拼(盒)', '西峡香菇(份) ', '蟹味菇(袋)', '白玉菇(2)']

Cluster 1: ['本地小毛白菜', '南瓜尖', '豌豆尖', '马齿苋', '本地菠菜', '黑油菜', '黄花菜', '快菜', '田七', '冰草', '紫苏', '薄荷叶', '蒲公英', '丝瓜尖', '芥菜', '大芥兰', '面条菜', '荠菜', '马兰头', '甘蓝叶', '洪山菜苔', '青菜苔', '鲜粽子叶', '艾蒿', '奶白菜苗', '菊花油菜', '双沟白菜', '洪山菜薹珍品手提袋', '洪山菜薹莲藕拼装礼盒', '冰草(盒)', '紫苏(份)', '襄甜红菜苔(袋)', '白蒿', '鱼腥草', '春菜', '槐花', '蔡甸藜蒿(份)', '红珊瑚(粗叶)', '红橡叶', '绿牛油', '芝麻苋菜', '鲜粽叶', '鲜粽叶(袋)(1)', '外地茼蒿(份)', '龙牙菜', '黄白菜(1)', '大白菜秧', '木耳菜(份)', '芥兰', '油菜苔', '紫贝菜', '鲜粽叶(袋)(2)', '鲜粽叶(袋)(3)', '紫白菜(1)', '紫白菜(2)', '藕', '菱角', '红莲藕带', '野生粉藕', '净藕(2)', '鲜藕带(袋)', '荸荠(份)', '洪湖莲藕(脆藕)', '野藕(2)', '藕尖', '紫圆茄', '花茄子', '青茄子(2)', '圆茄子(1)', '青杭椒(1)', '青杭椒(2)', '红线椒', '水果辣椒(橙色)', '芜湖青椒(2)', '小皱皮', '余干椒', '辣妹子', '紫尖椒', '紫螺丝椒', '水果辣椒', '七彩椒(份)', '灯笼椒(份)', '红灯笼椒(份)', '芜湖青椒(份)', '灯笼椒(2)', '红椒(份)', '青红尖椒组合装(份)', '白玉菇(1)', '蟹味菇(1)', '猴头菇', '鸡枞菌', '黑牛肝菌', '秀珍菇', '虫草花(盒)(1)', '茶树菇(袋)', '黑皮鸡枞菌', '赤松茸', '牛排菇', '活体银耳', '杏鲍菇(250克)', '赤松茸(盒)', '牛排菇(盒)', '猪肚菇(盒)', '黑牛肝菌(盒)', '黑皮鸡枞菌(盒)', '杏鲍菇(份)', '双孢菇(份)', '金针菇(份)', '鲜木耳(2)', '海鲜菇(2)', '姬菇(2)', '西峡花菇(2)', '花菇(一人份)', '菌菇火锅套餐(份)', '菌蔬四宝(份)', '鹿茸菇(盒)', '虫草花', '绣球菌', '绣球菌(袋)', '蟹味菇(2)', '白玉菇(盒)', '蟹味菇(盒)', '虫草花(盒)(2)', '和丰阳光海鲜菇(包)']

Cluster 14: ['苋菜', '菠菜', '红薯尖', '枝江红菜苔', '甜白菜', '小皱皮(份)', '金针菇(1)']

Cluster 6: ['云南生菜', '金针菇(盒)']

Cluster 5: ['竹叶菜', '上海青', '黄白菜(2)', '螺丝椒']

Cluster 7: ['小白菜', '木耳菜', '东门口小白菜', '菜心(份)', '黄心菜(2)', '荸荠', '姜蒜小米椒组合装(小份)', '海鲜菇(袋)(3)']

Cluster 4: ['牛首油菜', '茼蒿', '菜心', '小青菜(份)', '双孢菇(盒)']

Cluster 11: ['大白菜']

Cluster 10: ['云南油麦菜', '泡泡椒(精品)', '小米椒(份)']

Cluster 12: ['黄心菜(1)', '小青菜(1)', '上海青(份)', '青茄子(1)', '红椒(1)', '西峡花菇(1)', '白玉菇(袋)', '金针菇(袋)(2)']

Cluster 13: ['娃娃菜', '云南油麦菜(份)', '青梗散花', '螺丝椒(份)']

Cluster 18: ['奶白菜', '枝江青梗散花', '洪湖莲藕(粉藕)']

