基于时间序列分析与贪心算法的定价与补货策略问题研究

摘要

随着大数据科学的发展，生鲜商超蔬菜类商品的定价策略和补货策略也能够通过大数据和机器学习的方式进行更加合理的规划。本文针对蔬菜类商品的自动定价与补货决策问题，通过斯皮尔曼相关分析与 FP-growth 模型，进行对各品类、各单品的时间分布和订单分布的分析，找寻之间的相互关系，通过对各品类销售总量与成本加成定价关系的拟合关系以及利用时间序列分析模型预测的未来一周各品类需求量和成本价，给出未来一周的补货总量和定价策略，并通过贪心算法给出数据集限定后的 7 月 1 日的单品补货量和定价策略，最后给出了商超还需要采集的相关数据。

首先，对附件 1-4 的数据进行预处理。观察发现附件 2 中明细存在“退货”记录，我们将**“退货”数据及其所对应的“销售”数据进行匹配后删除**，并按照 “单品编码”与附件 1 中的“分类名称”和“单品名称”进行归类，联合附件 3 中“批发价格(元/千克)”对**利润、成本价格、销售量等相关数据进行求解**。

针对问题一，首先利用总销售量饼状图，得到总销量的排序规律，再利用各品类、部分单品销售量的时间分布曲线，得出**销售量呈现出明显的季节周期分布规律等重要结论**。接着利用 120s 的时间间隔将出售的单个订单进行分割组合，建立 FP-growth 模型，得到**各品类、各单品在订单中的分布规律**，并得到**各品类、各单品的支持度与之间的关联规则置信度**，同时通过斯皮尔曼相关性分析，发现**花叶类与花菜类、花叶类与食用菌、食用菌与水生根茎类为强相关关系，茄类与水生根茎类、辣椒类、食用菌不相关**。

针对问题二，通过斯皮尔曼相关性分析，发现蔬菜各品类销售量与成本加成率相关性较低，但与定价为中负相关关系，故使用蔬菜各品类销售量与定价进行拟合。使用剔除异常数据点的数据，建立时间序列分析模型，分别得到各品类未来七天需求量和成本价的预测值。通过**计算函数最大值的方式得到最佳定价策略与补货策略**，获得七天总利润为 22667.5 元，并利用敏感性分析发现“花叶类”与“食用菌”数据较为敏感，其他均具有很强的稳定性。

针对问题三，首先找出可供销售的单品，建立起保质期与损耗率的关系，利用附件 4 的损耗率计算出合适当日销售的单品，考虑商超的多种约束情况，使用背包问题贪心算法，得到**利润最大的单品定价策略与补货策略**，单日总利润为 971.1 元。

针对问题四，我们总结了六个商超补货和定价需考虑的影响因素，分别**为客流量数据、天气数据、库存状况、竞争对手的定价、供应商的配送效率、时间和可靠性与消费者反馈意见和数据**，并给出了相应因素对于商超蔬菜商品补货的定价的建议及理由。

本文依据提供的商超数据与实际情况，利用多种分析方法与数学模型，分析了各品类、各单品之间的分布规律与相互关系，**从不同方面和约束下提供了利润最大的单品定价策略与补货策略**。我们是模型具有实用性、精准性与稳定性，切实解决了困扰商超管理者的问题。

关键词：商超定价，补货策略，FP-growth 模型，ARIMA 模型，成本加成定价法，斯皮尔曼相关性分析，贪心算法

一、 问题重述

1.1 问题背景

生鲜超市中蔬菜类商品的保鲜期都比较短，且长时间没卖出去的蔬菜品相会变差，大部分品种必须在当天全部售出。因此，各商品的历史销售和需求情况是商超每天进行补货时依据的重要因素。由于商超销售的蔬菜品种众多、产地不尽相同，而蔬菜的进货交易时间通常是凌晨 3:00-4:00，商超补货时无法知道具体单品和进货价格的情况。蔬菜的定价通常采用“成本加成定价”的方法，即按产品单位成本加上一定比例的利润来制定产品价格，大多数企业是按成本利润率来确定所加利润的大小。而对于运损和品相变差的商品，商超通常会进行打折销售。可靠的市场需求分析，对于商超制定合理的补货决策和定价决策从而实现收益最大化十分重要。在不同的季节，顾客对于蔬菜类商品的需求有所变化，而蔬菜供应品种供应较为丰富的时间集中在 4 月至 10 月，商超销售空间的限制使得合理的销售组合变得极为重要。

1.2 问题重述

现有一商超的从 2020 年 7 月 1 日至 2023 年 6 月 30 日各商品的信息、销售流水明细批发价格与近期的损耗率的相关数据，其中该商超的商品均为蔬菜类商品，我们团队希望依据上述数据，通过更进一步的分析，分析以下问题：

问题一：在数据中，我们可以看出该商超的蔬菜类商品不同品类或不同单品之间，在销售时间等维度上可能存在一定的关联关系。我们需要分析蔬菜各品类及单品销售量的在不同日期、不同季节、不同时段的分布规律及相互关系。

问题二：我们需要分析各蔬菜品类的销售总量与成本加成定价的关系，并通过模型为商超各蔬菜品类未来一周（2023 年 7 月 1 日-7 日）的日补货总量和定价策略，以使商超收益最大化。

问题三：由于销售空间有限，商超计划优化单品补货策略，以确保可售单品总数在 27-33 之间，并满足每个单品的最低陈列要求为 2.5 千克。我们需要根据 2023 年 6 月 24 日至 30 日的销售情况，提供 7 月 1 日的单品补货数量和定价策略，旨在在满足市场对各类蔬菜商品需求的基础上，最大限度地增加商超的收益。

问题四：为了更好地制定蔬菜商品的补货和定价决策，我们需要给出商超还需要采集的相关数据类型，以及这些数据对解决上述问题的帮助。

二、 问题分析

2.1 问题一的分析

根据日常生活经验可以发现，蔬菜类商品不同品类或不同单品的销量有一定的分布规律，受季节性影响，一些蔬菜销量分布有明显的季节周期波动性，比如西瓜、南瓜、黄瓜等在夏季更受人们的青睐，而冬季则有更多的白菜、萝卜、芥菜等。除了季节性因素，地域对于蔬菜的销量分布也有很大的影响，在不同的地区，气候、饮食习惯、文化等会影响蔬菜的种植和消费。例如，南方热带地区更倾向于种热带水果和蔬菜，而寒冷地区更倾向于种植根菜类，这种地域差异就造成了各种蔬菜品类销量地域分布上的差别。各种蔬菜品类和单品之间也存在着一定的相关关系，人们往往习惯于同时购买多种蔬菜，某种蔬菜的销量增加可能带动另一种蔬菜销量的增加，此外，一种蔬菜价格上涨，也会导致消费者转向价格更低的或更易获得的替代品，从而影响其

他品类蔬菜的销量。

问题 1 首先需要我们分析某一家商超的蔬菜类商品不同品类及单品销量的分布规律，故我们不需要考虑地域因素，而是重点需要分析时间季节因素对于分布规律的影响，我们首先可以探究蔬菜各品类及单品销售量时间分布的规律，接着根据数据集的分布规律寻找合适的相关性分析模型，探究品类两两之间的相关关系。此外，我们还可以进一步使用 FP-growth 算法探究品类与品类之间是如何相互影响，如何关联在一起的，最后用可视化的方式呈现出结果。单品的处理思路与品类相同，但其个数远多于品类，给结果的可视化呈现带来了较大的困难，针对于此，我们可以先找出关联度较大的几组单品，再去呈现可视化结就更为清晰可观。

2.2 问题二的分析

问题二中只需要针对品类进行分析。我们需要根据已有的数据集给出 6 个蔬菜品类的销售总量与成本加成定价的关系，其中成本加成定价可以由利润率等定量给出，然后据此进行拟合得到合理的函数关系。

接着，我们需要预测未来一周商超的销售总量。认为需求量和成本的发展趋势会延伸到未来一周，可以由时间序列分析进行预测，得到需求量和成本在未来一周的预测值。

最后我们需要给出合适的定价策略使得商超的收益最大，收益是由利润率以及销售量决定的，而利润率又由定价与成本价决定，成本价同样可由时间序列分析进行预测，利用预测得到的销售量及成本价就可以推出最优的定价，从而实现商超利润最大化。

2.3 问题三的分析

首先对数据进行筛选处理，通过附件 2 中的数据将 6 月 24 日至 30 日可售的品种筛选出来，并利用附件 4 建立了可售品种的保质期和损耗率的关系。接着，由于商超对可售单品总数各单品订购量的最小陈列量进行了要求，因此我们通过保质期和进货量计算得到约束条件，并将满足约束条件的品种筛选出来。

接着，我们在保证每个单品基础需求量的情况下，运用背包问题的贪心算法，得到每个品类中各个单品的进货策略以及定价方案从而实现商超利润最大化。

2.4 问题四的分析

生鲜经营是商超中一个十分重要的板块，而蔬菜类又是最为灵活，变化最快的一个类别，通过采集各类数据，掌握规律并据此来制定合理补货和定价决策对于商超盈利十分关键。我们可以通过查阅文献，列举所有可能影响补货和定价的因素，并给出具体的意见以及理由作为支持。

三、 模型假设

（1） 假设销售时间间隔为 120s 的购买订单是不同的。在连续的订单销售中，若两单销售时间间隔小于两分钟，我们认为其是一次性购买的，以区分订单。

（2） 假设某商品销售量与其在销售订单中出现的次数是正相关的，我们可以通过其在假设(1)的分割订单中某商品出现的次数反映出其销售量。

（3） 假设需求量和成本的发展趋势会延伸到未来。我们假设客户的需求量可以通过随时间的变化进行预测，利用时间序列分析即可得出未来的销售量。

（4） 假设该商超的蔬菜定价仅仅采用成本加成定价法来定价，不采用其他的定价方式。

（5） 假设蔬菜的损耗率在一个平均值附近呈正态分布，认为与平均值差在 3σ 以内的蔬菜保质期至少为一天。

四、 符号说明

符号	说明
$Price$	蔬菜定价
$Profit$	利润
C	成本
r	成本加成率（利润率）
N	销售量
N_D	需求量
T_b	保质期
R_d	损耗率
$\overline{R_d}$	损耗率平均值

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 数据预处理

观察附录 2 中的数据，我们发现存在两类销售类型，分别为“销售”与“退货”。由于在接下来的问题中，需要求解的模型均只与完成“销售”了的商品相关，因此我们需要将标记为“退货”的数据及其对应的“销售”数据删除。

注意到附件 2 的每一条销售流水明细中，均包括“销售日期”“扫码销售时间”“单品编码”“销量(千克)”“销售单价(元/千克)”“销售类型”与“是否打折销售”这七条信息，将“退货”数据与“销售”数据的信息进行匹配，我们可以得到标记为“退货”的数据极其对应的“销售”数据。其中，匹配信息的条件为：二者的“单品编码”“销量(千克)”、“是否打折销售”、“销售单价(元/千克)”及“销售日期”相同。

但在这种情况下我们识别出了许多重复结果，是由于当天同品类退货次数过多导致的，故我们在匹配某个退货单时，仅仅对当天的识别到的第一单进行剔除，并且也在数据集中进行剔除防止其被多次识别，从而我们一共识别出了 400 个相对应的销售单，61 个退货单并未识别，其可能是由于退货为前一天的售货单导致的，之后我们将 461 个退货单与识别到的 400 个对应的销售单进行剔除，进行了数据清洗。

之后我们首先将“附件 1”与“附件 2”进行合并，利用“单品编号”进行检索，并且插入了对应的“单品名称”以及“分类名称”。由于“单品名称”中含有精品以及供货商的信息，我们首先要考虑的是单品之间的销售量规律，故问题一中暂时不考虑供货商以及精品的影响，在数据中仅仅保留单品名称便于以后的分析。

但为处理后面的问题，其供货商具有一定的影响，故我们不再对单品的供货商以及精品的信息进行删除，重新将四个附件的数据合并，除了第三个附件的合并方式较为繁琐，其余如之前 1 与 2 的合并方法一致，附件 3 由于其进货日期不同，进货价也不同，所以我们利用匹配的方式，通过寻找进货时间比销售时间早且最为邻近的进行匹配，从而获得每个订单商品所对应的属于它的进货价格，进而做下一步分析。

