基于 K-Means 聚类和 ARIMA 模型的定价与补货决策

吴阳诚、莫力炬、张航

2023 年 9 月 10 日

摘要

本文目的是分析历史数据的规律，从而建立蔬菜类商品的自动定价与补货决策。

【背景】生鲜商超需要结合历史数据，得到市场需求及进货价格的规律，从而在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**，并确定“**成本加成定价**”，最终达到**总利润最大化**的目的。

【方法】本文主要采用了 **K-Means 聚类**、**关系计数矩阵**、**控制变量**、**ARIMA 时间序列**、**0-1 规划**等方法，从高维度、与时间高度相关的历史流水数据中分析规律，并结合**供给侧和需求侧**进行逻辑分析，最终构建模型最大化总利润。

【分析与结果】

问题一：一方面，运用 Excel，统计品类及单品的累计销量，结合蔬菜的时令性，分成不同季度进行统计，并计算方差等指标，发现它们的销量分布是不均匀的，各品类均存在销量突出的单品。另一方面，从原数据中挖掘出各个顾客的购买信息，采用“关系计数矩阵”的新方法，得到各品类间的**“互补性”**与**“替代性”**的关系；针对同一品类中各单品，通过 K-Means 聚类，分析各单品间的**相似程度**，进而得出“互补性”与“替代性”的关系。

问题二：第一小问中，采取控制变量的方法，设置总销量、进价的波动阈值，确保模型数据具有所在时间段上“供需环境”的稳定性，再采用 MLP 多元线性回归模型 $KSW = \beta_0 + \beta_1 \frac{1}{P} + \beta_2 P + \beta_3 T_m + \varepsilon$，得到各品类的销售总量与成本加成定价的关系，拟合程度（R^2 平均达到 90%）极佳。

第二小问中，首先通过 **ARIMA 时间序列**预测出各品类未来一周的销量以及进货价的置信区间，再结合第一小问得到的销售总量与成本加成定价的关系，最终构建非线性规划模型，得出总利润最大化的各品类定价与补货策略。

问题三：找出有供给的单品集合后，通过**时间序列和聚类综合赋权**，对可选单品集合进一步优选，再设置 0-1 逻辑变量，结合题设约束构建 0-1 规划模型或多目标规划模型，得出最大化总利润的单品补货量和定价策略。

问题四：结合“供需关系”的逻辑分析及前几问的模型构建，应当收集更多供应商信息，构建客户的反馈渠道等等，从而更好地制定蔬菜商品的补货和定价决策。

【评价与改进】本文中的有“关系计数矩阵”、时间序列和聚类综合赋权等的创新点，不过，为了操作的易行度，部分处理的精细程度尚有提升空间。

关键词：互补性与替代性 K-Means 聚类 MLP 模型 ARIMA 模型 0-1 规划

1 问题重述

生鲜商超中的蔬菜类产品往往保鲜期较短，若某类蔬菜当日无法卖完，则对商家来说会造成亏损。为了尽量避免这种情况的出现，并获得尽可能高的利润，商家往往选择在凌晨进货。但这也导致了商家进货前无法知道具体的单品与确切的成本价。同时，商家如何采用“成本加成定价”法进行合理定价取得最高利润的问题，也亟待解决。因此，尝试通过进货价格分析与市场需求分析进行科学合理的进货决策与定价决策。

根据以上背景及四个所给附件，试解决以下问题：

1.1 问题一

蔬菜各品类及单品之间可能存在一定联系，尝试分析蔬菜各品类与单品间销售量的分布规律与相互关系。

1.2 问题二

现以品类为单位进货，欲建立模型得到各蔬菜品类的销售总量与成本加成定价间的关系；同时通过进货价格分析与市场需求分析，进行对未来一周日补货总量与定价策略的科学制定，从而获得最大收益。

1.3 问题三

由于蔬菜类商品销售空间有限，欲制定单品补货计划，并且控制单品总数在 27-33 个，各单品订购量满足大于等于 2.5 千克。现要求根据 2023 年 6 月 24 日-33 日的可售品种，给出 7 月 1 日的单品补货量和定价策略，尽可能满足市场需求，从而获得最大收益。

1.4 问题四

试提出收集其他的相关数据使得超商能更好地进行蔬菜商品的进货决策与定价决策，并给出理由。

2 问题分析

2.1 问题一的分析

题目要求分析蔬菜各品类与单品间销售量的分布规律与关系。运用附件 1、2。对于销售量在各品类与各单品间的分布规律，从供给侧视角看，考虑到蔬菜具有时令性，用 Excel 得到各个季节各品类销售量占总销售量之比，以及各单品销售量占其所在品类销售量之比，从而反映题目所要求的分布规律，同时也体现了各品类与单品的受欢迎程度。对于各品类及单品销售量间的关系，查阅相关资料，有充分理由假设总体具有如下关系：各品类间蔬菜属于**互补**商品，即顾客为烹饪，会购买不同品类的蔬菜，故**部分品类间蔬菜销售量互补性强，替代性弱**；同品类中的各单品则属于**替代**商品，即顾客较少购买同品类的不同单品，故**同品类中的部分单品销售量互补性弱，销售型强**。从顾客视角出发，将极短时间内的多次购买记录视作同一顾客所为。于是建

立关系计数矩阵，对同一顾客同时消费的品类进行计数，从而体现品类间的关系。在用 Python 对各个顾客的行为作出统计分析后，得到各品类与单品间的关系，尝试验证假设。

2.2 问题二的分析

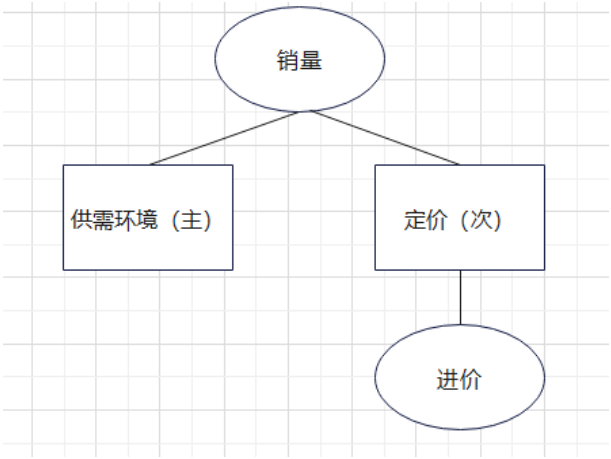

图 1: 问题二主要思路

主要思路如上图所示：题目要求得到各蔬菜品类的销售总量与成本加成定价间的关系，同时要求制定未来一周的日补货总量与定价策略，以期获得最大收益。运用附件 1、2、3。以品类为单位进货，首先运用 Excel 将各周内各品类中单品的销量累加，作为该品类的销售总量，再建立算法得到各周内各品类定价的均价，采取控制变量的方法，确保模型数据具有所在时间段上“供需环境”的稳定性，再采用 **MLP 多元线性回归模型**，从而得到各蔬菜品类的销售总量与成本加成定价间的关系。为制定未来一周的日补货总量与定价策略，以需求侧的视角考虑，可利用 Python 对各品类蔬菜过去销量总量数据及定价均价进行分析，利用时间序列预测未来一周顾客对各品类蔬菜的需求与定价预期。以获得最大收益为**目标函数**，以由过去数据所预测的各品类成本及销量与定价的关系为**约束条件**，建立目标规划模型，并给出具体可行的算法。

2.3 问题三的分析

题目要求在满足对各单品数量、订购量、可售种类的限制下，制定 7 月 1 日的日各单品补货量与定价策略，以期获得最大收益。类似于问题二，将利用 Python 预测可售单品的当日成本，并且得到各可售单品的销售总量与成本加成定价间的关系。由于数据量庞大，利用时间序列预测法与聚类法得到各单品的权重，并取权重高的部分单品存入全指标组集合。然后采用 0-1 规划的思想，以尽量取得最大收益为**目标函数**，以题中所给对单品补货的各项约束、销售总量与成本加成定价间的关系等为**约束条件**建立目标规划模型，并给出具体可行的算法。

2.4 问题四的分析

题目要求提出其他的相关数据关数据使得超商能更好地进行蔬菜商品的进货决策与定价决策，并给出理由。分别从供给侧视角与需求侧视角考虑，思考相关数据如何对进货决策与定价

决策产生影响，从而获得更高的收益。

3 模型假设

1. 假设商超每日补货的各个单品都有售出。
2. 由于不知道确切补货量，假设补货菜品全部售出。
3. 为确定相近时间内的消费同属一个顾客，假设该商超仅有一个收银台。
4. 假定 2023 年 6 月 24 日-33 日的可售品种即为 7 月 1 日可售品种。