Cluster 17: ['本地黄心油菜', '竹叶菜(份)', '高瓜(1)', '大龙茄子', '圆茄子(2)', '青尖椒', '小米椒', '青红杭椒组合装(份)', '姬菇(包)', '海鲜菇(袋)(1)', '杏鲍菇(袋)', '海鲜菇(份)', '虫草花(份)', '杏鲍菇(2)', '海鲜菇(包)']
Cluster 15: ['云南生菜(份)', '紫茄子(2)']
Cluster 9: ['菠菜(份)', '保康高山大白菜', '奶白菜(份)']
Cluster 3: ['西兰花', '净藕(1)']
Cluster 20: ['莲蓬(个)', '长线茄', '青线椒(份)', '青线椒', '平菇', '杏鲍菇(1)', '金针菇(袋)(1)', '金针菇(袋)(3)', '海鲜菇(袋)(4)']
Cluster 19: ['芜湖青椒(1)']
Cluster 2: ['西峡香菇(1)']

附录 2：

k = [0.15808431902506123, 1.649484536082474, 0.2702049570300109, 0.5298438725963999, 0.17190339759208245, 0.23601851851851843, 0.032529319221967924, 0.4084680903946159, 0.03126923076923077, 0.16758918918918916, 0.8828262702122772, 0.19045662437297134, 0.5127462415863422, 0.16238095238095238, 0.08098035238477108, 0.4358747593940915, 0.0640209646890537, 0.06967629815745399, 0.34097893432465876, 0.5365257628814406, 0.2595997771380805, 0.8555665722379606, -1.914473990683231, -1.3540943979837314, -1.4406827936248134, -0.8046178999058382, 0.12181867321575096, -0.35090292792293687, 2.6264452716475803, -0.22742851609829987, 0.34583952196375284, -0.4246207479976524, 1.4765256335633001, 0.6858994378513421, 0.1453586741425176, -0.8190357515657618, -1.5921354166666268, 9.394453308311814, -5.012271062271069, -1.586047868220814, 0.12938636363636105, 0.19360477940583554, -0.42546886342278556, 5.331644893629805, 2.6195590847497705]

b = [0.020919478431030758, 2.7755575615628914e-17, 0.1370810614871094, -0.009097008221421121, 0.022780242484848556, 5.551115123125783e-17, 0.6969589054157137, 0.09559161823187545, -1.3877787807814457e-17, 0.18624468468468502, -0.4546325388918695, 0.008750221304219596, 0.11729091386232526, 2.7755575615628914e-17, 0.052419564436121946, -0.02768081010963286, 0.8398892238081876, 0.1592805695142372, 4.770003717472122, 0.010670485242622263, 4.428073637292236, 8.108317280453255, 19.522523291925477, 13.653430060717147, 18.948904054275232, 11.27271557408038, 1.2899621313801302, 7.884839884442773, 5.741980531044579, 6.402262336867576, 0.1669724548956193, 16.907102866733066, 6.187704586465079, -2.5112648344784407, 1.7255903782619102, 16.33479749478079, 25.60550624999975, -15.633733283337005, 40.64775274725278, 33.130644798234414, 2.09050000000006, 4.176339930119487, 20.390390696453444, 13.541514811052004, -6.28701969027148]

Q1 部分代码

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import spearmanr
from sklearn.preprocessing import MinMaxScaler

# 示例数据
DanDict1 = {'花叶类': 41546.11599999995, '花菜类': 7815.231000000082, '水生根茎类': 4794.958999999989, '茄类': 5862.958000000013, '辣椒类': 22913.315000000308, '食用菌': 14466.955000000005}
DanDict2 = {'花叶类': 55171.796000000104, '花菜类': 11890.340000000011, '水生根茎类': 8445.68600000003, '茄类': 8204.904999999972, '辣椒类': 21963.592000000124, '食用菌': 13720.533}
DanDict3 = {'花叶类': 52868.353000000105, '花菜类': 11375.210000000061, '水生根茎类': 12291.234000000055, '茄类': 3854.516999999997, '辣椒类': 20847.27400000011, '食用菌': 21753.308999999976}
DanDict4 = {'花叶类': 48934.71300000001, '花菜类': 10685.670000000046, '水生根茎类': 15049.474000000058, '茄类': 4509.402000000027, '辣椒类': 25864.44800000005, '食用菌': 26145.927999999996}
labels = ['花叶类', '花菜类', '水生根茎类', '茄类', '辣椒类', '食用菌']
x1 = list(DanDict1.values())
x2 = list(DanDict2.values())
x3 = list(DanDict3.values())
x4 = list(DanDict4.values())

data = np.empty((4, 6))

data[0] = x1
data[1] = x2
data[2] = x3
data[3] = x4
# 创建 MinMaxScaler 对象
scaler = MinMaxScaler()