5.1.2 蔬菜各品类及单品销售量时间分布的探究

①首先考虑各品类销售量的时间分布规律，将同一品类不同单品的销售量进行加

和，得到各品类的总销售量，各品类总销售量饼状图如图 1 所示。

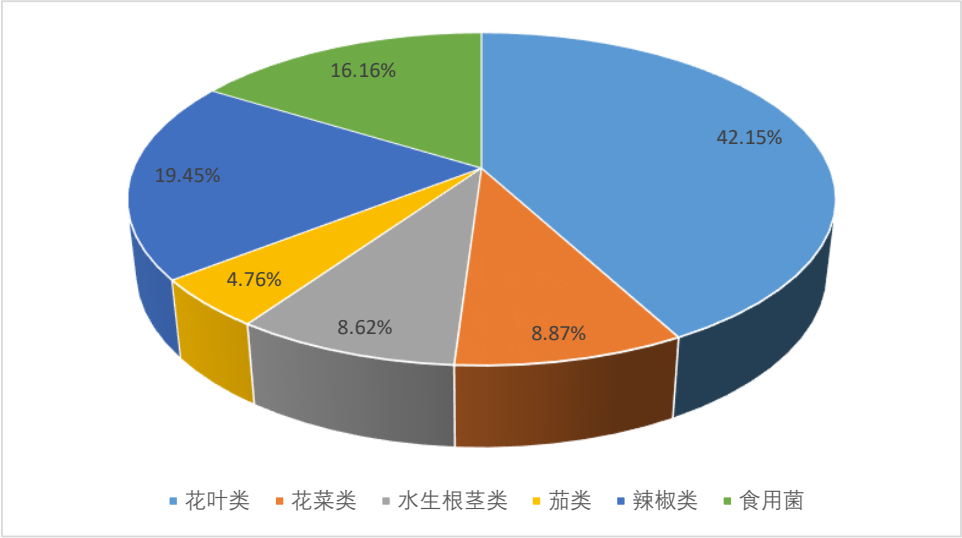


图 1 各品类总销售量饼状图

我们发现，在所有的销售流水明细的总和中，花叶类菜品的总销售量最高，达到所有蔬菜总销售量的 42.15%，其次是辣椒类，占 19.45%，紧接着是食用菌、花菜类、水生根茎类和茄类，分别占 16.16%、8.87%、8.62%和 4.76%。通过阅读文献，我们发现，在中国最受欢迎的蔬菜菜品是花叶类，在广东省，宴席上可以没有大鱼大肉，但一定要有绿色蔬菜，即花叶类蔬菜。中国绝大部分地区的人们喜欢口味较重，喜欢吃辣，湖南、江西等地的人们无辣不欢。这些原因造就了花叶类蔬菜和辣椒类菜品的高销量[1]。

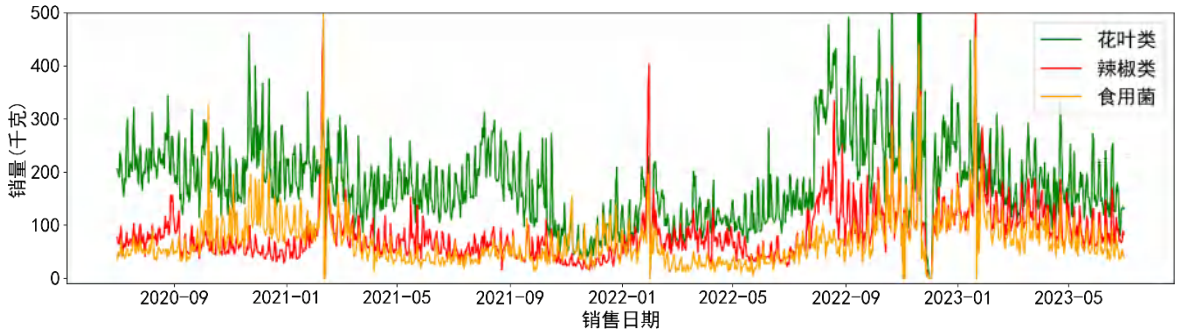


图 2 部分品类日销售总量的时间分布图(1)

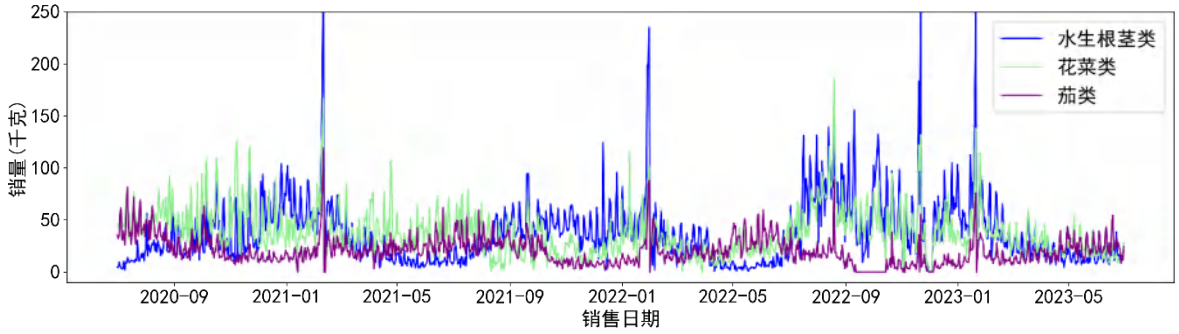


图 3 部分品类日销售总量的时间分布图(2)

考虑到销量是一个典型的随时间波动的变量，在图 2 中，我们以天为单位做出花叶类、辣椒类、食用菌这三个销量相对较高的蔬菜品类的销售数据，绘制出各品类蔬

菜销量随时间变化曲线。同理，在图 3 中，我们做出销量相对较少的花菜类、水生根茎类和茄类的销量随时间的变化曲线。根据图片可以看出，花叶类在全年明显多于其他蔬菜品类。在数据集中包含了完整的三个季节周期，可以很明显地看出这三类蔬菜品类都呈现出明显的季节性，特别是食用菌类和水生根茎类，它们在 2 月销量明显增多。

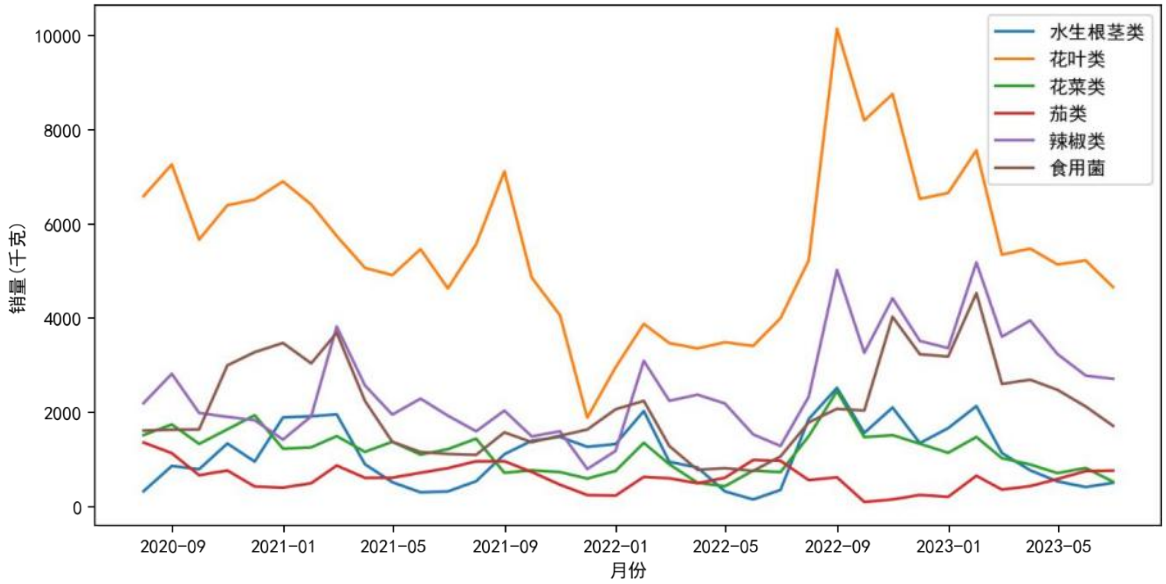


图 4 各品类月销售量的时间分布图

在图 4 中，我们将所有六类蔬菜品类的月销量随时间的变化曲线完整地画在一幅图中，可以看出其随时间的分布规律，均呈现一定程度的周期性波动，花叶类在 9 月前后销量明显增加，辣椒类、水生根茎类和食用菌类在每年 2 月前后销量增多，花菜类和茄类的季节性波动相对较小。

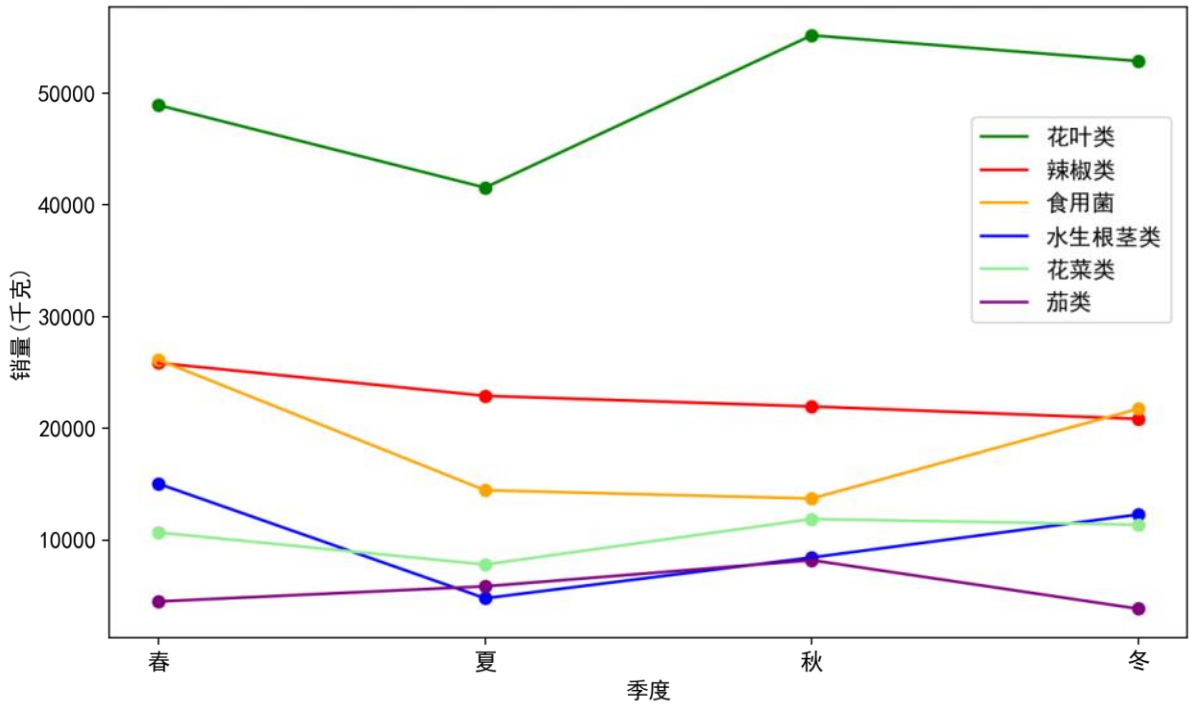


图 5 各品类极度销售量的四季分布图

从图 5 各蔬菜品类销售量随季节的变化曲线可以看出，随着时间的推移，序列的季节波动能够基本维持稳定，故我们在后续进行时间序列分析时采用加法模型。

②然后我们考虑数量较多的单品销售量的时间分布规律，通过筛选，选出了销售量最大的四个单品分别为芜湖青椒(1)、西兰花、净藕(1)和大白菜，其日销售量的时间分布图如图 6 所示。

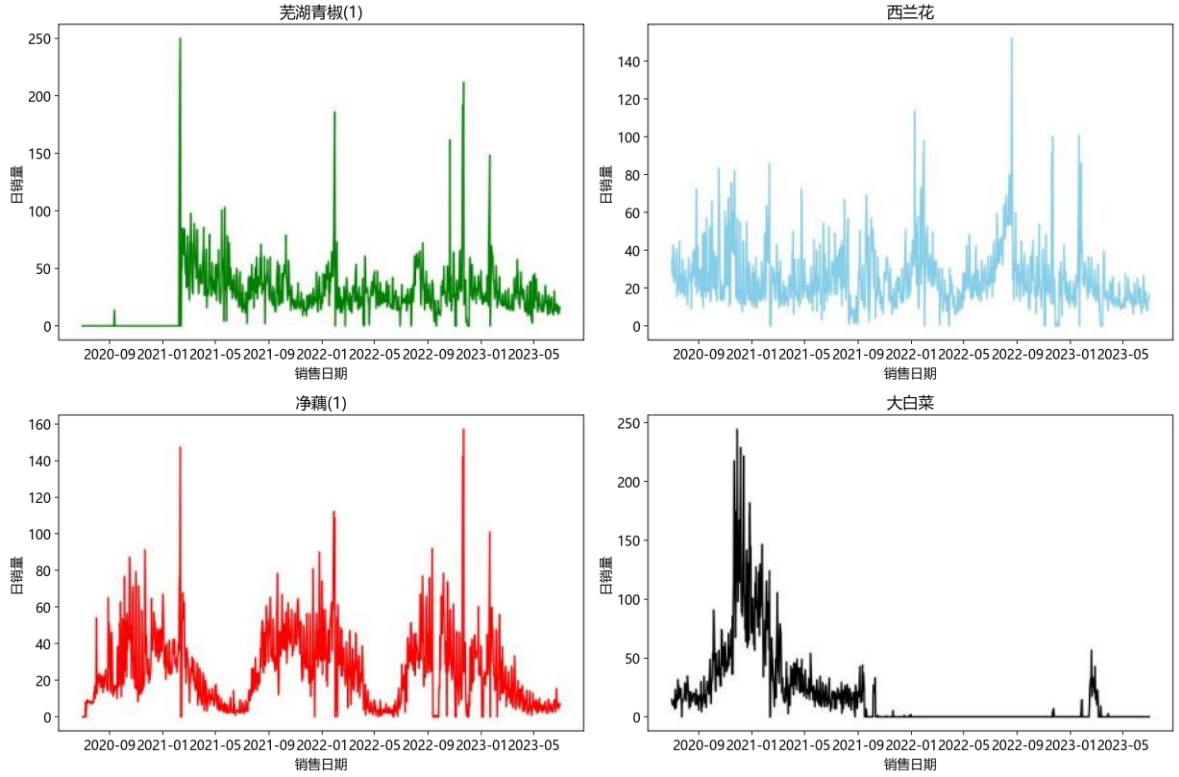


图 6 四个单品销售量较高的日销售量时间分布图

通过图示可以看出，单品的季节性分布极强，且取决于年份。可以得出，净藕的销售量完全按照季节分布，具有严格的季节规律，而后两年的大白菜基本从未进货，不同年份可能由于季节等因素使得产量较低或品质较差，从而使得其销量降低。虽然我们只作了四个总销量最大的单品的日销量时间分布规律，但大部分数据都具有类似性，其可以反映其余数据的时间分布规律。