4 符号及变量说明

4.1 符号及变量说明

变量	含义	单位
K_i	第 i 个蔬菜品类（i=1,2,3,4,5,6）	
J_i	i 品类涵盖的单品总数	个
g_{ij}	i 品类下第 j 个单品（$j \in \{1,2,\ldots,J_i\}$）	
$PD^{(n)}_{y_{m\,d}}$	以 y 年 m 月 d 日为中心的连续 n 天 $y \in \{20,21,22,23\}$ $m \in \{1,2,\ldots,12\}$ $d \in \{1,2,\ldots,31\}$	
T_m	具体年份月份 $T_m \in \{1,2,\ldots,12\}$ $m \in \{1,2,\ldots,37\}$	
S_n	具体年份季度 $S_n \in \{1,2,3,4\}$ $n \in \{1,2,\ldots,13\}$	
gt_{ij}	该单品上市的月份	月
KT_i	该品类上市的月份	月
$gsw_{ij}^{(t)}$	该单品在 t 月份下的销量 $t \in \{1,2,\ldots,12\}$	千克
$KSW_i^{(t)}$	该品类在 t 月份下的销量 $t \in \{1,2,\ldots,12\}$	千克
$sgsw_{ij}^{(s)}$	该单品在 s 季度下的销量 $s \in \{1,2,3,4\}$	千克
$SKSW_i^{(s)}$	该品类在 s 季度下的销量 $s \in \{1,2,3,4\}$	千克
$gn_{ij}^{(t)}$	单品在 t 月份下的需求程度 $t \in \{1,2,\ldots,12\}$	千克
$KN_i^{(t)}$	大类在 t 月份下的需求程度 $t \in \{1,2,\ldots,12\}$	千克
RI	关系计数矩阵	
A	判断矩阵	
a_{ij}	判断矩阵的元素	
ri_{ij}	顾客同时购买 K_i, K_j 品类的次数 $(i \neq j)$	次
ri_{ii}	K_i 类总共被购买的次数	次
$p_{ij}^{(t)}$	该单品在 t 月份下的定价 $t \in \{1,2,\ldots,12\}$	元
$P_i^{(t)}$	该品类在 t 月份下的等效定价 $t \in \{1,2,\ldots,12\}$	元
$r_{ij}^{(t)}$	该单品在 t 月份下的损耗率 $t \in \{1,2,\ldots,12\}$	元
$R_i^{(t)}$	该品类在 t 月份下的等效损耗率 $t \in \{1,2,\ldots,12\}$	元
$d_{ij}^{(t)}$	该单品在 t 月的折后价与原价之比 $t \in \{1,2,\ldots,12\}$	
$D_i^{(t)}$	该品类在 t 月的等效折后价与原价之比 $t \in \{1,2,\ldots,12\}$	
$C_i^{(t)}$	某品类在 t 月的进货总成本 $t \in \{1,2,\ldots,12\}$	元
$S_i^{(t)}$	某品类在 t 月的总销售额 $t \in \{1,2,\ldots,12\}$	元
$Pro_i^{(t)}$	该品类 t 月的总利润 $t \in \{1,2,\ldots,12\}$	元
$RPro_i^{(t)}$	该品类 t 月的单位折损利润率 $t \in \{1,2,\ldots,12\}$	
Ω	全体指标对 (i,j) 集合	
x_{ij}	是否选购该单品 $x_{ij} = 0$ *or* 1	

表 2: 符号及变量说明

5 模型建立

5.1 问题一的模型建立

5.1.1 品类间

对于问题一，首先构建如下思维导图，以方便对问题的研究：

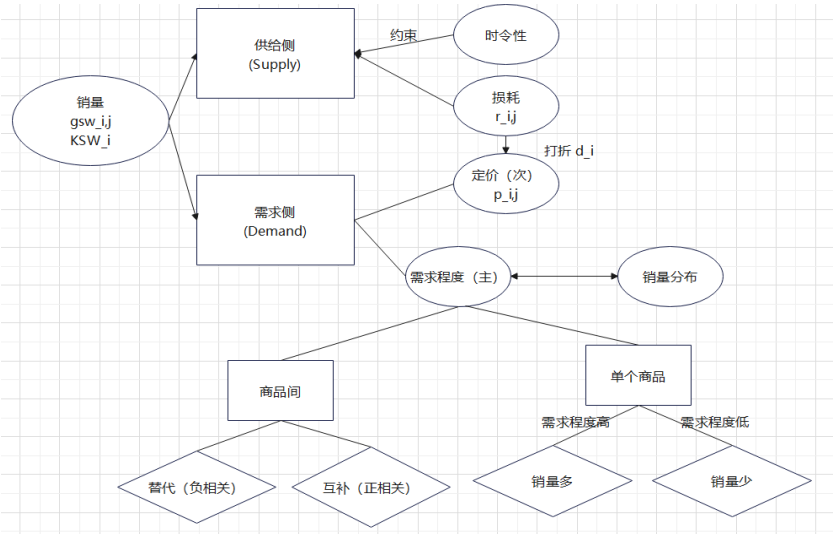

图 2: 问题一思维导图

如图 1 所示，认为影响销量的因素分为供给侧因素和需求侧因素，定价作为次要因素影响着销量，而需求程度则作为主要因素对销量分布起决定性作用。显然，单个商品的需求程度高低影响其销量的多少；而商品与商品间则存在着替代与互补的性质，对销量分布与商品间关系的研究具有重大意义。

欲分析蔬菜各品类及单品销售量的分布规律及相互关系，考虑到品类数量较少，可直接由 Excel 公式得到历年来各品类蔬菜总销售量及各季度销售量，记为 $SKSW_i^{(s)}$，再得出各销售量所占该时间段总销售量的比值，并作图如下：

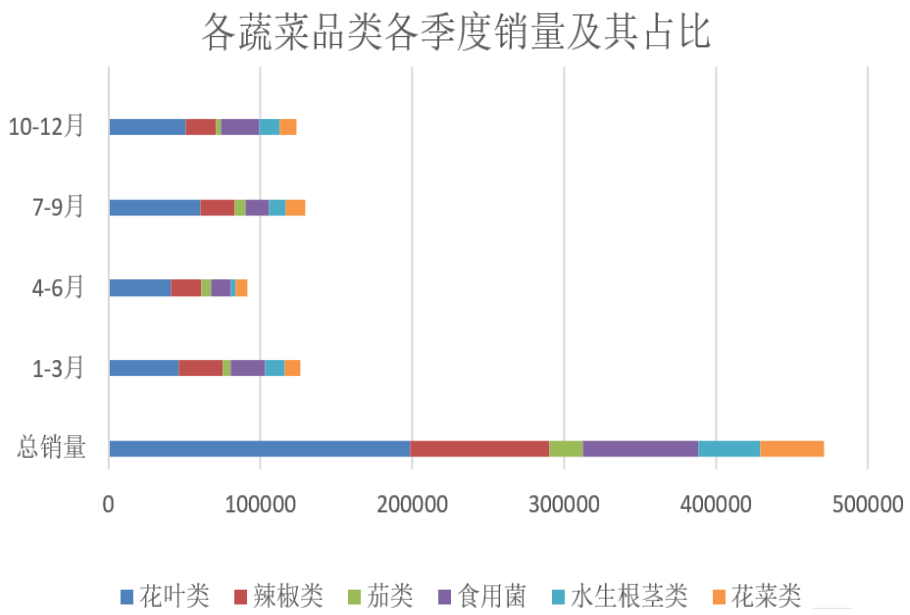

图 3: 各蔬菜品类各季度销量及其占比

根据此图，显然可以得到各品类有以下分布规律：

1. 无论何时，花叶类蔬菜的销量占比都是最高，故花叶类蔬菜的总体市场需求最高。
2. 花菜类蔬菜的总销售量占比最小，茄类次之，故这两类蔬菜的总体市场需求并不高。
3. 四个季度中，4-6 月总蔬菜品类销量占比最小，7-9 月总蔬菜品类销量占比最大。季节变化对各蔬菜品类销量影响较大。
4. 各蔬菜品类的销量随季节改变的变化趋势相似，即大致呈正相关。

以上各蔬菜品类销售量的分布规律能在需求侧直观反映各品类的市场需求差别；从供给侧看，由于蔬菜具有时令性，不同季节供货量不同，也是影响各品类销售量的重要因素。而对于规律 4，无法否定问题分析中对品类间销售量关系的假设。因此引入关系计数矩阵 RI。建立步骤如下：

1. 由于模型假设该商超只有一个收银台，故极短时间内的连续购买记录视作同一顾客所为，取定 3 秒为判定标准。若顾客购买记录中出现了 K_i 品类蔬菜，则对计数器 ri_{ii} 加一。在此基础上，若出现一个顾客同时购买多品类蔬菜的情况，则进行下一步操作，如: 若一个顾客同时购买 $K_1K_2K_3$ 品类，则对计数器 $ri_{12}ri_{21}ri_{23}ri_{32}ri_{31}ri_{13}$ 分别加一
2. 将 6×6 的计数结果填入计数矩阵。
3. 计数结果的结果体现了各蔬菜品类的相关性。即 ri_{ij} 数值越大，K_i 与 K_j 的互补性越强，替代性越弱，反之则反。但前提是证明该矩阵的合理性。