# 对矩阵进行归一化
data = scaler.fit_transform(data)
print(data)

# 计算斯皮尔曼相关系数
corr_matrix, _ = spearmanr(data)

# 绘制热力图
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', square=True)
```

```
# 设置坐标轴刻度
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus']=False
plt.xticks(np.arange(len(labels)), labels)
plt.yticks(np.arange(len(labels)), labels)
plt.title('Pearson Correlation Heatmap')
plt.show()
```

Q2 部分代码：

```
# -*- coding: UTF-8 -*-
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt


#
# k=0.04376948174989643
# b=98.36940360870086

# k=0.04597297577687116
# b=4.978010415017721

# k=0.07046190490921798
# b=0.49696991230982945

# k=0.044701795073087414
# b=4.742678378904323

# k=0.012817619119099618
# b=58.42116238131479

k=0.03451442621778222
b=13.294241509373819

# 执行线性拟合
y_pred = []
for i in range(len(x)):
    y_pred.append(k*x[i]+b)

slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)

# 打印拟合结果
print("斜率（slope）：", slope)
print("截距（intercept）：", intercept)
```

```
print("相关系数（r_value）：", r_value)
print("p 值（p_value）：", p_value)
print("标准误差（std_err）：", std_err)

# 绘制折线图
plt.plot(x, y_pred, label='拟合曲线')

# 绘制散点
plt.scatter(x, y, color='red', label='真实点')

# 添加图例
plt.legend()

# 添加标题和坐标轴标签
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.title('Linear Equation: y = {}x + {}'.format(k, b))
plt.xlabel('定价（元/千克）')
plt.ylabel('销售总量（千克）')

# 显示图形
plt.show()
```

Q3 部分代码

```
# -*- coding: UTF-8 -*-
import pandas as pd

data1 = pd.read_excel(r"D:\sxjm\2 - 副本.xlsx")
data2 = pd.read_excel(r"D:\sxjm\3 - 副本.xlsx")
data3 = pd.read_excel(r"D:\sxjm\5.xlsx")
LiRun = {}
sold1 = {}
sold2 = {}
sold3 = {}
sold4 = {}
sold5 = {}
sold6 = {}
sold7 = {}


for i in range(len(data1["销售单价(元/千克)"])):
    a = data1["销售日期"][i].strftime('%Y-%m-%d %H:%M:%S')
    if data1["单品编码"][i] in LiRun:
        LiRun[data1["单品编码"][i]] += data1["销售单价(元/千克)"][i] * data1["销量(千克)"][i]
```

```
    if a[8:10] == "24":
        if data1["单品编码"][i] in sold1:
            sold1[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold1[data1["单品编码"][i]] = data1["销量(千克)"][i]
    elif a[8:10] == "25":
        if data1["单品编码"][i] in sold2:
            sold2[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold2[data1["单品编码"][i]] = data1["销量(千克)"][i]
    elif a[8:10] == "26":
        if data1["单品编码"][i] in sold3:
            sold3[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold3[data1["单品编码"][i]] = data1["销量(千克)"][i]
    elif a[8:10] == "27":
        if data1["单品编码"][i] in sold4:
            sold4[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold4[data1["单品编码"][i]] = data1["销量(千克)"][i]
    elif a[8:10] == "28":
        if data1["单品编码"][i] in sold5:
            sold5[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold5[data1["单品编码"][i]] = data1["销量(千克)"][i]
    if a[8:10] == "29":
        if data1["单品编码"][i] in sold6:
            sold6[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold6[data1["单品编码"][i]] = data1["销量(千克)"][i]
    else:
        if data1["单品编码"][i] in sold7:
            sold7[data1["单品编码"][i]] += data1["销量(千克)"][i]
        else:
            sold7[data1["单品编码"][i]] = data1["销量(千克)"][i]
else:
    LiRun[data1["单品编码"][i]] = data1["销售单价(元/千克)"][i] * data1["销量(千克)"][i]
    if a[8:10] == "24":
        sold1[data1["单品编码"][i]] = data1["销量(千克)"][i]
    elif a[8:10] == "25":
        sold2[data1["单品编码"][i]] = data1["销量(千克)"][i]
    elif a[8:10] == "26":
        sold3[data1["单品编码"][i]] = data1["销量(千克)"][i]
```