5.1.3 蔬菜各品类及单品销售量订单分布与相互关系的探究

5.1.3.1 斯皮尔曼相关系数

研究变量之间的相关性通常用到的分析方法是皮尔逊相关分析和斯皮尔曼相关分析， 其中皮尔逊相关分析要求变量服从正态分布，而蔬菜各品类及单品销售量数据并不服从正态分布，因此我们选择斯皮尔曼相关分析方法对蔬菜各品类及单品销售量进行两两分析，找到其中的关联关系。斯皮尔曼相关系数定义式为

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{1}$$

其中，d_i为两个变量之间的等级差。我们依据该式对蔬菜品类两两之间进行斯皮尔曼相关性分析，并将所得的相关系数以热力图的方式呈现，如图 7 所示。

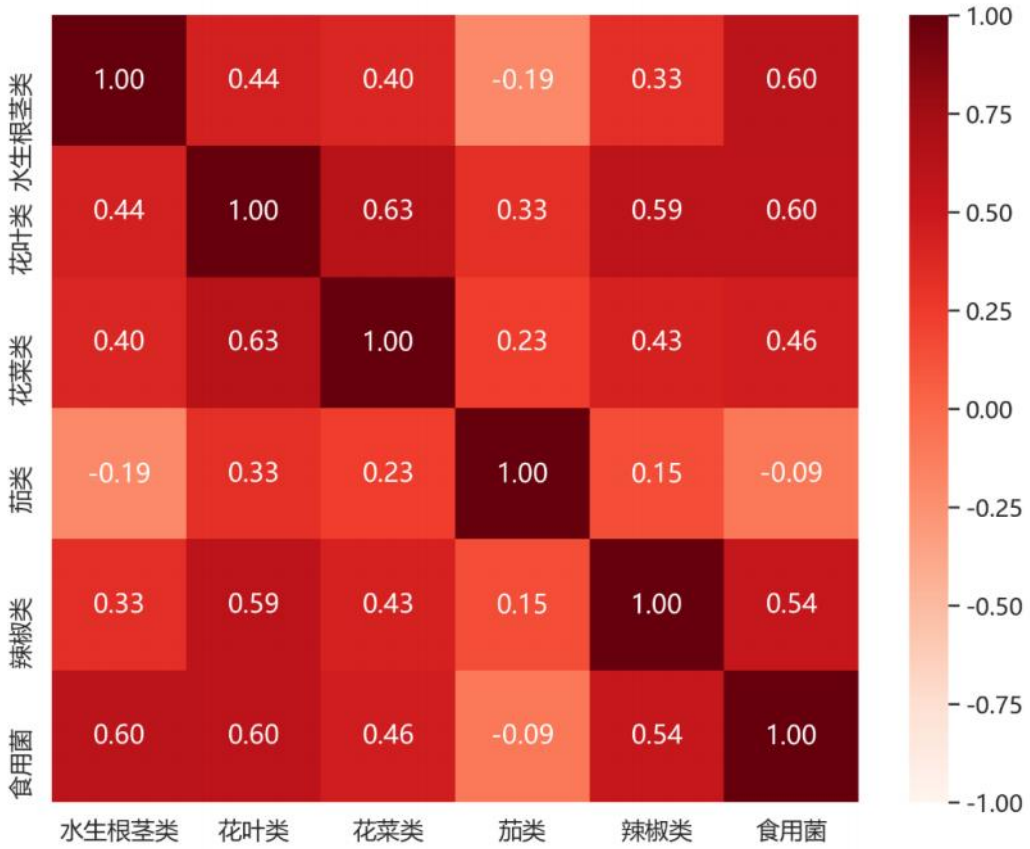


图 7 各品类之间的斯皮尔曼相关系数分布

我们认为相关系数绝对值达到 0.6 以上为强相关关系，达到 0.4 为中等相关关系，0.2 到 0.4 之间为弱相关，0.2 以下为不相关，且负数相关系数为负相关。由此可见花叶类与花菜类、花叶类与食用菌、食用菌与水生根茎类为强相关关系，茄类与水生根茎类、辣椒类、食用菌不相关。

5.1.3.2 基于 FP-growth 模型的订单分布与相互关系模型建立

为分析蔬菜各品类以及单品销售量的相互关系，我们将此关系重点放在顾客单次购买上，对于数据集，我们认为两次扫码购物时间间隔大于 120s 时即认为其是两个不同的顾客订单，从而将其分割成不同的订单，如表 1 所示，我们通过发掘单次订单购买中的商品关联性，进而分析其相互关系。

这个一个频率模式问题，其比较典型的有 Apriori，FP-growth 和 Eclat 三个算法，由于数据集较为庞大，我们考虑到运算效率的问题，选取 FP-growth 模型，对同一顾客订单中的不同品类或不同单品之间相互关系进行探究。

表 1 数据集

序号	商品类别
1	花叶类，花菜类，辣椒类，茄类
2	花菜类，水生根茎类，茄类，食用菌
3	花叶类，辣椒类，食用菌
4	花叶类，辣椒类，花菜类
……	……

以各品类的相关关系为例，遍历数据集，统计所有商品类别的出现次数（频次），并计算两种菜品之间的支持度，其中支持度为数据集中同时包含这两菜品的记录所占的比例，即为菜品 A 和菜品 B 同时被购买的概率，记做：

$$Support(A,B) = P(A \cup B) \tag{2}$$

接着，我们选取最小支持度，并删除频数小于最小支持度的商品进而进行下一步分析，这是因为支持度大于或等于最小支持度时才能保证该支持度有效。

在删除最小支持度的商品类别后，对每一个项集按照降序重新排列，示例如表 2 所示。

表 2 排序后的数据集

序号	商品类别
1	花叶类，花菜类，辣椒类，茄类
2	花菜类，茄类，食用菌
3	花叶类，辣椒类，食用菌
4	花叶类，花菜类，辣椒类
……	……

定义以下关联规则，根据数据集挖掘出的结果，例如 {花叶类}→{茄类}，规则的左侧称为先导，右侧称为后继。建立 FP 树，把第二步重新排列后的记录插入到 FP 树中，如果记录中的结点在树的该路径中有则该结点计数加一，否则分支。初始状态下，FP 树只有一个空的根节点，我们进一步插入第一个项集，结果如图 8 所示。

图 8 插入第一个项集后的 FP 树

插入第二个项集，由于起点不是以花叶类为开始，而是花菜类，故在在根节点另起一个新的分支。同理，剩余项集依次插入 FP 树，可得其部分结果如图 9 所示。

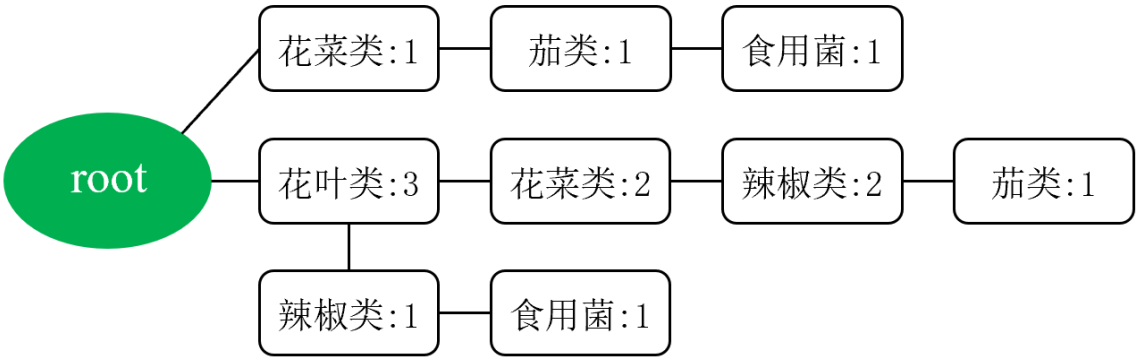


图 9 构建完的 FP 树

从 FP 树中找出频繁项集。图中下方一列叫做头指针表，树中相同名字的节点要链接起来，链表的第一个元素就是头指针表里的元素。虚线相连的表示同一个商品，各个连接的数字加起来就是该商品出现的总次数。

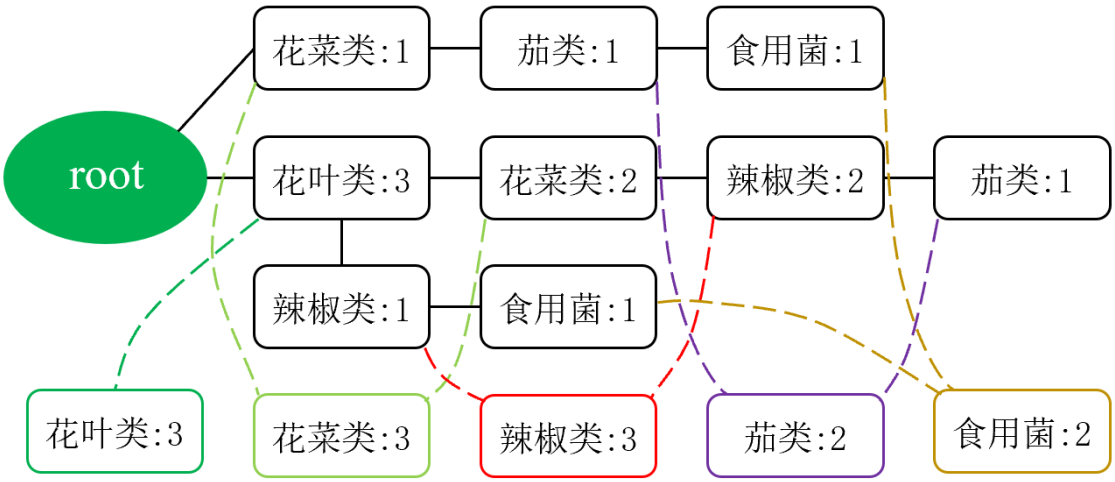


图 10 寻找频繁项集示意图

找到频繁项集后，我们利用其算出可信度和支持度与提升度，进而利用较高的参数即可得到相关性较强的商品类别，进而分析其相关关系。其中，可信度（置信度）为菜品 A 购买的同时，菜品 B 也被购买的概率，记做

$$Confidence(A \rightarrow B) = P(B \mid A) = \frac{Support(A,B)}{Support(A)} \tag{3}$$

提升度表示购买菜品 A 对买菜品 B 事件出现影响程度，以 1 为衡量标准，大于即为促进，等于即为独立，小于即为抑制，记作：

$$Lift(A \rightarrow B) = \frac{Support(A,B)}{Support(A) \cdot Support(B)} \tag{4}$$

同时我们定义最小置信度与强关联规则。项集之间的置信度必须大于等于最小置信度才能证明该置信度有效。同时满足最小支持度和最小置信度规则的规则称为强关联规则。

5. 1. 3. 3 基于 FP-growth 模型的订单分布与相互关系模型求解

我们以 120s 的分割间隔将单个销售流水明细进行订单合并，从而得到每次购买的商品，从而进行模型求解，由于数据集太多，故我们仅仅考虑两个商品类别或名称之间的关联规则。

1.蔬菜商品品类的订单分布与相互关系

①蔬菜商品品类的订单分布规律

由于数据集庞大，我们以最小支持度和最小置信度均为 0.05 的条件下，首先对商品类别进行了求解，在这些订单中，其每个类别的支持度如下图所示，支持度体现了其在某个订单中出现的概率。