4. 两两作比得到判断矩阵 A，其中元素 a_{ij} 进行如下操作：$\begin{cases} a_i^j = \frac{ri_{ij}}{ri_{ii}} & i < j \\ a_i^j = 1 & i = j \\ a_i^j = \frac{1}{a_{ji}} & i > j \end{cases}$，从而可以检验矩阵 A 的一致性，最终证明“替代与互补”的自洽性。

处理过后得到的矩阵如下：

1	0.243992	0.05181	0.124933	0.07809	0.054112
4.098496	1	0.050157	0.119279	0.076254	0.041298
19.30124	19.93737	1	0.148525	0.092138	0.039444
8.004302	8.38371	6.732859	1	0.090201	0.03993
12.80577	13.11412	10.85328	11.08634	1	0.046647
18.48024	24.21408	25.35227	25.04382	21.43773	1

图 4: 判断矩阵 A

显然，矩阵 A 具有一致性，由各元素的大小也进一步得出以下相互关系：

1. 花叶类植物与辣椒类植物间互补性最强，替代性最弱；茄类植物与水生根茎类、食用菌类植物间的互补性最弱，替代性最强。
2. 食用菌类与辣椒类、水生根茎类、花叶类植物间均有较强互补性，较弱替代性。

5.1.2 单品间

品类间与单品间的核心处理思想一致，但由于单品的数量庞大，作图制表的效率较低。因此选择采用 **K-Means 聚类算法**，把样本划分为若干类别，使得同类样本间的相似度高，不同类样本间的相似度低，样本数减少后，再对其作与品类相似的处理，复杂度将大大降低。以下是利用 K-Means 聚类算法减少样本数的过程：

1. 预处理，将数据标准化并过滤异常点。
2. 随机选取 K 个（此处取 $K = 3$）中心，即选择迭代寻找 K 个簇，分别记为 $\mu_1^{(0)}, \mu_2^{(0)}, \mu_3^{(0)}$。
3. 定义损失函数

   $J(c, \mu) = \min \sum_{i=1}^{M} \parallel x_i - \mu_{c_i} \parallel^2$

   其中 x_i 代表第 i 个样本,c_i 是 x_i 所属簇,μ_{c_i} 代表簇对应的中心点,M 是样本总数。
4. 开始迭代过程。首先将 x_i 分配至距其最近的中心：

   $c_i^t \leftarrow \arg\min k \parallel x_i - \mu_k^t \parallel^2$

   再更新各中心：

   $\mu_k^{(t)} \leftarrow \arg min_{\mu} \sum_{i:c_i^t=k}^{b} \parallel x_i - \mu \parallel^2$

5. 重复前一步骤，以达到减小损失的目的，J 单调递减直到极小值，中心点和样本划分的类别同时收敛。

将六品类蔬菜分别进行如上 K-Means 聚类算法。以样本数量最多的花叶类蔬菜为例，得到以下三维聚类图，其中每个样本点的三个维度分别是销量、定价与顾客购买量：

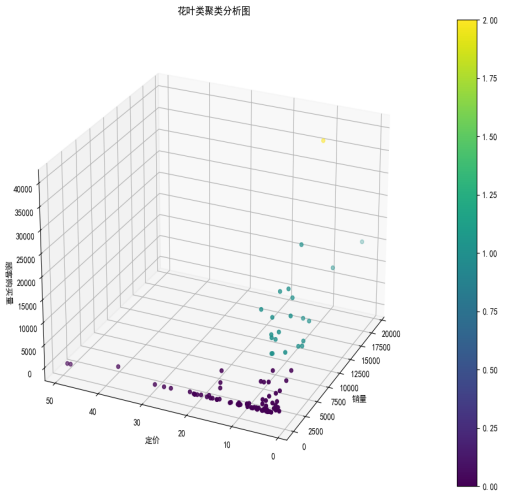

图 5: 花叶类蔬菜的聚类图

以聚类结果为样本，重复品类间的操作，可以得到以下分布规律与相互关系：

1. 花叶类蔬菜的销售量集中在大白菜、云南生菜以及含于其所在簇的蔬菜。
2. 各单品销售量随季节变动的变化明显，且由于相关性较高，大部分单品随季节变化的规律近似。
3. 聚类结束后同簇的单品显示出较高的相似性，即较高的替代性，较低的互补性。
4. 聚类结束后离中心点较远的点所代表的单品与其余单品显示出较高的互补性，较低的替代性。

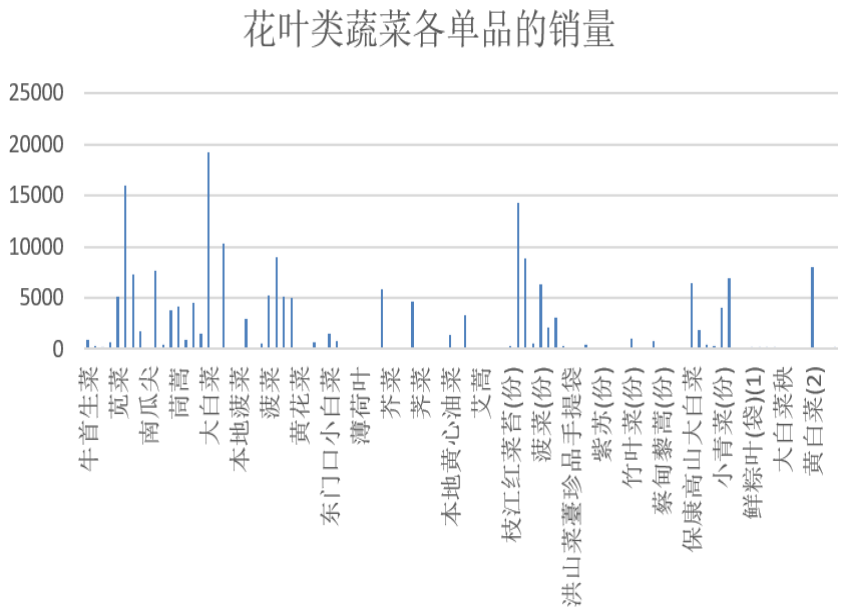

图 6: 柱状图表明销售量集中在少数单品

对于分布规律，可由 Excel 分析数据直观看出；对于相互关系，挑出小部分单品，生成判断矩阵以检验结论正确性。对六个品类蔬菜重复操作，推广普遍相互关系。

对于前文对相互关系的假设，我们的确发现了蔬菜间“替代性与互补性”的关系，但由于品类间的功能不适配、品类内单品间差异性较大等原因，假设在细节上显得过于理想，但从数据总体上看仍具有普遍意义。

5.2 问题二的模型建立

5.2.1 销售总量与定价关系

题目要求以蔬菜品类为单位分析各品类销售总量与成本加成定价的关系。认为影响销量的因素主要分为**供需关系**与**定价**，由于题目要求品类销售总量与成本加成定价的关系，故需对变量供需环境进行控制，即以定价为自变量，销量为因变量。将大量数据压缩成以日为单位的单品销售量与定价数据 $p_{ij}^{(d)}$，所需品类销售量可简单通过累加该日单品销售量得到；为得到等效品类定价，需将大量数据压缩成以日为单位的品类定价 $P_i^{(d)}$，选择利用如下公式得到均值：

$$P_i^{(d)} = \frac{\sum_j p_{ij}^{(d)}}{J_i} \tag{1}$$

观察图六花叶类回归分析图，对于销量与定价的相互关系，合理猜测应遵循“薄利多销”原则，即销量与定价宏观上应呈现负相关。为防止不规则变动因素导致的突变影响结论的给出，引入两个约束条件：

1. 销量变动率设定阈值 e_1 应在 ± 0.1
2. 进价变动率设定阈值 e_2 应在 ± 0.01

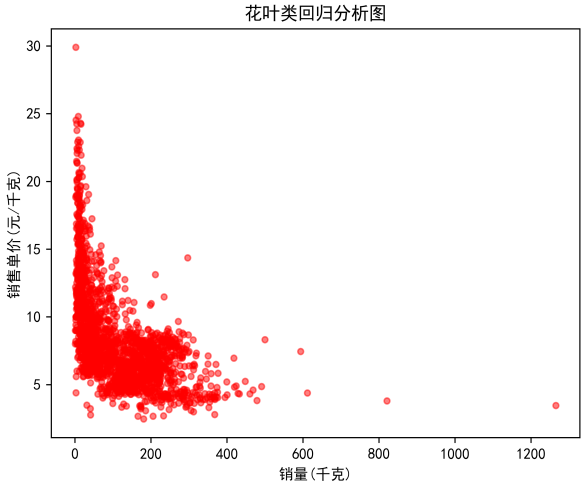

图 7: 花叶类回归分析

为方便后续的预测操作，可以得到以 y 年 m 月 d 日为对称点的连续十四天的等效定价 $P_{ij}^{PD_{y,m,d}^{(14)}}$。用 Python 随机生成多个天数，得到以这些天数为对称点的多个等效定价 $P_{ij}^{D_{ymd}^{+14}}$，利用多个数据点进行多元线性回归拟合关于销量和定价的关系曲线，并最终得到各品类销量关于定价的函数关系。以下是线性回归的拟合举例：