```
        elif a[8:10] == "27":
            sold4[data1["单品编码"][i]] = data1["销量(千克)"][i]
        elif a[8:10] == "28":
            sold5[data1["单品编码"][i]] = data1["销量(千克)"][i]
        if a[8:10] == "29":
            sold6[data1["单品编码"][i]] = data1["销量(千克)"][i]
        else:
            sold7[data1["单品编码"][i]] = data1["销量(千克)"][i]

sunhao = {}
print(data3)
for i in range(len(data3["单品编码"])):
    sunhao[data3["单品编码"][i]] = data3["损耗率(%)"][i]

for key, value in sold1.items():
    if value <= 2.5 * (1-sunhao[key]):
        sold1[key] = 2.5
    else:
        sold1[key] = value / (1-sunhao[key])

for key, value in sold2.items():
    if value <= 2.5 * (1-sunhao[key]):
        sold2[key] = 2.5
    else:
        sold2[key] = value / (1-sunhao[key])

for key, value in sold3.items():
    if value <= 2.5 * (1-sunhao[key]):
        sold3[key] = 2.5
    else:
        sold3[key] = value / (1-sunhao[key])

for key, value in sold4.items():
    if value <= 2.5 * (1-sunhao[key]):
        sold4[key] = 2.5
    else:
            sold4[key] = value / (1-sunhao[key])

    for key, value in sold5.items():
        if value <= 2.5 * (1-sunhao[key]):
            sold5[key] = 2.5
        else:
            sold5[key] = value / (1-sunhao[key])

```

```
    for key, value in sold6.items():
        if value <= 2.5 * (1-sunhao[key]):
            sold6[key] = 2.5
        else:
            sold6[key] = value / (1-sunhao[key])

    for key, value in sold7.items():
        if value <= 2.5 * (1-sunhao[key]):
            sold7[key] = 2.5
        else:
            sold7[key] = value / (1-sunhao[key])

    for i in range(len(data2["日期"])):
        a = data2["日期"][i].strftime('%Y-%m-%d %H:%M:%S')
        if data2["单品编码"][i] in LiRun:
            if a[8:10] == "24":
                if data2["单品编码"][i] in sold1:
                                LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold1[data2["单品编码"][i]]
                else:
                    max_value = float('-inf')
                    for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
                        if data2["单品编码"][i] in dictionary:
                            value = dictionary[data2["单品编码"][i]]
                            if value > max_value:
                                max_value = value
                                LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value
            elif a[8:10] == "25":
                if data2["单品编码"][i] in sold2:
                                LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold2[data2["单品编码"][i]]
                else:
                    max_value = float('-inf')
                    for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
                        if data2["单品编码"][i] in dictionary:
                            value = dictionary[data2["单品编码"][i]]
                            if value > max_value:
                                max_value = value
                                LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value
            elif a[8:10] == "26":
                if data2["单品编码"][i] in sold3:
```

```
                    LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold3[data2["单品编码"][i]]
            else:
                max_value = float('-inf')
                for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
                    if data2["单品编码"][i] in dictionary:
                        value = dictionary[data2["单品编码"][i]]
                        if value > max_value:
                            max_value = value
                            LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value
        elif a[8:10] == "27":
            if data2["单品编码"][i] in sold4:
                    LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold4[data2["单品编码"][i]]
            else:
                max_value = float('-inf')
                for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
                    if data2["单品编码"][i] in dictionary:
                        value = dictionary[data2["单品编码"][i]]
                        if value > max_value:
                            max_value = value
                            LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value
        elif a[8:10] == "28":
            if data2["单品编码"][i] in sold5:
                    LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold5[data2["单品编码"][i]]
            else:
                max_value = float('-inf')
                for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
                    if data2["单品编码"][i] in dictionary:
                        value = dictionary[data2["单品编码"][i]]
                        if value > max_value:
                            max_value = value
                            LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value
        if a[8:10] == "29":
            if data2["单品编码"][i] in sold6:
                    LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold6[data2["单品编码"][i]]
            else:
                max_value = float('-inf')
                for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
```

```
                    if data2["单品编码"][i] in dictionary:
                        value = dictionary[data2["单品编码"][i]]
                        if value > max_value:
                            max_value = value
                            LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value
            else:
                if data2["单品编码"][i] in sold7:
                            LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * sold7[data2["单品编码"][i]]
                else:
                    max_value = float('-inf')
                    for dictionary in [sold1, sold2, sold3, sold4, sold5, sold6, sold7]:
                        if data2["单品编码"][i] in dictionary:
                            value = dictionary[data2["单品编码"][i]]
                            if value > max_value:
                                max_value = value
                                LiRun[data2["单品编码"][i]] -= data2["批发价格(元/千克)"][i] * max_value

    print(LiRun)

    sorted_dict = sorted(LiRun.items(), key=lambda x: x[1])
    print(sorted_dict)
```