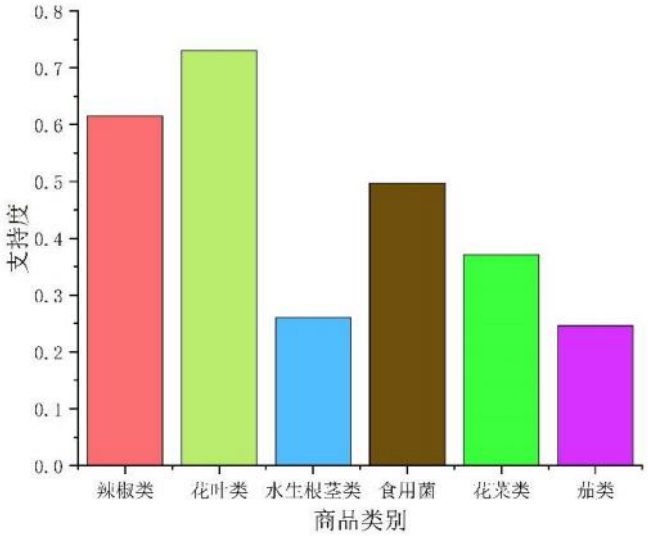


图 11 各个商品类别的支持度

可见，花叶类的支持度最高，其在三个订单中就会出现两个，这也说明其单品种类多，而且基本每家都会购买，水生根经类和茄类的购买订单较少，说明其相对小众化，并非家家喜爱。由于某个商品类别的支持度表示购买其的订单比例，故从图中就能变相的反映出各种类别销售量的分布规律。

②蔬菜商品品类之间的相互关系

对于相互关系，我们应当考虑的是置信度，置信度表示关联的可靠性，其越高，两者越具有关联性。我们将结果呈现出热力图的形式，结果如图 12 所示

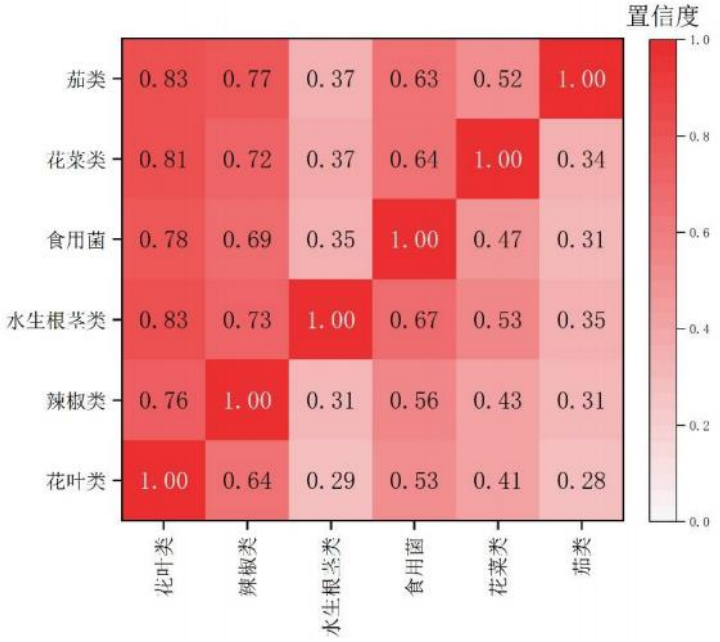


图 12 各个商品类别之间的置信度

其左侧为品类 A，底部为品类 B，数值的大小代表买完品类 A 后再购买品类 B 的概率，能够直观地看出购买东西的关联性，图中颜色越深，其关联性越强，可以看出，基本上购买任何商品种类的人均会购买花叶类，可见花叶类与各个菜系的组合繁多，而基本上很少有人会一起购买水生根茎类和茄类，其原因不仅仅是这两类商品的支持度低，还有可能因为不同类别的蔬菜难以混合搭配，比如辣椒炒菜是一个十分大众的做菜方式，但很多菜无法和茄类、水生根茎类很好地搭配。

2.蔬菜单品的订单分布规律与相互关系

①蔬菜单品的订单分布规律

由于单品比类别的规模更多，故对于每个单品的支持度，其均较小，我们保证模型的一致性，同样设置最小支持度为 0.05，以及最小置信概率为 0.05，建立 FP 树，其满足模型要求的单品极其对应的支持度如下图所示。

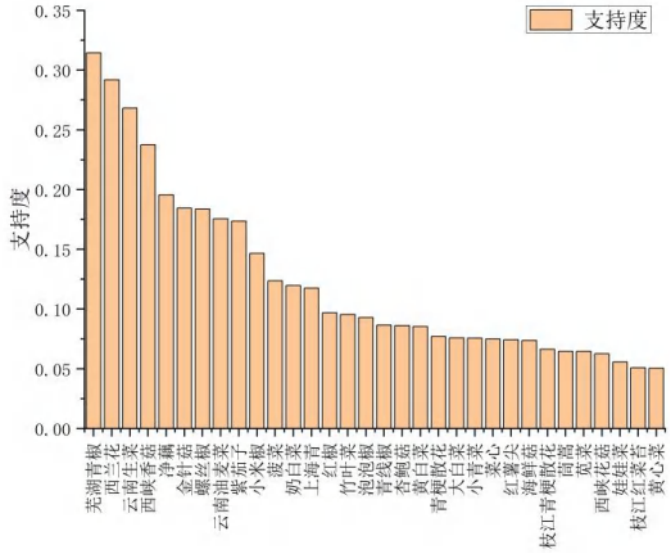


图 13 部分单品的支持度

可见，青椒、西兰花和生菜支持度均大于 0.25，基本在四单中人们就会买一次，其受众广泛，在更小的支持度范围内的单品仍然有部分人购买，但购买次数较少，在图 13 中并未列出的单品的购买率已经过低了，其受众较小，故可以反映出商品销售量随着单品的分布规律。

②蔬菜单品之间的的相互关系

此时，由于单品过多，故很难考虑所有的组合可能性，故我们在此仅仅罗列支持度较大的几个单品相互之间的关联性置信度，其如图 14 所示。

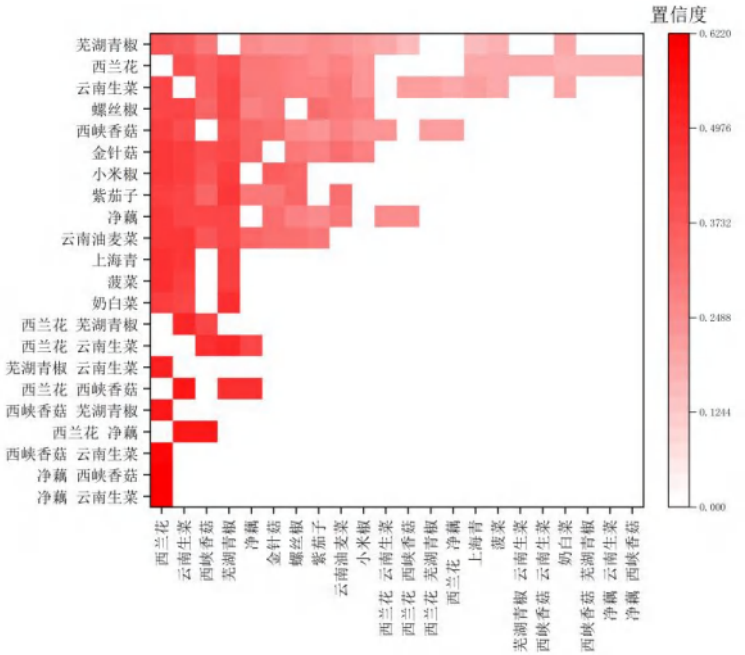


图 14 部分单品之间的置信度

在商品类别中，我们已经对其进行解读过了，在此仅仅对某些数据进行分析，我们可以看出，在购买各个单品时，基本都会购买西兰花、云南生菜等，可见这些菜品能够搭配多种菜系，并且深受喜爱。而对于图中并未列出的单品，其购买率较低，我们并未放出。

5.2 问题二模型的建立与求解

由题可知，该商超的蔬菜定价一般采用“成本加成定价”的方法[2]，即按产品单位成本加上预期单位利润来制定产品价格的方法。在问题二的求解中，我们使用成本加成率（利润率）来确定价格和利润的大小，即价格有下式给出

$$Price = C + C \times r = C \times (1 + r) \tag{5}$$

$$Profit = C \times r \tag{6}$$

其中，$Price$为蔬菜定价，$Profit$为利润，C为成本，r为成本加成率。

在本问题中，为了能够得到利润最大的补货策略，需要通过时间序列分析模型得到成本和需求量的预测值，以及购买量和成本加成率或购买量和定价的关系。

5.2.1 基于回归分析的销售总量与成本加成定价关系的探究

对附件 2 和附件 3 数据进行处理，将附件 2 中的每一条明细与所对应附件 3 中的进货明细进行匹配，并通过计算求解得到过去三年每一天各品类的成本加成率。通过斯皮尔曼相关分析方法对蔬菜各品类销售量与成本加成率及定价进行分析，结果如图 15 所示。

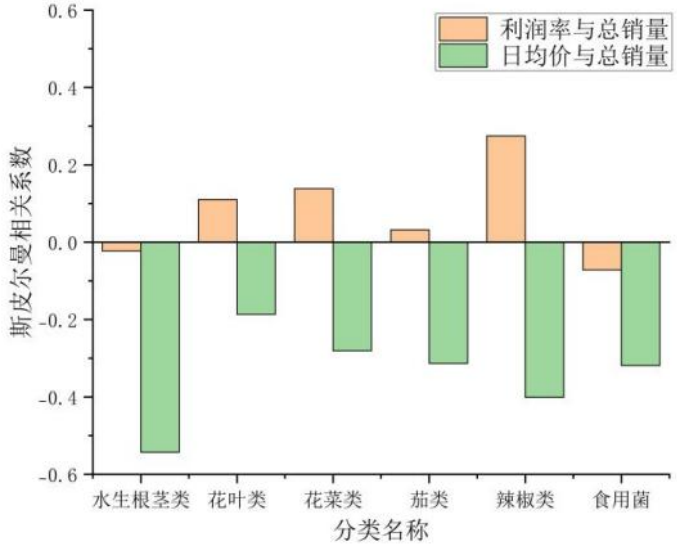


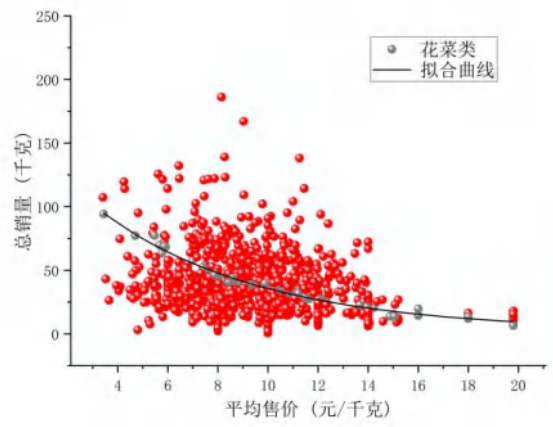
图 15 各品类的斯皮尔曼相关分析

我们发现蔬菜各品类销售量与成本加成率相关性较低，但与定价为中相关关系，因此我们利用定价与销售量采取拟合的方式得到其关系曲线。

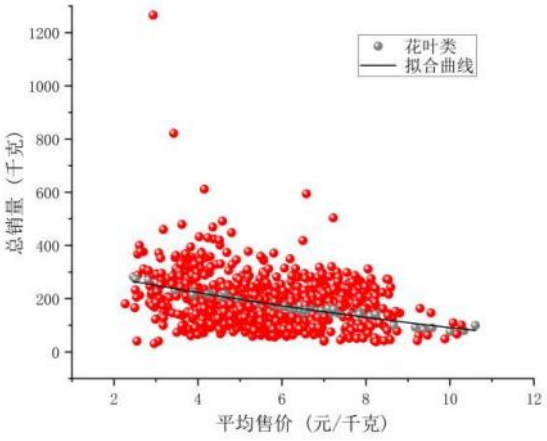
大量文献表明，定价$Price$与蔬菜各品类销售量N呈现以下关系[3][4][5]

$$N = a + \exp(-b \times Price) \tag{7}$$

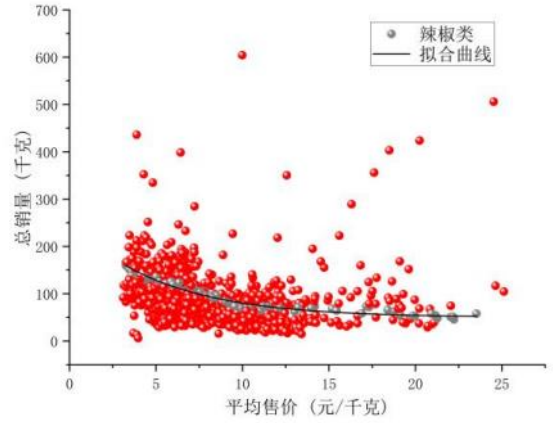
其中，a, b为待定系数。与我们发现蔬菜各品类销售量与定价相关性较弱的结果不符，究其原因，该商超的顾客人数有限，需求量有限，而论文的前提假设为该商超的需求量无限大，顾客的购买量仅仅与顾客的购买意愿有关，购买意愿仅取决于蔬菜定价，即定价越高，购买意愿越强，关系满足上式。