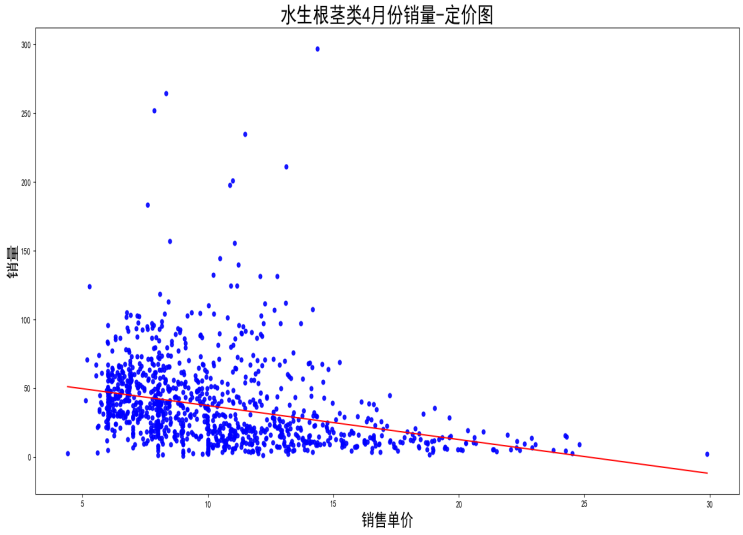

图 8: 水生根茎类植物 4 月份销量-定价线性回归拟合

对于多元线性回归，首先建立如下模型：

$$KSW_i^{(t)} = \beta_0 + \beta_1 \frac{1}{P_i} + \beta_2 P_i + \beta_3 T_m + \varepsilon_i \tag{2}$$

其中 β_0 是截距 ε 是误差项，表示模型不能完美地解释因变量的所有变异。目标是估计模型中的系数 β_1, β_2, 使得模型可以最好地拟合观测数据。使用最小二乘法来估计这些参数，该方法的目标是最小化观测数据与模型预测值之间的平方误差的总和。一旦估计出参数，就可以使用模型进行拟合和预测。通过将自变量的值代入模型，可以得到对因变量的预测值。利用以下公式大致求得品类的销售总量与成本加成定价的关系。

$$KSW_i^{(t)} = \beta_0 + \beta_1 \frac{1}{\hat{P}_i} + \beta_2 \hat{P}_i + \beta_3 \times 7 \tag{3}$$

具体关系如下表所示：

蔬菜品类	KSW
水生根茎类	$-79.436100700200361/x - 3.115095396199161x + 78.48592499275213$
花叶类	$687.61800054520091/x7.7066354009683575x + 17.966695518935346$
花菜类	$-58.111530136465771/x - 3.3396881238903813x + 77.0663516292102$
茄类	$110.666629886466131/x - 0.06931194215921938x + 7.948933682178499$
辣椒类	$659.49438429653321/x3.664734069815529x + -30.917672996846708$
食用菌	$-676.55274887915681/x - 11.949000425482215x + 274.50620576931027$

表 3: 各蔬菜品类销售总量与成本加成定价间的关系.

5.2.2 未来一周日补货量与定价策略

以周为单位,确定未来一周品类日补货量与定价策略,目标是获得最大利润。周利润 $Pro_i^{D_{ymd+3}^{(7)}}$ 由周总成本 $C_i^{D_{ymd+3}^{(7)}}$、周等效定价 $P_i^{D_{ymd+3}^{(7)}}$、周进货量 $M_i^{D_{ymd+3}^{(7)}}$、周进货价 $B_i^{D_{ymd+3}^{(7)}}$、周销售量 $N_i^{D_{ymd+3}^{(7)}}$、等效损耗率 $R_i^{D_{ymd+3}^{(7)}}$、等效折后价与原价之比 $D_i^{D_{ymd+3}^{(7)}}$ 共同决定。（本题接下来均以周为单位，为方便阅读，接下来省略角标 $D_{ymd+3}^{(7)}$）可以得到：

$$Pro_i = N_i \left[(1 - R_i)P_i + R_i(P_i \times D_i)\right] - B_i M_i \qquad (M_i \geq N_i) \tag{4}$$

$$RPro_i = \frac{Pro_i}{C_i} \tag{5}$$

品类周进货价等等效值同样通过求均值的方法得到历史数据。式中周销售量与周进货价需要通过 ARIMA 时间序列模型预测得到。

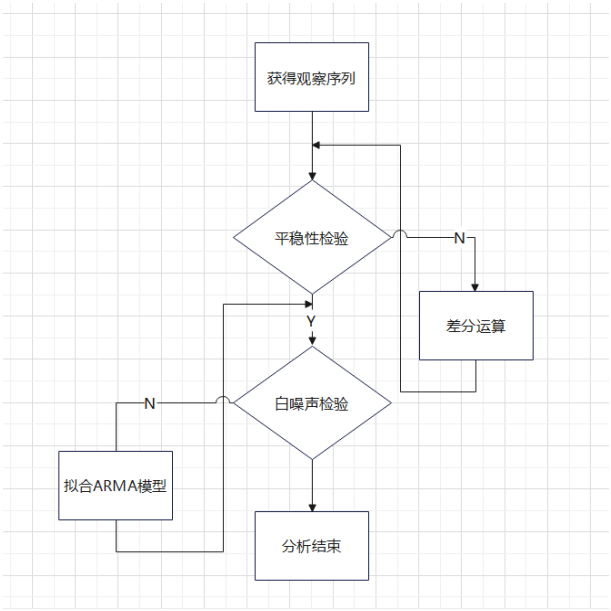

图 9: 差分平稳时间序列建模步骤

以下步骤进行差分平稳时间序列建模

1. 首先计算周销量 N_i（记为随机变量 X）与周进货量 B_i（记为随机变量 Y）的均值 μ 与方差 δ^2。由时序图检验法知，若均值 μ 与方差 δ^2 是常值，则进行白噪声检测；反之重复进行差分运算直至通过平稳性检验。

2. 进行白噪声检验，利用残差序列检验是否白噪声。使用 Ljung-Box 检验或 Q 统计量来定量检验残差序列是否是白噪声。这些检验基于残差序列的自相关性，它们的零假设序列是白噪声。如果 p 值显著大于显著性水平 0.05，则不能拒绝零假设，表明序列是白噪声。

3. 若非白噪声，则进行拟合 ARMA 模型。p 阶自回归模型与 q 阶移动平均模型如下：

$$x_t = \phi_0 + \phi_1 x_{t-1} +, \ldots, +\phi_p x_{t-p} + \varepsilon_t \tag{6}$$

$$x_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} -, \ldots, -\theta_q \varepsilon_{t-q} \tag{7}$$

于是自回归移动平均模型如下：

$$x_t = \phi_0 + \phi_1 x_{t-1} +, \ldots, +\phi_p x_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} -, \ldots, -\theta_q \varepsilon_{t-q} \tag{8}$$

拟合 ARMA 模型后返回白噪声检验。

代入已有历史数据至 ARMA 模型后，利用 Python 得到置信区间 α。根据所得的置信区间，N_i 与 B_i 的预测结果应在如下区间：

$$\begin{cases} \hat{N}_i \in [0, b_{i_1}] \\ \hat{B}_i \in [0, b_{i_2}] \end{cases} \tag{9}$$

对于等效折后价与原价的比值 D_i，定义 A_{i_1} 与 A_{i_2} 分别作为 D_i 中的打折单品集合与未打折单品集合，则有如下关系得到 D_i：

$$D_i = \frac{\frac{1}{|A_{i_1}|}\sum_{(i,j)\in A_{i_1}} p_{ij}}{\frac{1}{|A_{i_2}|}\sum_{(i,j)\in A_{i_2}} p_{ij}} \tag{10}$$

其中数据均从三年的时间范围得到。

当下，我们欲以商超获得最大收益为目标函数：

$$\max \qquad Pro = \sum_{i=1}^{6}\left\{\hat{N}_i \times [(1-R_i)\times P_i + R_i \times (P_i \times D_i)] - \hat{B}_i \times M_i\right\} \tag{11}$$

以 N_i, P_i 间的关系、各参数的预测方法及结果为约束条件建立非线性规划模型并进行优化。

$$\begin{cases} M_i \geq N_i \\ \hat{N}_i \in [a_{i_1}, b_{i_1}] \\ \hat{B}_i \in [a_{i_2}, b_{i_2}] \\ P_i = f(\hat{N}_i) \\ D_i = \frac{\frac{1}{|A_{i_1}|}\sum_{(i,j)\in A_{i_1}} p_{ij}}{\frac{1}{|A_{i_2}|}\sum_{(i,j)\in A_{i_2}} p_{ij}} \end{cases} \tag{12}$$

即完整非线性规划模型如下：

$$\max \qquad Pro = \sum_{i=1}^{6}\left\{\hat{N}_i \times [(1-R_i)\times P_i + R_i \times (P_i \times D_i)] - \hat{B}_i \times M_i\right\}$$

$$s.t. \begin{cases} M_i \geq N_i \\ \hat{N}_i \in [a_{i_1}, b_{i_1}] \\ \hat{B}_i \in [a_{i_2}, b_{i_2}] \\ P_i = f(\hat{N}_i) \\ D_i = \frac{\frac{1}{|A_{i_1}|}\sum_{(i,j)\in A_{i_1}} p_{ij}}{\frac{1}{|A_{i_2}|}\sum_{(i,j)\in A_{i_2}} p_{ij}} \end{cases} \tag{13}$$

优化思想参考“序贯算法”，即按照重要性主次赋予各参数优先因子。首先以 M_i 最优为目标优化，找寻周进货量最优情况；由于 N_i, P_i 存在一定函数关系，故再调节 N_i, P_i 的值，将周等效定价优化至最优情况。最终得到的下周日补货总量和定价策略如下：

5.3 问题三的模型建立

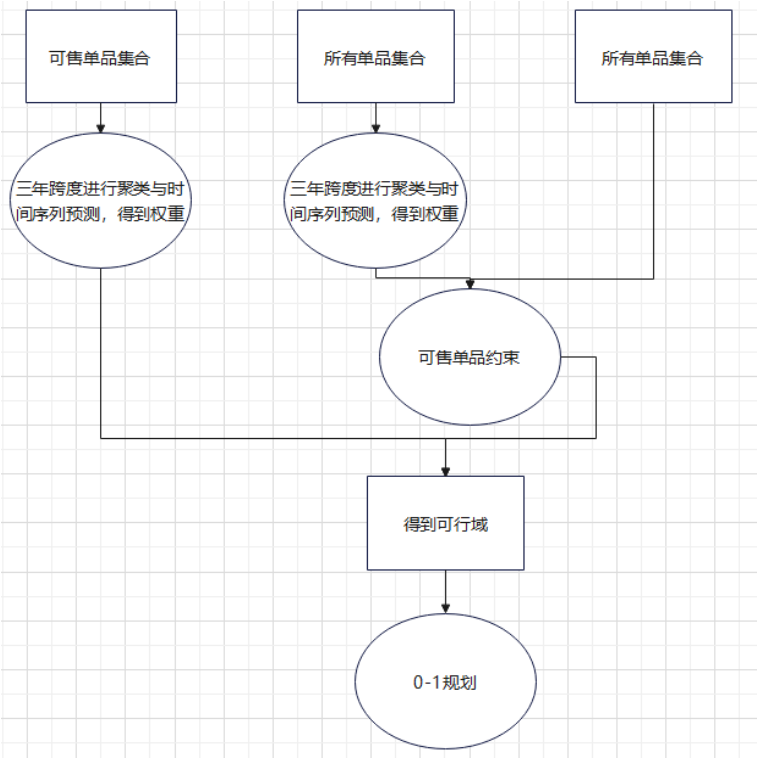

图 10: 问题三思维导图

问题三相较于问题二，将研究对象细化为 6 月 24-30 日的可售单品。为得到可行域，即 0-1 规划的研究对象，有三个侧重不同的思路。

1. 定义 “有供应集” $\tilde{\Omega}$，其元素全为 (i,j) 形式的指标。由于对象数量大，采取**聚类法与时间序列预测法**为各可售单品赋权值 $gn_{ij}\ \ (i,j)\in\tilde{\Omega}$，并按照权值从大到小进行排序。取前 E_i 件单品参与后续操作。这 E_i 件单品满足

$$\frac{E_i}{J_i}\leq\lambda\quad(0<\lambda<1) \tag{14}$$

其中，λ 是人为规定的常数，表示选取的高权重单品占所有单品的比例。将前 E_i 件单品的指标写入全指标组集合得到可行域 $\Omega_0^{(1)}$。

2. 直接在全体单品范围内进行**聚类法与时间序列预测法**操作，同样可以得到 $\Omega_0^{(2)}$。再结合约束有供应集 $\tilde{\Omega}$, 得到可行域 $\Omega_0^{(2)}\bigcap\tilde{\Omega}$。

3. 跳过赋权，对所有可售单品的数据进行处理，可行域即 $\tilde{\Omega}$。

找到可行域后将可行域统一记为 Ω_0。

为产生权重，首先为各单品销量找到 K 个聚类中心，定义 g_{i_s,j_s} 为第 S 个类的中心 $(S=1,2,\ldots,K)$。该月单品累计销量与该单品所属 S 类中心的累计销量作比，得到比例系数 $\gamma_{i,j}$

$$\gamma_{i,j} = \frac{\sum_t gsw_{ij}^{(t)}}{\sum_t gsw_{i_s,j_s}^{(t)}} \quad (i,j) \in \Omega_s \tag{15}$$

若预测出中心的权值 $gn_{i_s,j_s}^{(PD_{23_74}^{(7)})}$ 后，自然该类其余

$$gn_{i,j}^{(PD_{23_74}^{(7)})} = \gamma_{i,j} \times gn_{i_s,j_s}^{(PD_{23_74}^{(7)})} \tag{16}$$

于是利用时间序列预测各中心的 $gn_{i_s,j_s}^{(PD_{23_74}^{(7)})}$。通过 QTS 时间序列预测，以过去三年各单品的销量数据为基础，以周为单位预测得到各中心的权值。

x_{ij} 作为判断变量表示该单品是否选择进货。接下来进行 0-1 规划。以获得最大利润为目标函数：

$$\max Pro = \sum_{(i,j)\in\tilde{\Omega}} \hat{n}_{ij} \times [(1-r_{ij}) \times p_{ij} + r_{ij} \times (p_{ij} \times q_{ij})] \times x_{ij} - \sum_{(i,j)\in\tilde{\Omega}} \hat{b}_{ij} \times x_{ij} \times m_{ij} \tag{17}$$

而据题意转化的约束条件如下：

$$\begin{cases} M = \sum_i \sum_j x_{ij} \in [27, 33] \\ \sum_j x_{ij} \geq 1 \quad (1 = 1, 2, \ldots, 6) \\ m_{ij} \geq 2.5 \qquad (i,j) \in \{(i,j) \mid x_{ij} = 1\} \\ x_{ij} = 0 \ or \ 1 \quad (i,j) \in \Omega_0 \end{cases} \tag{18}$$

则构建的 0-1 规划数学模型如下：

$\max \quad Pro = \sum_{(i,j)\in\tilde{\Omega}} \hat{n}_{ij} \times [(1-r_{ij}) \times p_{ij} + r_{ij} \times (p_{ij} \times q_{ij})] \times x_{ij} - \sum_{(i,j)\in\tilde{\Omega}} \hat{b}_{ij} \times x_{ij} \times m_{ij}$

$$s.t. \begin{cases} M = \sum_i \sum_j x_{ij} \in [27, 33] \\ \sum_j x_{ij} \geq 1 \quad (1 = 1, 2, \ldots, 6) \\ m_{ij} \geq 2.5 \qquad (i,j) \in \{(i,j) \mid x_{ij} = 1\} \\ x_{ij} = 0 \ or \ 1 \quad (i,j) \in \Omega_0 \end{cases} \tag{19}$$

利用 Lingo 对该 0-1 目标规划模型进行求解，得到 7 月 1 日的最佳单品补货量和最佳定价策略如下：

5.4 问题四

5.4.1 供给侧

供应商信息：了解不同供应商的交货可靠性、价格和质量可以帮助商超选择最合适的供应商，并确保稳定供应。在某单品供应不稳定时，利用问题一的聚类结果与关系计数矩阵，可以找到与该单品互补性较弱、替代性较强的单品进行进货，以填补供货缺口，满足市场需求。

5.4.2 需求侧

反馈和投诉：消费者反馈和投诉数据可以帮助商超改进蔬菜品质和服务，提高消费者满意度。蔬菜类超商面临的问题之一就是进货种类与进货量，而消费者反馈和投诉数据可以灵敏地反映市场需求，对于超商进货决策乃至定价决策具有重要指导意义。

6 模型检验

针对问题二 MPL 模型进行检验。随机抽取一种品类的蔬菜，对该回归模型进行 F 检验，即检验全部解释变量对被解释变量的共同影响是否显著。给出食用菌品类销售总量与成本加成定价关系的检验结果。