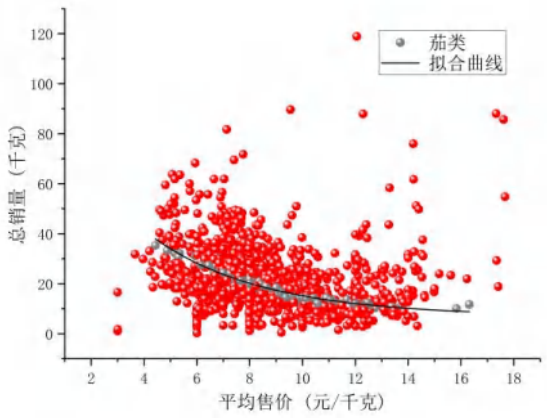
(a)花菜类



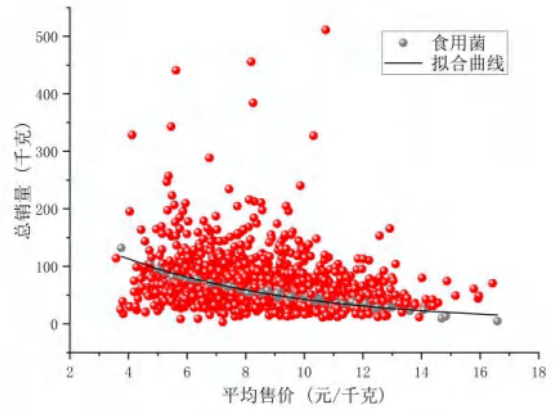
(b)花叶类



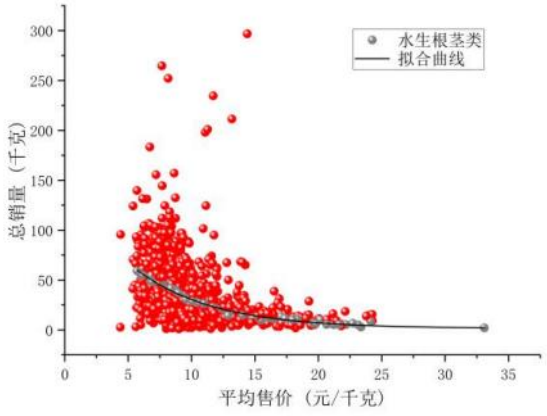
(c)辣椒类



(d)茄类



(e)食用菌类



(f)水生根茎类

图 16 各品类日总销量与平均售价的关系

因此，在本问题中，数据集按照总销量满足正态分布时，平均定价$Price$与蔬菜各品类销售量N呈现式(7)的关系。首先我们作出散点图，可以看出散点在坐标轴下方形成了一个覆盖面，且日总销量整体趋势在随售价的上升逐渐下降，除了极个别的数值极大点较为稀疏，其余的数据类似于以销量为正态分布，这也是十分合理的，结合我们日常需求来看，在价格较低时，人们总是以自身需求去购买，而家家户户的需求是满足随机正态分布的，而价格较高时，人们逐渐考虑价格因素，从而购买量下降，故

正态分布中心处的数据点可以认为即是受到价格影响而变化的数据。因此我们假定图像覆盖面的纵向宽度中心处为正态分布的平均值，其上面的点满足上述情况，使用这些点进行拟合，其拟合结果以及拟合优度R^2评分如下，散点图及曲线如图 16：

(1)花菜类：$R^2 = 0.98$

$$N = 2.43 + 157.63 \cdot \exp(-Price/6.43) \tag{8}$$

(2)花叶类：$R^2 = 0.94$

$$N = -208.15 + 549.49 \cdot \exp(-Price/16.49) \tag{9}$$

(3)辣椒类：$R^2 = 0.94$

$$N = 49.61 + 202.37 \cdot \exp(-Price/5.31) \tag{10}$$

(4)茄类：$R^2 = 0.96$

$$N = 6.98 + 88.86 \cdot \exp(-Price/4.19) \tag{11}$$

(5)食用菌：$R^2 = 0.96$

$$N = 1.94 + 215.91 \cdot \exp(-Price/6.00) \tag{12}$$

(6)水生根茎类：$R^2 = 0.97$

$$N = 1.43 + 149.40 \cdot \exp(-Price/6.10) \tag{13}$$

根据计算结果我们可以进一步计算定价，从而使得利润最大化。

5.2.2 基于时间序列分析模型的成本和需求量的预测

5.2.2.1 时间序列分析模型的建立

考虑到成本和需求量均与时间相关，我们采用时间序列分析模型的方法对未来一周成本和需求量进行预测。

一个时间序列可能包括以下四种影响因素：长期趋势，指的是在长时间内表现为持续向上或向下或平稳的趋势；季节变动，指的是受季节等因素影响产生的周期性波动；循环变动，是指一种较长时间的上下起伏周期性波动；不规则变动，是受偶然因素影响所产生的不规则波动。

由问题一的分布规律我们得知，成本和需求量的长期趋势T、季节变动S、循环变动C与不规则变动I是相互独立的，即时间序列是由四种因素直接叠加而成，因此我们使用加法模型对影响因素进行表示：

$$Y = T + S + C + I \tag{14}$$

使用时间序列分析模型中的 ARIMA 模型，即将自回归模型(AR)、移动平均模型(MA)和差分法结合，得到差分自回归移动平均模型(ARIMA)：

$$P_{ARIMA} = P_{ARIMA}(p, d, q) \tag{15}$$

其中，p为序列滞后的p期，反映在 AR 模型中，描述当前值与p期内历史值之间的关系，用变量自身的历史数据对自身进行预测：

$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + a_3 X_{t-3} + \cdots + a_p X_{t-p} + u_t \tag{16}$$

其中，u_t为随机干扰项。d是需要对数据进行差分的阶数，用于将非平稳时间序列进行d阶差分，转化为平稳时间序列。q是序列进行的移动平均次数，MA 模型认为随机干扰项是一个q阶的移动平均，得到

$$u_t = b_1 u_{t-1} + b_2 u_{t-2} + b_3 u_{t-3} + \cdots b_q u_{t-q} \tag{17}$$

5.2.2.2 时间序列分析模型的求解与成本和需求量的预测

首先我们利用 3σ准则剔除了较为庞大的异常数据点，我们认为这些数据点是由于附近餐厅举办大型宴席产生的，而我们着重于分析整体时间规律，故这些异常点会对我们的模型产生不利影响。然后利用时间序列模型，我们可以预测出未来一周各品类蔬菜的销售量，据此来确定补货总量，其预测数据如表 3 所示，并且我们在表后给出各自的时间序列分析，并附上所进行的时间序列分析趋势图。

表 3 品类时间序列分析预测值

预测值	分类名称	7.1	7.2	7.3	7.4	7.5	7.6	7.7
需求量(kg)	花菜类	20.604	17.388	17.009	17.205	17.298	17.288	17.252
	花叶类	135.215	138.315	138.807	138.388	138.048	137.922	137.882
	辣椒类	80.196	81.294	83.380	82.089	82.099	82.110	82.121
	茄类	22.775	22.254	22.093	22.036	22.009	21.991	21.975
	食用菌	46.991	50.790	50.793	50.796	50.799	50.802	50.805
	水生根茎类	18.985	19.555	19.723	19.889	20.052	20.213	20.373
成本价(￥/kg)	花菜类	7.93	7.93	7.93	7.93	7.93	7.93	7.93
	花叶类	3.23	3.23	3.22	3.22	3.22	3.22	3.22
	辣椒类	3.69	3.69	3.69	3.69	3.68	3.68	3.68
	茄类	4.71	4.68	4.68	4.69	4.69	4.69	4.69
	食用菌	4.01	3.87	3.84	3.83	3.82	3.82	3.82
	水生根茎类	12.36	12.28	12.21	12.14	12.07	11.99	11.93

①花菜类：

首先考虑花菜类销量，在差分为 0 阶、1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，认为该序列为平稳的时间序列。ARIMA 模型最佳参数为(2,1,1)，从 Q 统计量结果分析可以得到，Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.452，模型表现一般，但模型基本满足要求。得到时间序列图如下图所示。

考虑花菜类的成本，在差分为 0 阶时，在 1%水平上呈现显著性，拒绝原假设，认为该序列为平稳的时间序列。ARIMA 模型最佳参数为(0,1,1)，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.918，模型表现优秀，模型基本满足要求。

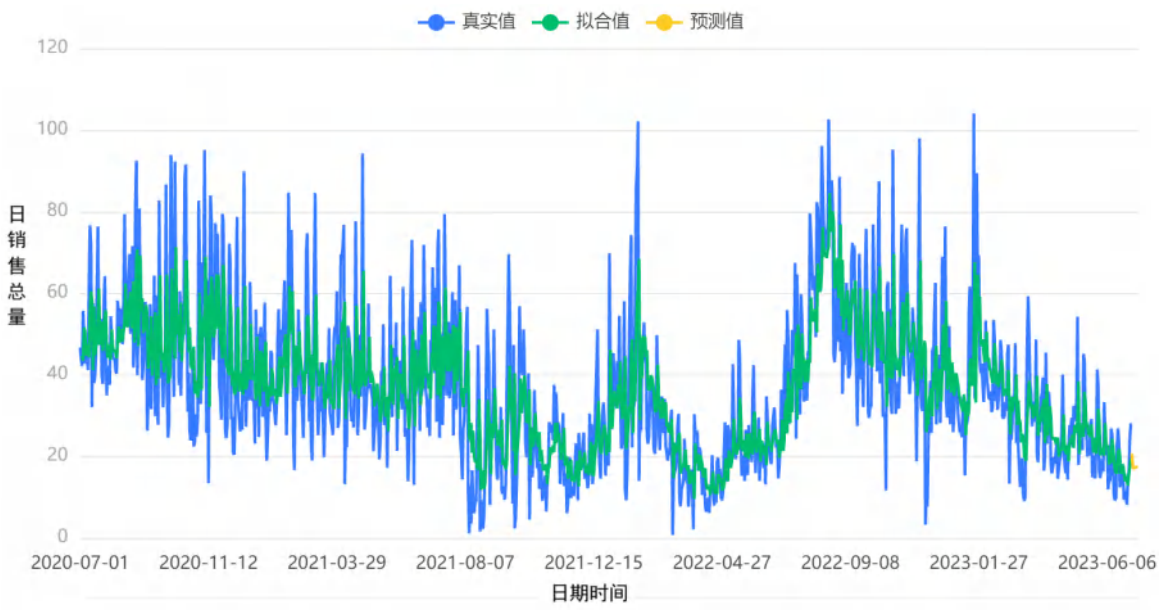


图 17 花菜类日销售总量时间序列分析

图 18 花菜类成本价格时间序列分析

②花叶类：

考虑花叶类的销售量，在差分为 0 阶、1 阶、2 阶时，显著性 P 值为小于 0.05，在 1%水平上呈现显著性，拒绝原假设，认为该序列为平稳的时间序列。ARIMA 模型的最佳参数为(2,1,1)，从 Q 统计量结果分析可以得到，Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.606，模型表现较好，模型基本满足要求。得到时间序列图如下图所示。

考虑花叶类的成本，在差分为 0 阶时，显著性 P 值为小于 0.05，在 1%水平上呈现显著性，拒绝原假设，认为该序列为平稳的时间序列。ARIMA 模型的最佳参数为(0,1,1)，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.88，模型表现优秀，模型基本满足要求。

图 19 花叶类日销售总量时间序列分析

图 20 花叶类成本价格时间序列分析

③辣椒类：

考虑辣椒类的销售量，在差分为 1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，认为该序列为平稳的时间序列。ARIMA 模型的最佳参数为(0,1,4)，从 Q 统计量结果分析可以得到，Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.629，模型表现较为良好，模型基本满足要求。得到时间序列图如下图所示。

考虑辣椒类的成本，在差分为 0 阶、1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，认为该序列为平稳的时间序列。ARIMA 模型的最佳参数为(0,1,1)，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.959，模型表现优秀，模型基本满足要求。

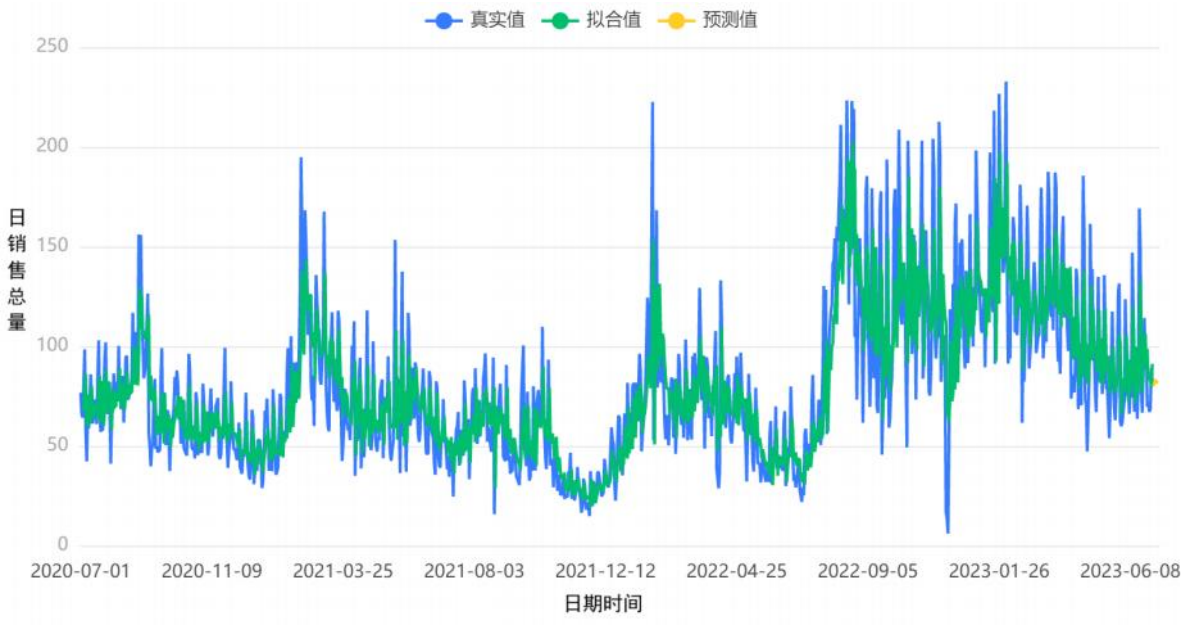


图 21 辣椒类日销售总量时间序列分析

图 22 辣椒类成本价格时间序列分析

④茄类：

考虑茄类的销售量，在差分为 0 阶、1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。ARIMA 模型最优参数为(1,1,1)，从 Q 统计量结果分析可以得到，Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.53，模型表现较好，模型基本满足要求。得到时间序列图如下图所示。