参数	参数估计值	参数置信区间
β_1	-676.553	[-970.829, -382.277]
β_2	0.675	[-0.123, 1.472]
β_3	-11.949	[-14.568, -9.330]
R^2=0.9400, F=58.636, p<0.0001		

表 4: MLP 模型的计算结果

结果分析：表 4 显示 $R^2 = 0.9400$，即因变量 (销售量) 的 94%。可由检验模型确定,F 值远远超过 F 检验的临界值, p 值远小于 α, 因而模型从整体来看是可用的。

7 模型评价

7.1 模型优点

1. 针对问题一：关系计数矩阵的建立可以在样本较少时，直观展现不同蔬菜间的相互关系，并且利用判断矩阵检验 “替代与互补” 的自洽性；K-Means 聚类算法可以高效地将多个样本收敛至极少的样本，方便进行数据处理与归纳，同时展现同簇蔬菜与不同簇蔬菜间存在的相互关系。

2. 针对问题二：对于大量数据的压缩可以大大提高算法的效率，同时满足题目的要求。时间序列能对数据进行科学预测，为非线性规划模型的建立与优化提供基础。

3. 针对问题三：三种寻找可行域的思路侧重点不同：赋权难度大的思想得到的权值更加精确，科学性更强，但赋权所需时间更长；赋权难度小的思路权值科学性稍弱，但能够缩短赋权所需时间；不进行赋权的思路直接对各单品进行分析，数据可靠性最大，但求解 0-1 规划的难度相当大。可以根据需求灵活选用不同思路，构建 0-1 规划模型进行求解。同时赋权的过程综合运用聚类法与时间序列预测法，科学性较高。

7.2 模型缺点

1. 针对问题一：为提高效率，对时间的划分尺度较大，精度不高；且聚类结果并非全局最优而是局部最优的情况是难免的。

2. 针对问题二：得到各蔬菜品类“周等效”、“日等效”值时采用较便捷的方法直接取均值，科学性有待考究，可以以科学的方式进行赋权。

3. 针对问题三：0-1 规划时间复杂度高，运算量较大，对精确结论的得出是一个考验。同时，对于题目所提及的“尽量满足市场对各品类蔬菜商品需求的前提”的要求，并没有很好地体现。

8 模型改进

针对模型三的缺点分析，提出对原本规划模型的改进。由于时间复杂度高，运算量较大，可以采用蒙特卡洛算法进行随机试验，快速得到近似的结果。而对于题目所提及的“尽量满足市场对各品类蔬菜商品需求的前提”的要求，采用目标规划模型进行改良。

引入 $d_i^{\pm}$ 作为第 i 个数据量关于理想值的正/负偏差变量，$(0 \leq i \leq 6)$。目标函数为：

$$\min \quad \omega_1 d_0^- + \omega_2 \sum_{i=1}^{6} d_i^- \tag{20}$$

设定理想点 $Pro^{(0)}$, 即存在约束条件：

$$Pro^{(0)} = Pro - d_0^+ + d_0^- \tag{21}$$

由原本 0-1 规划模型的约束条件，引入正/负偏差变量的概念后，得到完整的目标规划模型：

$$\min \quad \omega_1 d_0^- + \omega_2 \textstyle\sum_{i=1}^{6} d_i^-$$

$$s.t. \begin{cases} Pro^{(0)} = Pro - d_0^+ + d_0^- \\ Pro = \sum_{(i,j)\in\tilde{\Omega}} \hat{n}_{ij} \times [(1 - r_{ij}) \times p_{ij} + r_{ij} \times (p_{ij} \times q_{ij})] \times x_{ij} - \sum_{(i,j)\in\tilde{\Omega}} \hat{b}_{ij} \times x_{ij} \times m_{ij} \\ M = \sum_i \sum_j x_{ij} \in [27, 33] \\ \sum_j x_{ij} - d_i^+ + d_i^- = 0 \\ \sum_j x_{ij} \geq 1 \quad (1 = 1, 2, \ldots, 6) \\ x_{ij} = 0 \ or \ 1 \qquad (i, j) \in \tilde{\Omega} \\ d_0^{\pm}, d_1^{\pm}, \ldots, d_6^{\pm} \geq 0 \end{cases} \tag{22}$$

9 模型推广与应用

1. 数据驱动的决策：该模型可以基于历史销售数据、季节性变化、供应链情况等因素进行分析和预测，帮助制定更精确的定价和补货策略。这样可以降低过多的库存或库存不足，提高库存的周转率。

2. 实时调整：该模型可以随着市场情况的变化而调整定价和补货策略，使决策更加科学。这对于应对季节性需求波动或突发事件（如供应链中断）非常有帮助。

3. 提高竞争力：通过更准确的定价策略，企业可以在市场上提供更有竞争力的价格，吸引更多的客户，并提高市场份额。

4. 降低风险：该模型可以帮助识别潜在的风险和机会，预测需求不足或过剩的情况，从而降低了因库存问题而导致的损失。

5. 数据分析和洞察：建立数学模型后，可以通过分析模型的输出结果来获得有关市场趋势、客户偏好和产品表现的更多洞察，从而指导更深入的战略决策。

总之，该数学模型可以帮助大部分易腐烂和有限保质期的食品和非食品产品商品供应商或零售商更有效地管理库存、定价和补货，从而提高效益、降低成本、提高竞争力，并更好地满足客户需求。这对于现代商业环境中的生鲜类商品供应链管理至关重要。

10 参考文献

1. 司守奎、孙兆亮. 数学建模算法与应用，第 2 版. 国防工业出版社，2015.

2. Paul A. Samuelson.Economics. McGraw-Hill,1948.Page769.

3. 张良均、谭立云、刘名军、江建明.Python 数据分析与挖掘实践，第 2 版. 机械工业出版社，2019.

11 附录

MPL 回归

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from scipy import stats
import matplotlib.pyplot as plt
import pandas as pd


def del_bad_points(X,y):
    # 计算Z-Score
    z_scores = stats.zscore(X)

    # 设置阈值
    threshold = 3
```

```
    # 检测离群点
    outliers = np.where(np.abs(z_scores) > threshold)[0]

    # 删除离群点
    filtered_X = np.delete(X, outliers, axis=0)
    filtered_Y = np.delete(y, outliers)
    return filtered_X, filtered_Y


def eval(coef, intercept, y_pred):
    writer = pd.ExcelWriter('MLP_Verify.xlsx', engine='openpyxl')
    # 计算置信区间
    alpha = 0.05  # 置信水平
    n = X.shape[0]  # 样本数量
    p = X.shape[1]  # 自变量数量
    dof = n - p - 1  # 自由度
    t_value = stats.t.ppf(1 - alpha / 2, dof)  # t分布临界值

    # 计算参数的标准误差
    X_mean = np.mean(X, axis=0)
    X_centered = X - X_mean
    MSE = np.sum((y - y_pred) ** 2) / dof  # 均方误差
    X_cov = np.dot(X_centered.T, X_centered)
    X_cov_inv = np.linalg.inv(X_cov)
    se = np.sqrt(np.diagonal(MSE * X_cov_inv))

    # 计算置信区间
    lower_bound = coef - t_value * se
    upper_bound = coef + t_value * se

    # 计算决定系数
    r_squared = r2_score(y, y_pred)

    # 计算F统计量和p值
    SSR = np.sum((y_pred - np.mean(y)) ** 2)  # 回归平方和
    SSE = np.sum((y - y_pred) ** 2)  # 残差平方和
    MSS = SSR / p  # 回归均方和
    MSE = SSE / dof  # 残差均方和
```

```
    F_value = MSS / MSE
    p_value = 1 - stats.f.cdf(F_value, p, dof)

    # 计算剩余方差
    residual_variance = SSE / dof

    # 创建汇总表
    summary_table = pd.DataFrame({
        'Variable': ['Intercept'] + [f'X{i + 1}' for i in range(p)],
        'Coefficient': np.insert(coef, 0, intercept),
        'Lower Bound': np.insert(lower_bound, 0, intercept),
        'Upper Bound': np.insert(upper_bound, 0, intercept),
        'R-squared': np.repeat(np.array(e), len(coef) + 1)
    })
    summary_table['R-squared'] += r_squared
    summary_table['F-value'] = F_value
    summary_table['p-value'] = p_value
    summary_table['Residual Variance'] = residual_variance

    # 打印汇总表
    # print(summary_table['R-squared'])
    summary_table.to_excel(writer, sheet_name=f'{k}')
    writer.close()


def residual_cal(y_pred):
    # 计算残差
    residuals = y - y_pred

    # 绘制残差图
    plt.scatter(y_pred, residuals)
    plt.axhline(y=0, color='red', linestyle='--')
    plt.xlabel('Predicted Values')
    plt.ylabel('Residuals')
    plt.title('Residual Plot')
    plt.show()


def fix_time(X, y):
```

```
    fixed_t = 4
    # 创建多元线性回归模型
    model = LinearRegression()
    # 拟合模型
    model.fit(X, y)
    # 获取回归系数和截距
    coefficients = model.coef_
    intercept = model.intercept_
    # 输出回归系数和截距
    print("回归系数:", coefficients)
    print("截距:", intercept)

    # 预测值
    y_pred = model.predict(X)
    # 在固定x1的情况下, 生成预测值
    x1_range = np.linspace(X.T[0].min(), X.T[0].max(), 100)  # x1的取值范围
    x2_range = np.linspace(X.T[2].min(), X.T[2].max(), 100)
    x3_range = np.linspace(X.T[3].min(), X.T[3].max(), 100)
    X_fixed_x1 = np.column_stack((x1_range, np.full_like(x1_range, fixed_t), x2_ran
# 固定x1的自变量矩阵
    y_pred_fixed_x1 = model.predict(X_fixed_x1)  # 预测值

    # 在固定x1的情况下绘制图像
    plt.rc('font', family='SimHei')
    plt.plot(x1_range, y_pred_fixed_x1, color='red')
    index = [X[:, 1] == fixed_t]
    plt.scatter(X[index][:, 0], y[index], color='blue', alpha=0.9, s=25)
    plt.xlabel('销售单价', fontsize=20)
    plt.ylabel('销量', fontsize=20)
    plt.title(f'{k}{fixed_t}月份销量-定价图', fontsize=25)
    plt.show()
    plt.savefig(rf"F:\2023数模\sale-price-graph\{k}{fixed_t}月份销量-定价图.png", d


if __name__ == '__main__':
    e = 0.8
    NUM = 4
    count = 0
    price = pd.read_excel('kindDivided.xlsx')
```

```
price['销售日期'] = pd.to_datetime(price['销售日期'])
price['month'] = price['销售日期'].dt.month
x = price[['销售单价(元/千克)', 'month', '销量(千克)', '种类']]
kinds = price['种类'].unique()
coef = []
for k in kinds:
    X = x[x['种类'] == k][['销售单价(元/千克)', 'month']].to_numpy()
    y = x[x['种类'] == k]['销量(千克)'].to_numpy()
    # cross_terms = np.multiply(X[:, 0], X[:, 1])
    # X = np.column_stack((X, cross_terms))
    X = np.column_stack((X, np.reciprocal(X[:, 0])))

    # 去除离群点
    X_filted,y_filted = del_bad_points(X, y)

    # 建立模型
    model = LinearRegression()

    model.fit(X, y)

    # 获取回归系数和截距
    coefficients = model.coef_
    intercept = model.intercept_
    coef.append(np.concatenate((coefficients, [intercept])))

    # 输出回归系数和截距
    print(f'{k}:')
    print("回归系数:", coefficients)
    print("截距:", intercept)
    print('KSW=' + str(coefficients[2]) + ' 1/x ' + str(coefficients[0]) + ' x
```

```
        plt.legend()
        plt.show()

    np.savetxt('coefs.txt', np.array(coef), fmt='%.8f')
```

聚类

```
import numpy as np
import pandas as pd
from collections import Counter
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# 准备种类信息
name_path = '附件1.xlsx'
name_df = pd.read_excel(name_path)
name_dict = {}
for index, row in name_df.iterrows():
    name_dict[row[1]] = row[3]
print(name_dict)

data_path = 'customerBased_3s.xlsx'
df1 = pd.read_excel(data_path)
df1.columns = ['销售日期', '组别']
goods = df1['组别'].apply(lambda x: x.split('␣')[::2])
goods_sum = np.concatenate(goods.values)
customer_num = Counter(goods_sum)
print(customer_num)

sales_path = 'monthlyPlus.xlsx'
df2 = pd.read_excel(sales_path)
grouped = df2.groupby('单品名称')\
    .agg({'销量(千克)': 'sum', '销售单价(元/千克)': 'mean'}).reset_index()

kinds = list(set(name_dict.values()))
info_all_kind = [[] for _ in range(6)]
for index, row in grouped.iterrows():
    for k in kinds:
        if name_dict[row[0]] == k:
            info_all_kind[kinds.index(k)].append([row[1], row[2], customer_num[row[
```

```
print(info_all_kind)

# 开始聚类
print('begin:')
kmeans = KMeans(n_clusters=3)

for i in range(len(kinds)):
    kmeans.fit(info_all_kind[i])
    labels = kmeans.labels_
    cluster_centers = kmeans.cluster_centers_

    x = [x[0] for x in info_all_kind[i]]
    y = [y[1] for y in info_all_kind[i]]
    z = [z[2] for z in info_all_kind[i]]

    plt.rc('font', family='SimHei')

    ax = plt.axes(projection='3d')
    fig = ax.scatter3D(x, y, z, c=labels, cmap='viridis')
    ax.set_xlabel('销量')
    ax.set_ylabel('定价')
    ax.set_zlabel('顾客购买量')
    ax.set_title(f'{kinds[i]}聚类分析图')
    plt.colorbar(fig)

    plt.savefig(rf'F:\2023数模\图\{kinds[i]}聚类分析图.png', dpi=500)
    plt.show()
```

打折对比

```
import pandas as pd

# 读取数据
data_path = 'withName.xlsx'
df = pd.read_excel(data_path)

# grouped = df1.groupby([df1['销售日期'].dt.to_period('M'), '种类'])['销量(千克)'].
# print(grouped)

g = df.groupby(['种类', '是否打折销售'])['销售单价(元/千克)'].mean()
```

```
g.to_excel('打折价格对比.xlsx', index=True)
```

折线图绘制

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_excel('monthlyFull.xlsx')
discount = pd.read_excel('打折价格对比.xlsx')

discount_dict = {}
for index, row in discount.iterrows():
    if not pd.isna(row[0]):
        kind = row[0]
        salePrice = row[2]
    else:
        discount_dict[kind] = row[2] / salePrice

data['折损利润率'] = data['利润率'] * (1 - data['损耗率']) \
                    + (data['销售单价(元/千克)'] * data['种类'].replace(discount_d
                    / data['批发价格(元/千克)'] * data['损耗率']
grouped = data.groupby(['销售日期', '种类']).agg({'销量(千克)': 'sum', '折损利润率'

kinds = discount_dict.keys()

plt.rc('font', family='SimHei')

for k in kinds:
    data_k = grouped[grouped['种类'] == k]
    print(data_k)
    fig, ax1 = plt.subplots()

    ax1.plot(data_k['销售日期'], data_k['销量(千克)'], 'b-', label='销量(千克)')
    ax1.set_xlabel('时间')
    ax1.set_ylabel('销量(千克)', color='b')
    ax1.tick_params('y', colors='b')

    ax2 = ax1.twinx()

    # 绘制第二组数据（右坐标轴）
    ax2.plot(data_k['销售日期'], data_k['折损利润率'], 'r-', label='折损利润率')
```

```
    ax2.set_ylabel('折损利润率', color='r')
    ax2.tick_params('y', colors='r')

    lines, labels = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax1.set_xticklabels(data_k['销售日期'], rotation=45, ha='right', fontsize=6)
    ax1.legend(lines + lines2, labels + labels2, loc='upper right')
    ax1.set_title(f'{k}销量与成本加成分析图')

    # plt.show()
    plt.savefig(rf'F:\2023数模\折线图\{k}销量与成本加成分析图.png', dpi=500)
```

时间序列分析

```
from itertools import product
import numpy as np
import pandas as pd
import statsmodels.api as sm
import warnings
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.stattools import adfuller

# 检验时间序列平稳性
def verify(ts, kind):
    # ADF检验
    adf_pvalue = adfuller(ts)[1]
    diff_count = 0

    # 进行差分直到p值符合要求
    while adf_pvalue >= 0.01:
        ts = np.diff(ts)
        adf_pvalue = adfuller(ts)[1]
        diff_count += 1

    # Ljung-Box检验
    ljung_box_pvalue = acorr_ljungbox(ts)

    # 输出差分次数和Ljung-Box检验结果
    print(kind, ":")
    print("差分次数:", diff_count)
    print("Ljung-Box检验p值:", ljung_box_pvalue['lb_pvalue'].max())
```

```
    return diff_count

# 网格搜索寻找最优参数
def para_search(p, d, q):
    # 初始化最小BIC值
    best_bic = np.inf

    # 网格搜索
    for param in product(p, d, q):
        try:
            model = sm.tsa.ARIMA(ts, order=param)
            model_fit = model.fit()
            bic = model_fit.bic
            if bic < best_bic:
                best_bic = bic
                best_order = param
        except:
            continue

    return (best_order, best_bic)


if __name__ == '__main__':
    warnings.filterwarnings("ignore")
    df_merge = pd.read_excel('kindDividedWeekly.xlsx')
    kinds = df_merge['种类'].unique()
    # 定义参数范围
    p = range(1, 4)  # 自回归阶数
    q = range(1, 4)  # 移动平均阶数

    for k in kinds:
        # 创建时间序列
        ts = df_merge.set_index('销售日期')
        # 对进价分析
        # ts = ts[ts['种类'] == k]['批发价格(元/千克)'].tolist()
        # 对销量分析
        ts = ts[ts['种类'] == k]['销量(千克)'].tolist()
```

```
        diff_count = verify(ts, k)
        d = [diff_count]
        param = para_search(p, d, q)
        print(param)

        model = sm.tsa.ARIMA(ts, order=param[0])
        model_fit = model.fit()

        # 获取预测结果的置信区间 (置信水平为95%)
        forecast = model_fit.get_forecast(steps=1, alpha=0.2)
        confidence_interval = forecast.conf_int()

        # 输出预测结果的置信区间
        print(f"{k}的预测值和置信区间:")
        print(forecast.predicted_mean, confidence_interval)
```