考虑茄类的成本，在差分为 0 阶、1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。ARIMA 模型最优参数为(2,1,2)，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.908，模型表现优秀，模型基本满足要求。

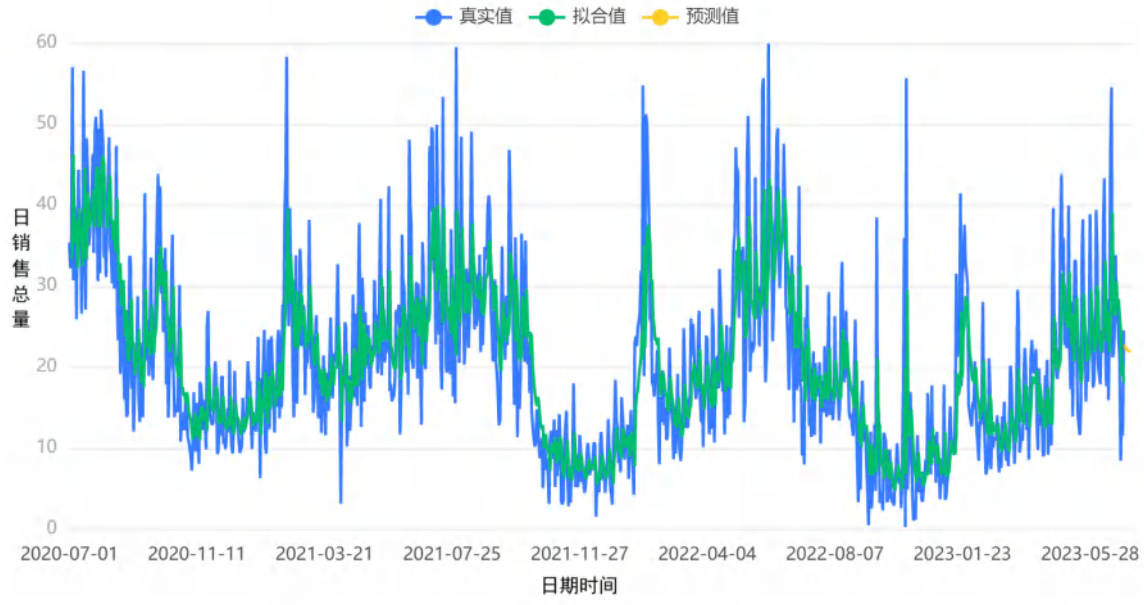


图 23 茄类日销售总量时间序列分析

图 24 茄类成本价格时间序列分析

⑤食用菌：

考虑食用菌的销售量，在差分为 1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。ARIMA 模型最优参数为(0,1,2)，从 Q 统计量结果分析可以得到，Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.597，模型表现较好，模型基本满足要求。得到时间序列图如下图所示。

考虑食用菌的成本，在差分为 1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。ARIMA 模型最优参数为(1,1,1)，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.607，模型表现较为良好，模型基本满足要求。

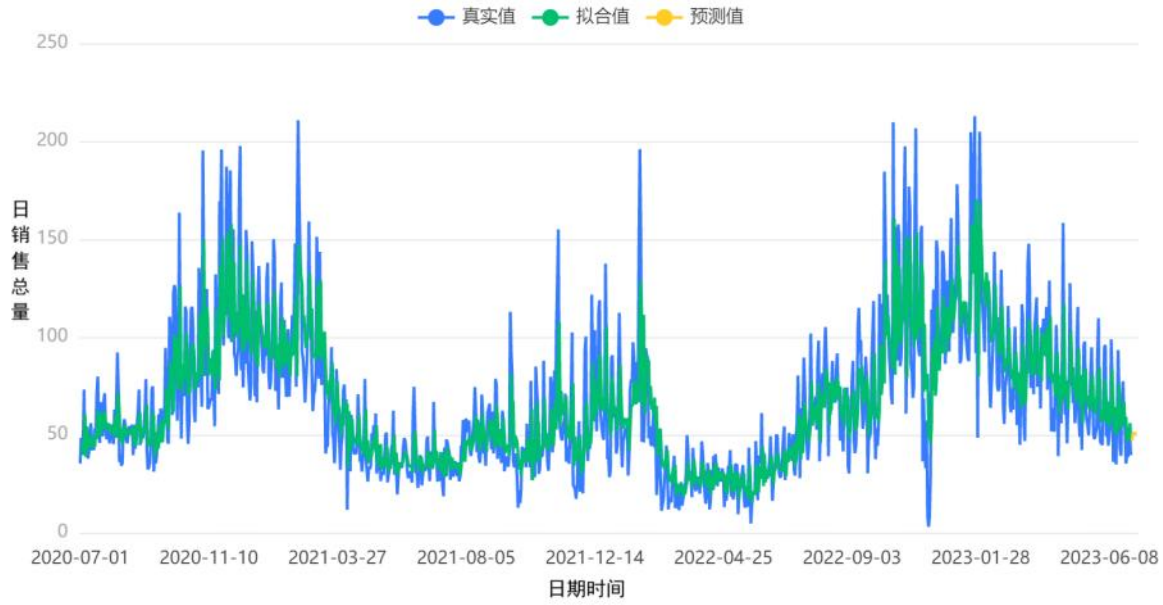


图 25 食用菌日销售总量时间序列分析

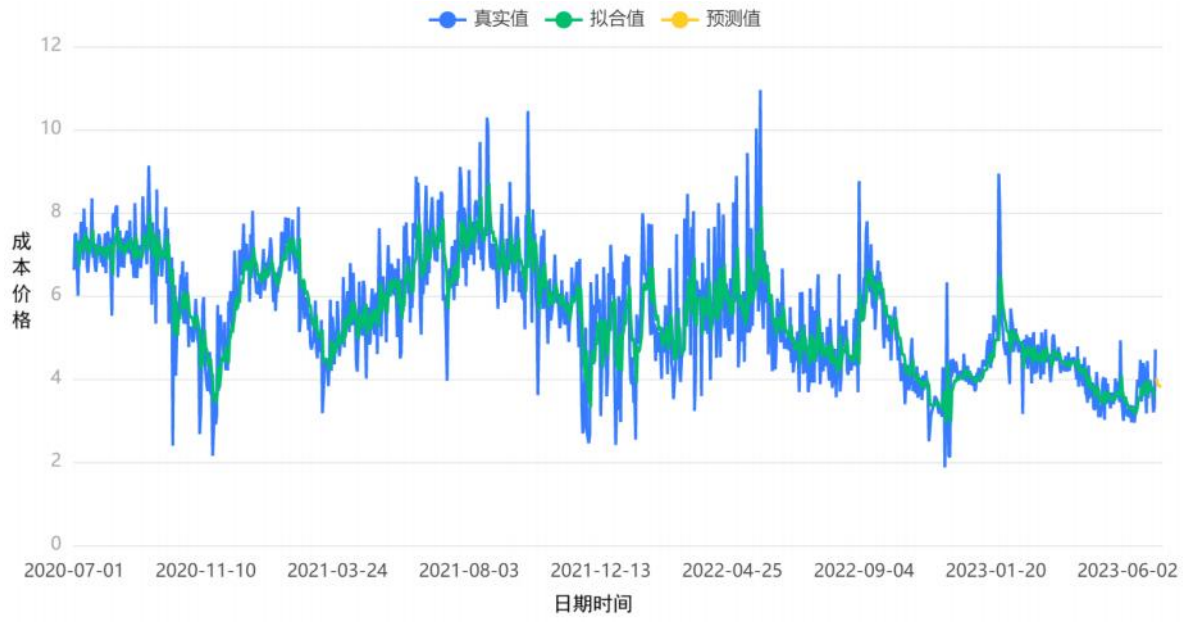


图 26 食用菌成本价格时间序列分析

⑥水生根茎类：

考虑水生根茎类的销售量，在差分为 0 阶、1 阶、2 阶时，在 1%水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。ARIMA 模型最佳参数为(1,0,2)，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.593，模型表现较好，模型基本满足要求。得到时间序列图如下图所示。

对于成本的时间序列分析，在差分为 1 阶时，在 1%的水平上呈现显著性，拒绝序列不平稳的原假设，该序列为平稳的时间序列。ARIMA 模型最佳参数为(1,0,1)，基于变量：加权进货价，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R^2为 0.667，模型表现较为良好，模型基本满足要求。

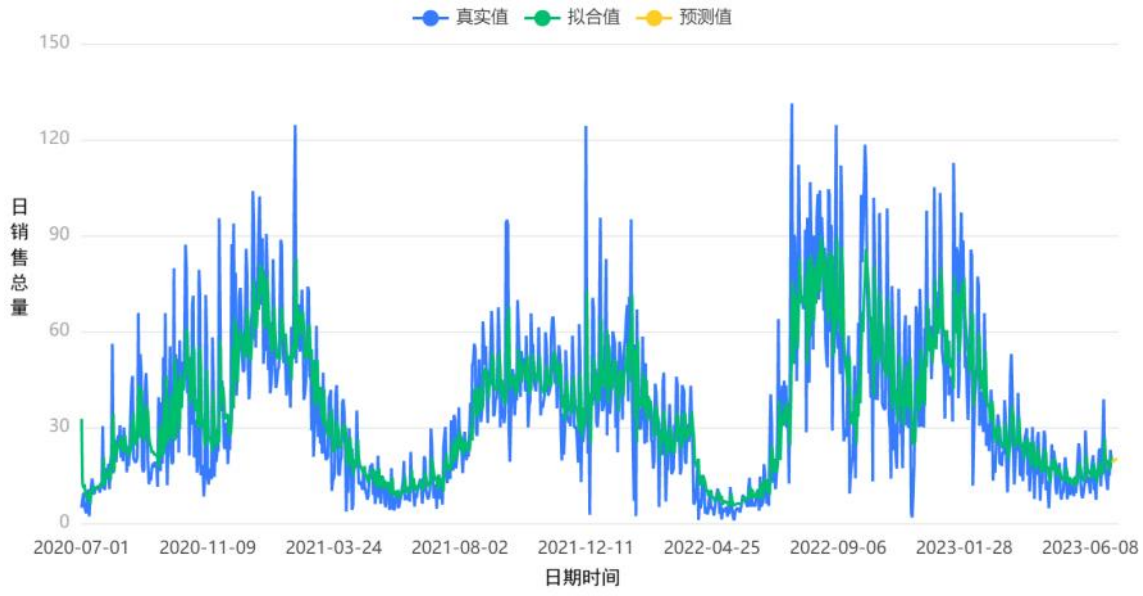


图 27 水生根茎类日销售总量时间序列分析

图 28 水生根茎类成本价格时间序列分析

5.2.3 定价策略与最大利润

在本问题中，我们的补货计划既需要满足顾客的需求量N_D，也需要使我们的利润最大化。因此，我们提出以下约束条件

$$N_D \leq N \tag{18}$$

$$Profit = \max\{Profit(N, Price)\} \tag{19}$$

通过(8)-(13)式中定价与销售量的关系以及约束条件，我们得到最佳定价策略如表 4 所示，成本加成率由式(6)给出。

表 4 各品类定价策略与补货策略

计算值	分类名称	7.1	7.2	7.3	7.4	7.5	7.6	7.7
成本加成定价（￥/kg）	花菜类	13.89	15.14	15.31	15.22	15.18	15.19	15.20
	花叶类	7.75	7.61	7.58	7.60	7.62	7.62	7.63
	辣椒类	10.03	9.85	9.51	9.71	9.71	9.71	9.71
	茄类	14.38	14.52	14.57	14.58	14.59	14.60	14.60
	食用菌	9.40	8.92	8.92	8.92	8.92	8.92	8.91
	水生根茎类	13.06	12.87	12.81	12.76	12.70	12.65	12.60
进货量（kg）	花菜类	20.604	17.388	17.009	17.205	17.298	17.288	17.252
	花叶类	135.215	138.315	138.807	138.388	138.048	137.922	137.882
	辣椒类	80.196	81.294	83.380	82.089	82.099	82.110	82.121
	茄类	22.775	22.254	22.093	22.036	22.009	21.991	21.975
	食用菌	46.991	50.790	50.793	50.796	50.799	50.802	50.805
	水生根茎类	18.985	19.555	19.723	19.889	20.052	20.213	20.373
成本加成率	花菜类	0.75	0.91	0.93	0.92	0.92	0.92	0.92
	花叶类	-0.02	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
	辣椒类	0.27	0.24	0.20	0.23	0.23	0.23	0.23
	茄类	0.81	0.83	0.84	0.84	0.84	0.84	0.84
	食用菌	0.19	0.12	0.12	0.12	0.12	0.12	0.12
	水生根茎类	0.65	0.62	0.62	0.61	0.60	0.60	0.59