进价利润数据整合

```
import pandas as pd

def PriceIn():
    df = pd.read_excel('附件3.xlsx')

    # 将销售日期列转换为日期类型
    df['销售日期'] = pd.to_datetime(df['日期'])

    # 按照一个月为周期来合并销售日期，并按单品编码列的不同来合并所有的行，并将销量(
    df_merged = df.groupby([df['销售日期'].dt.to_period('M'), '单品编码'])['批发价格
    df_merged.to_excel('PriceInmonthly.xlsx', index=True)

def profit_and_waste_kind():
    priceIn = pd.read_excel('PriceInWithName.xlsx')
    priceSale = pd.read_excel('kindDivided.xlsx')
    waste = pd.read_excel("附件4-复制.xlsx")

    priceIn = priceIn.groupby([[priceIn['销售日期'].dt.to_period('D'), '种类']])['批
    priceIn['new_index'] = priceIn['种类'] + priceIn['销售日期'].astype(str)
    priceSale['new_index'] = priceSale['种类'] + priceSale['销售日期'].astype(str)
    priceIn.set_index('new_index', inplace=True)
    priceSale.set_index('new_index', inplace=True)
```

```
    merged_df = pd.concat([priceIn, priceSale], axis=1, join='inner')
    merged_df = merged_df.T.drop_duplicates().T
    waste_dict = {}
    for index, row in waste.iterrows():
        waste_dict[row[1]] = row[2] / 100

    merged_df['损耗率'] = merged_df['单品名称'].replace(waste_dict)
    merged_df['利润'] = merged_df['销售单价(元/千克)'] - merged_df['批发价格(元/千克
    merged_df['利润率'] = (merged_df['销售单价(元/千克)'] - merged_df['批发价格(元/
    print(merged_df)

    merged_df = merged_df[['单品名称', '种类', '销售日期', '销售单价(元/千克)', '批
    # merged_df.to_excel('monthlyUltra.xlsx', 'w', index=False)


def PriceIn_kindDiv():
    df = pd.read_excel('kindDivided.xlsx')
    priceIn = pd.read_excel('PriceInWithName.xlsx')
    priceIn['销售日期'] = priceIn['销售日期'].dt.to_period('D').dt.to_timestamp()
    df['销售日期'] = pd.to_datetime(df['销售日期'])

    priceIn_grouped = priceIn.groupby(['种类', pd.Grouper(key='销售日期', freq='W')
    df_grouped = df.groupby(['种类', pd.Grouper(key='销售日期', freq='W')]).agg(
        {'销量(千克)': 'sum', '销售单价(元/千克)': 'mean'}).reset_index()
    print(priceIn_grouped)
    print(df_grouped)

    priceIn_grouped['new_index'] = priceIn_grouped['种类'] + priceIn_grouped['销售日
    df_grouped['new_index'] = df_grouped['种类'] + df_grouped['销售日期'].astype(st

    priceIn_grouped.set_index('new_index', inplace=True)
    df_grouped.set_index('new_index', inplace=True)
    print(priceIn_grouped)
    print(df_grouped)

    merged_df = pd.concat([priceIn_grouped, df_grouped], axis=1, join='inner')
    merged_df = merged_df.T.drop_duplicates().T
    merged_df.to_excel('kindDividedInfo.xlsx', index=False)
    # df['销售日期'] = pd.to_datetime(df['销售日期'])
```

```
    # res = merged_df.groupby(['种类', pd.Grouper(freq='W')]).agg(
    #     {'销量(千克)': 'sum', '批发价格(元/千克)': 'mean'}).reset_index()


def profit_and_waste_monthly():
    priceIn = pd.read_excel('PriceInmonthly.xlsx')
    priceSale = pd.read_excel('monthlyPlus.xlsx')
    waste = pd.read_excel("附件4-复制.xlsx")
    print(waste)

    priceIn['new_index'] = priceIn['单品名称'] + priceIn['销售日期'].astype(str)
    priceSale['new_index'] = priceSale['单品名称'] + priceSale['销售日期'].astype(s
    priceIn.set_index('new_index', inplace=True)
    priceSale.set_index('new_index', inplace=True)

    merged_df = pd.concat([priceIn, priceSale], axis=1, join='inner')
    merged_df = merged_df.T.drop_duplicates().T
    waste_dict = {}
    for index, row in waste.iterrows():
        waste_dict[row[1]] = row[2] / 100

    merged_df['损耗率'] = merged_df['单品名称'].replace(waste_dict)
    merged_df['利润'] = merged_df['销售单价(元/千克)'] - merged_df['批发价格(元/千克
    merged_df['利润率'] = (merged_df['销售单价(元/千克)'] - merged_df['批发价格(元/

    merged_df = merged_df[['单品名称', '种类', '销售日期', '销售单价(元/千克)', '批
    # merged_df.to_excel('monthlyUltra.xlsx', 'w', index=False)


if __name__ == '__main__':
    # profit_and_waste_monthly()
    # profit_and_waste_kind()
    PriceIn_kindDiv()
```

lstsetlanguage=C

```
    import random
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
```

```
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import curve_fit


def cre_kindDiv_file():
    df = pd.read_excel('withName.xlsx')

    df['销售日期'] = pd.to_datetime(df['销售日期'])
    df_merged = df.groupby([df['销售日期'].dt.to_period('D'), '种类']).agg(
        {'销量(千克)': 'sum', '销售单价(元/千克)': 'mean'}).reset_index()
    print(df_merged)
    df_merged.to_excel('kindDivided.xlsx', index=True)


def func(x, a, b):
    return a * np.log(b * x)


def regression(df, kind):
    df_kind = df[['销量(千克)', '销售单价(元/千克)']]

    x = df_kind['销量(千克)'].to_numpy()
    y = df_kind['销售单价(元/千克)'].to_numpy()
    # 转换特征为多项式特征
    # poly_features = PolynomialFeatures(degree=3)
    # X_poly = poly_features.fit_transform(x.reshape(-1, 1))
    #
    # # 创建多项式回归模型
    # model = LinearRegression()
    # model.fit(X_poly, y)
    #
    # # 预测结果
    # y_pred = model.predict(X_poly)
    # params, _ = curve_fit(func, x, y)
    # y_pred = func(x, *params)
    # 绘制散点和拟合曲线
    plt.rc('font', family='SimHei')
    plt.scatter(x, y, color='r', alpha=0.5, s=15)
    # plt.plot(x, y_pred, color='red', label='regress')
```

```
    plt.xlabel('销量(千克)')
    plt.ylabel('销售单价(元/千克)')
    plt.title(f"{kind}回归分析图")
    plt.show()
    plt.savefig(rf"F:\2023数模\sale-price-graph\{kind}回归分析图.png", dpi=600)


def rand_points(df):
    points = []
    while len(points) < 1:
        points.clear()
        randoms = np.random.randint(0, df.shape[0], 1000)
        for rand in randoms:
            if rand >= 7 and rand <= df.shape[0] - 7:
                tmp = df.iloc[rand - 7:rand + 7, :]
                sub1 = tmp['销量(千克)']
                sub2 = tmp['销售单价(元/千克)']
                rate1 = sub1.pct_change().dropna()
                rate2 = sub2.pct_change().dropna()
                if (rate1 < 0.3).all().all() and (rate2 < 0.2).all().all():
                    points.append(rand)
    res = pd.DataFrame()
    # points = random.sample(points, k=1)
    for i in points:
        tmp = df.iloc[i - 3:i + 4, :]
        res = pd.concat([res, tmp], axis=0)
    return res


if __name__ == '__main__':
    data_df = pd.read_excel('kindDivided.xlsx')
    kinds = data_df['种类'].unique()
    for k in kinds:
        df_kind = data_df[data_df['种类'] == k]
        # sub_df = rand_points(df_kind)
        # print(sub_df)
        regression(df_kind, k)
```