可以看出，花叶类的成本加成率是负值，但负值极小，即说明其基本上是以进货价售卖，这也是合情合理的，由于花叶类是家家户户均需要的蔬菜，并且我们可以看出其占据了大部分的销售量，而很多商超可能会选择亏本出售一些必需品来吸引顾客，从而达到引流的目的，这也是十分合理的。其余的商品利润率均为正且部分利润较大，这也能够保证在吸引顾客的同时商超还能赚取利润。

然后通过下式计算

$$Profit_{total} = \sum_{i=1}^{6} Price_i \cdot N_i \tag{20}$$

可以得到我们的进货策略的日总利润如表 5 所示。

表 5 日总利润

时间	7.1	7.2	7.3	7.4	7.5	7.6	7.7	总
总利润(￥)	3115.0	3283.1	3388.2	3266.5	3192.2	3215.6	3206.8	22667.5

5.2.4 敏感性分析

对未来的预测值是衡量时间序列分析模型性能的关键指标，我们依据 3σ准则，在原有数据集中还原异常数据，观察还原异常数据前后对未来一周销量预测值的变化大小来评价时间序列模型的敏感性。

下面给出了花菜类、花叶类、辣椒类、茄类、食用菌和水生根茎类增加异常数据前后，通过时间序列分析模型预测得到的未来一周利润率的变化率，从图中可以看出，时间序列分析模型对于花叶类与食用菌具有较强的敏感性，对于花菜类、辣椒类、茄类和水生根茎类具有较好的稳定性。

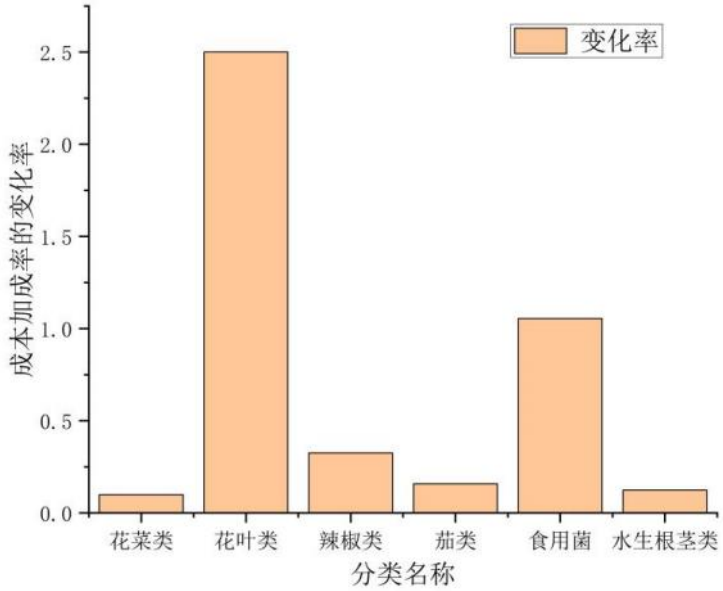


图 29 预测利润率变化率

5.3 问题三模型的建立与求解

5.3.1 约束条件与数据筛选

观察附件 2，发现仅有一部分单品可在 6 月 24 日至 6 月 30 日内销售，因此我们认为，只有这一部分单品可在 2023 年 7 月 1 日进行销售，一共 49 种单品。计算这 49 种单品的在 6 月 24 日至 6 月 30 日内的日均销售量、日均售价和日均成本价作为 7 月

1 日的基础需求量N、售价$Price$和成本价C。

由于该商超的蔬菜类商品的销售空间有限，商超希望制定新的单品的补货计划，要求将可售单品总数控制在 27 到 30 个的范围内，且各单品订购量满足最小陈列量 2.5 公斤的要求。由于蔬菜的保存时间较短，大部分蔬菜在存放一天之后不再新鲜，存放两天后不能食用，因此，我们需要引入附件 4 的损耗率，建立损耗率与保质期的关系。通过对该日可销售的 49 种单品的损耗率进行计算，发现其平均值为 10.29%。假设损耗率满足正态分布，根据 3σ准则，如图 30 所示，剔除其中的 1 个损耗率异常的单品，我们认为这 1 个单品保质期较短，易变质，导致利润降低。

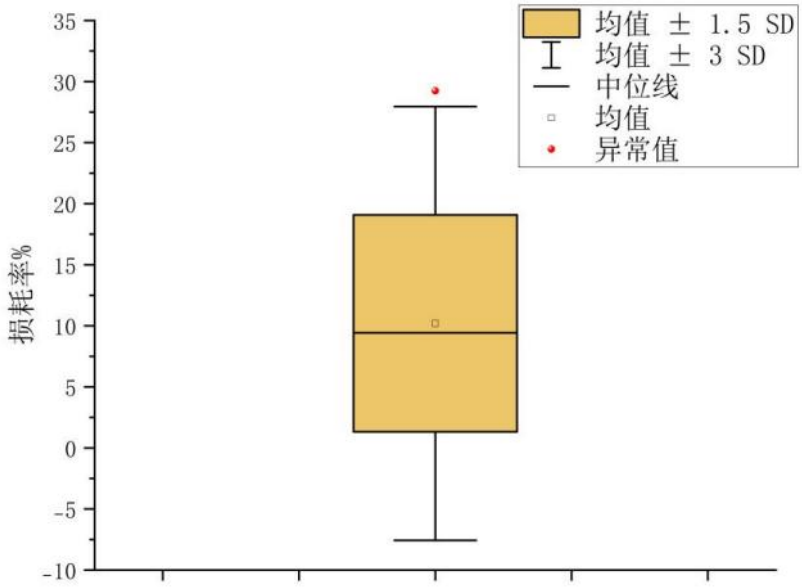


图 30 损耗率箱型图

在本问题中，我们假定损耗率满足 3σ准则的蔬菜类商品的保质期至少为 1 天，Chunyang Qian 等人认为[6]，损耗率与商品保质期的关系受到商品的存储条件、供应链效率等因素的影响，但从一般的角度来看，商品的损耗率与保质期之间存在负相关的关系，即保质期T_b越短，损耗率R_d可能越高，其可用下式表示：

$$T_b = \begin{cases} 1, R_d > \overline{R_d} \\ -\dfrac{R_d}{\overline{R_d}} + 2, R_d \ll \overline{R_d} \end{cases} \tag{21}$$

由于各单品订购量满足最小陈列量 2.5 千克，我们必须满足一下约束条件

$$T_b \cdot N \gg 2.5 \tag{22}$$

通过筛选得到 29 个满足约束条件的蔬菜单品。

5.3.2 基于背包问题贪心算法的定价策略模型建立与求解

在问题二的求解中，我们得到了 2023 年 7 月 1 日的各品类的需求量。本问题中，每种品类的单品进货量分别可以转化为典型的背包问题。以花菜类为例，背包总量为 22.863 千克，而可供装入的蔬菜单品仅为枝江青梗散花与西兰花，其基础需求量分别为 3.600 千克与 12.558 千克。在满足基础需求量的前提下，我们通过贪心算法将背包装满。

贪心算法的核心思想是每次选取当前状态下能够带来最大收益的菜品。首先根据单品的单位利润从高到低进行排序，得到“西兰花、枝江青梗散花”的排列顺序。然后从单位利润最高的单品品开始，尽量多的选择该单品，直到不能再选。继续下一个

单位重量价值次高的物品，尽量多的选择该单品，直到不能再选。重复上述步骤，直到背包的容量达到限制，或者所有物品都已经被考虑过。

结果如表 6-表 11 所示

表 6 花菜类定价策略与补货策略

单品名称	补货量(kg)	定价(￥/kg)	单位利润(￥/kg)	总利润(￥)
枝江青梗散花	3.600	12.86	12.35	44.5
西兰花	17.000	12.35	12.35	209.9

表 7 花叶类定价策略与补货策略

单品名称	补货量(kg)	定价(￥/kg)	单位利润(￥/kg)	总利润(￥)
上海青	20.215	8.00	3.87	78.2
云南油麦菜(份)	21.286	4.16	1.29	27.4
云南生菜(份)	32.286	4.46	0.89	28.6
奶白菜	6.423	4.79	2.25	14.5
娃娃菜	10.429	6.54	1.78	18.6
小青菜(1)	4.901	5.20	2.37	11.6
木耳菜	5.933	5.41	2.19	13.0
竹叶菜	13.297	3.77	1.44	19.1
红薯尖	4.507	5.29	2.14	9.6
苋菜	8.923	3.81	1.49	13.3
菠菜(份)	7.000	5.51	1.38	9.6

表 8 辣椒类定价策略与补货策略

单品名称	补货量(kg)	定价(￥/kg)	单位利润(￥/kg)	总利润(￥)
姜蒜小米椒组合装(小份)	7.000	4.72	2.26	15.8
小皱皮(份)	11.286	2.59	1.06	12.0
小米椒(份)	21.429	5.77	3.62	77.6
芜湖青椒(1)	14.233	5.20	1.83	26.1
螺丝椒	14.930	11.39	3.83	57.1
螺丝椒(份)	11.286	4.92	1.59	18.0

表 9 茄类定价策略与补货策略

单品名称	补货量(kg)	定价(￥/kg)	单位利润(￥/kg)	总利润(￥)
紫茄子(2)	10.903	6.00	2.27	24.7
长线茄	9.084	12.00	5.01	45.6
青茄子(1)	2.675	6.00	1.91	5.1

表 10 食用菌定价策略与补货策略

单品名称	补货量(kg)	定价(￥/kg)	单位利润(￥/kg)	总利润(￥)
双孢菇(盒)	10.000	5.03	1.63	16.3
海鲜菇(包)	8.857	2.75	0.80	7.0
西峡花菇(1)	11.996	24.00	8.40	100.8
金针菇(盒)	16.143	1.88	0.43	6.9

表 11 水生根茎类定价策略与补货策略

单品名称	补货量(kg)	定价(￥/kg)	单位利润(￥/kg)	总利润￥
净藕(1)	11.947	14.32	3.58	42.8
洪湖藕带	4.033	21.02	3.02	12.2
高瓜(1)	3.004	13.25	1.70	5.1

并且经过统计，此天的总利润为 971.1 元，即我们的模型得到的最优利润。

5.4 问题四的结果

经过查阅文献，为了更好地制定蔬菜商品补货和定价决策，商超可以采集以下相关数据：

①客流量数据：需了解每天或买个时间段的客流量。客流量的高低与蔬菜销售量的高低有直接的关系，客流量数据可以让商超了解一年中哪些天客流量较多[7]，则可以相应增加前一天的补货量或者开展促销活动来进一步扩大销售。同时也可了解到同一天中，客流量较大的时间段，通过保证该时间段的蔬菜供应充足。

②天气因素：需收集商超所在地天气的变化因素。因为天气因素具有难预测、难控制、对生鲜蔬菜的存储及流通极具破坏性等特征，天气因素一直以来就是导致蔬菜价格剧烈波动的主要外在因素之一。在常温、炎热天气下蔬菜更容易腐烂变质，且在高温下蔬菜的营养价值会加速流失，蔬菜质量变差则蔬菜定价也相应会降低。而雨雪天气造成的蔬菜的存储和运输成本的上升也会造成蔬菜价格的上升[8]。得到天气影响下蔬菜价格的波动规律可以更好地帮助商超进行补货和制定定价策略。

③库存状况：了解库存中各种蔬菜的存储天数。这有助于预测哪些蔬菜会因为过度库存而需要打折销售，而库存短缺蔬菜的则可以相应地提高价格销售。

④竞争对手的定价：了解当地附近同类商超的定价策略，促销方式等信息及数据。通过关注销售大环境下其他竞争商家的价格，并根据自身实力与竞争者实力的对比得出一个合理的价格。在竞争商超进行大促销期间,商超应降低部分同类商品售价以维持竞争力，或舍弃类似蔬菜品类的销售以减少蔬菜滞销的风险。

⑤供应商的配送效率、时间和可靠性：在蔬菜供应链环节中，蔬菜产品在生产环节、库存环节、运输环节都有不稳定性[9]，受天气、道路状况等因素的影响，蔬菜供货也有一定的延迟的风险，商超对此一定的把控后可以更好地调整补货及定价策略以应对供货不及时的情况。

⑥消费者反馈意见和数据：紧紧围绕消费者的需求，通过问卷调查或口头沟通等形式准确、全面地了解顾客对于商超中供应蔬菜的价格感知和需求强度，及时了解到消费者的喜好以及提出的建议。对于受顾客青睐的蔬菜品类增加补货以满足消费者的需求，根据附近居民的消费水平合理地调整蔬菜定价[10]，从而扩大商超的利润。

5.5 结果分析

5.5.1 问题一的结果分析

首先处理品类与单品的分布规律，我们作出销售总量的饼状图，其总销量由小到大排序为花叶类、辣椒类、食用菌、花菜类、水生根茎类和茄类。然后统计了各品类日销售量、月销售量以及季节销售量，通过分析得出：六类蔬菜基本都呈现出明显的季节周期分布规律，且随着时间的推移，除了几个庞大的数据点，序列的波动基本能

够维持稳定，满足时间序列分析中的加法模型条件。对于单品我们选取了销量最高的四个单品进行分析，可以分析得出部分单品比如净藕(1)完全按照季节规律性分布，而大白菜等部分单品不仅仅受到季节影响，年份影响也较大。

然后我们利用 120s 的时间间隔将出售的单个订单进行分割组合，并且利用 FP-growth 模型进行了销售量在不同类别以及单品之间的分布规律，通过得出有效的频繁项集得出各种类别以及单品在订单中出现的概率，即订单分布。对于品类，其支持度(代表出现概率)由大到小排列为：花叶类：0.73、辣椒类：0.61、食用菌：0.4973、花菜类：0.37123、水生根茎类：0.26021、茄类：0.24591；对于单品其结果如图 11，其数量较多我们不再一一列举。

对于不同品类和单品的相关关系，由于品类较少，我们首先通过计算其之间的斯皮尔曼相关系数得出：花叶类与花菜类、花叶类与食用菌、食用菌与水生根茎类为强相关关系，茄类与水生根茎类、辣椒类、食用菌不相关。

然后利用 FP-growth 统计出的频繁项集，分析其关联规则，其分别如图 12、图 14 所示，置信度越高，说明两者关联性越强，即越可能同时出现在一个订单内，比如辣椒类与多个类别均较高，说明其搭配其他菜系较容易，也深受人们喜爱。

5.5.2 问题二的结果分析

使用斯皮尔曼相关分析方法得出不同品类总销量和成本加成率与定价的相关性，发现任意蔬菜品类总销量与成本加成率为弱相关或不相关，而蔬菜品类总销量与定价为中负相关或弱负相关，因此我们选用定价与销售量采取拟合的方式得到其关系曲线，解析式如式(8)-(13)所示。

采用时间序列分析模型的方法对未来一周的成本和需求量进行预测，得到各蔬菜品类在未来一周的成本预测值和需求量预测值，模型评分较高，说明预测结果较好。通过题目中所给的约束条件，我们计算得到最佳定价策略与补货策略如表 4 所示，五个品类的成本加成率为正且较高，但花叶类的成本加成率为负，我们人为是由于商超想通过平价售卖广泛需求的蔬菜品类而吸引顾客，从而在赚取利润与吸引顾客之间达到平衡，据此最终得到七天总利润为 22667.5 元。

在敏感性检验中，我们选取评价时间序列分析模型好坏的重要依据——对蔬菜品类未来一周成本加成率的预测值作为观测指标，通过还原异常数据，观察时间序列分析模型预测得到的未来一周各蔬菜品类成本加成率的变化率，以此检验时间序列分析模型的敏感性。结果显示变化对花菜类、辣椒类、茄类和水生根茎类的影响小，而对花叶类和食用菌的影响大，说明花叶类的数据和食用菌的数据是该模型的敏感因素，且该模型对于其他四类蔬菜品类具有较好的稳定性。

5.5.3 问题三的结果分析

通过对可在 6 月 24 日至 6 月 30 日内销售的单品进行统计分析，得到单品损耗率的平均值，根据 3σ原则，剔除异常数据。建立起保质期与损耗率的关系，由各单品订购量满足最小陈列量的需求给出单品的约束条件。找出满足条件、可在 7 月 1 日售卖的蔬菜单品，如图所示。

利用背包问题贪心算法，在满足顾客需求的前提下，每次选取当前状态下能够带来最大收益的菜品，使得利润最大化。最终，我们得到最佳定价策略和补货策略如表 6-表 11 所示，其该天的最大利润为 971.1 元。

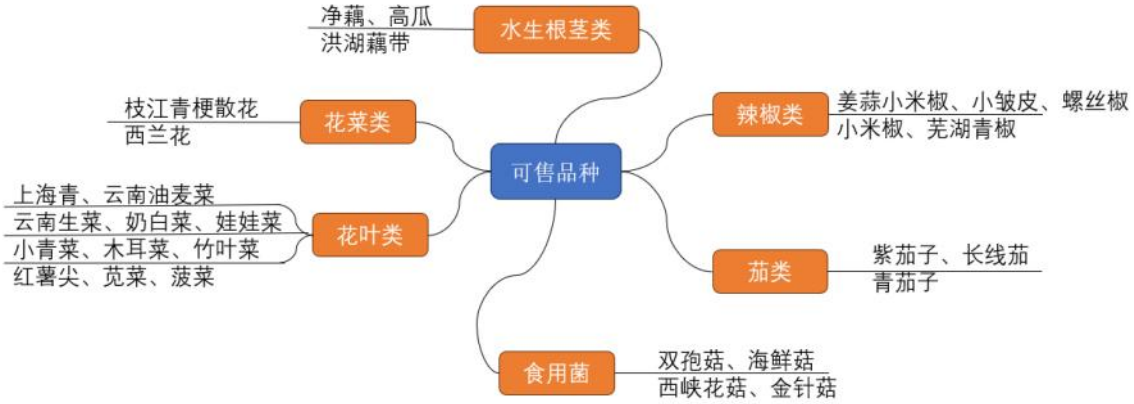


图 31 在 7 月 1 日售卖的蔬菜单品

5.5.4 问题四的讨论与分析

通过查阅相关文献，我们总结了六个商超补货和定价需考虑的影响因素，并给出了其对于商超蔬菜商品补货的定价的建议及理由：

①客流量数据，一年中客流量大的天数可以适当增加补货量，同时进行促销活动等来定价。

②天气，在高温天气蔬菜更易腐烂变质，营养价值也会加速流失，则定价也会相应降低。且一些雨雪天气造成的运输成本上升也会对定价造成影响。

③库存状况，快过期商品可以打折销售，库存短缺则可以提高售价。

④竞争对手的定价，关注其他竞争商家的价格，根据自身优势与对方优势的对比合理定价。

⑤供应商的配送效率、时间和可靠性，由于道路、天气等因素的影响造成的延迟供货商超也可以制定相应策略应对。

⑥消费者反馈意见和数据，通过全面了解消费者对供应蔬菜的价格感知及附近居民的消费水平来调整定价水平，受大家欢迎的蔬菜品类可以增加补货。

六、 模型的优缺点与改进

6.1 模型的优点

一、效率

我们采用的模型和算法效率较高。FP-growth 算法相比于同类的 Apriori 算法速度更快，其只需要对数据集遍历两次。且 FP 树将集合按照支持度降序排序，不同路径如果有相同前缀路径共用存储空间，使得数据得到了压缩。因此，我们的模型效率较高。

二、稳定性

我们的模型有可靠的数学基础，许多数学方法和理论目前在数学模型中普遍使用。对于 FP-growth，由于其具有最小置信度等限制参数，会轻易地剔除一些出现较少的数据点，则我们随意插入数据，其并不会对结果产生本质影响，因为在大范围数据集中由于出现次数过少其会被剔除。对于背包问题，我们采用了最适合该问题的贪心算法，有很强的稳定性，通过了灵敏度检验。

三、准确性

我们的模型预测结果较为准确。ARIMA 模型结合了自回归模型（AR 模型）与移动平均模型（MA 模型）的优点，既考虑了受历史影响的时间的大趋势，也考虑了一段时间内偶然事件的影响，对未来的预测有较高的准确性。不仅如此，在问题二中曲线拟合的 R-Square 均在 0.95 左右，拟合程度较好，预测数据准确。

四、普适性

我们的模型采用的方法普遍使用在国际科学研究方法和理论中，且计算代价不高，易于理解和实现。例如 ARIMA 模型可以用于处理更复杂的时间序列问题，可应用场景十分多，可以处理非线性、非平稳时间序列，具有很好的普适性。

五、实用性

商超销售蔬菜类商品的定价策略问题在生活中普遍存在，也是商超经营者最常考虑和最想解决的问题，我们的模型与实际情况密切相关，可以结合实际情况进行求解，这使得模型非常实用。

6.2 模型的缺点

一、我们在探究订单分布时，使用 FP 树第二次遍历会存储很多中间过程的值，会占用很多内存，构建较为昂贵。

二、拟合数据点是人为选择的，故存在一定的偶然误差。

6.3 模型的改进

一、我们在探究订单分布时，对 120s 为一个订单的假设过于简单，我们希望能够得到具体的订单来提高我们对订单分布探究的正确率。

二、我们在预测未来一周需求量和成本价的过程中，使用的 ARIMA 模型可能存在过拟合问题，且R^2较小，仍需要进行模型选择和参数调整。

三、我们可以在拟合部分进行提升，通过假设销售量满足正态分布从而将正态分布的中心值求出，然后利用其进行所需的拟合，能够消除偶然误差。

七、 参考文献

[1]姜晓东.中华饮食文化圈与肉制品风味设计思路研究[J].肉类工业,2011(07):33-40.
[2]王妤.从肯德基看成本加成定价法[J].经济研究导刊,2011(17):158-159+170.
[3]徐学军,何来刚.生鲜食品的二级补货系统优化研究[J].物流科技,2006(03):44-47.
[4]杨皓旭.蔬菜价格与销量的相关性分析研究——以油麦菜为例[J].食品安全导刊,2018(21):179-181.
[5]方新,袁奉娇,蔡继荣.生鲜农产品供应链的保鲜投入和货架服务优化决策及其协调契约研究[J].中国管理科学,2023,31(06):142-152.
[6]Chunyang Q ,Shuguang S ,Chenghu D , et al. A study on phenotypic micro-variation of stored melon based on weight loss rate[J]. Postharvest Biology and Technology,2023,204.
[7]吕高帆. 超市客流预测模型的研究与应用[D].北京工业大学,2020.
[8]闫英杰,陆媛媛.天气因素对蔬菜价格波动的影响研究综述[J].合作经济与科技,2015(12):130-131.
[9]毛莉莎. 供应链视角下蔬菜批发市场定价策略及产销模式研究[D].中南林业科技大学,2023.
[10]宁静,李富忠,魏杰等.山西省蔬菜价格波动特征与原因分析[J].农村经济与科技,2015,26(08):163-165+13.
[11]安琪.基于成本加成定价法的科技查新服务定价研究[J].图书馆研究与工作,2021(10):25-31+24.
[12]潘晓飞,解志恒,王淑云.考虑损失规避的生鲜品商超保鲜努力和定价的优化决策[J].公路交通科技,2022,39(06):177-185+190.

附录

表 1 支撑材料列表

文件名	功能
FP-growth.py	数据合并、处理以及 FP-growth 模型求解
问题 2 求解.py	用于求解问题二中的成本加成定价以及成本加成率
国赛 C.ipynb	用于 Jupiter 环境下的代码运行，包含数据导入、数据处理、数据导出、作图、以及计算等，为其他软件提供数据支持

表 2 论文支撑代码

用途：问题 1 数据处理

语言：Python

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth, association_rules
import openpyxl
import re
import os

'''--------------------------------数据导入--------------------------------'''
os.chdir("C:\\Users\\14716\\Desktop\\2023 国赛\\C 题\\附件")
#数据处理
#表 1
data = openpyxl.load_workbook('附件 1.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df_1 = pd.DataFrame(rows, columns=header)
print(df_1.columns)

#表 2
data = openpyxl.load_workbook('附件 2.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df_2 = pd.DataFrame(rows, columns=header)
print(df_2.columns)
#表 3
data = openpyxl.load_workbook('附件 3.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df_3 = pd.DataFrame(rows, columns=header)
```

```
print(df_3.columns)
#表 4
data = openpyxl.load_workbook('附件 4.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df_4 = pd.DataFrame(rows, columns=header)
print(df_4.columns)

# 选择附件 1 中的特定单品编号的列和其余信息列
df1_selected = df_1[['单品编码', '单品名称', '分类名称']]

# 在附件 2 的数据中，基于单品编号进行合并
df_merged_1 = pd.merge(df_2, df1_selected, on='单品编码', how='left')

# 选择附件 4 中的特定单品编号的列和其余信息列
df4_selected = df_4[['单品编码', '损耗率(%)']]

# 基于单品编号进行合并表 4
df_merged_2 = pd.merge(df_merged_1, df4_selected, on='单品编码', how='left')
# 选择附件 3 中的特定单品编号的列和其余信息列
df3_selected = df_3[['日期','单品编码', '批发价格(元/千克)']]

# 首先确保两个 DataFrame 都是按照日期排序的。
df_merged_2 = df_merged_2.sort_values('销售日期')
df3_selected = df3_selected.sort_values('日期')

df_merged = pd.merge_asof(df_merged_2, df3_selected, left_on='销售日期',
                                    right_on='日期', direction='backward', by='单品编码')

# 这里，'nearest'参数的作用是找到离左侧 DataFrame 的键最近的右侧 DataFrame 的键。
# 'by'参数的作用是让合并在'单品编码'这一列上进行。
df_merged['is_日期早于销售日期'] = df_merged['日期'] <= df_merged['销售日期']

'''
#在第一问中我们不考虑售卖单品的销售商以及精品的影响，故去除括号内的内容
pattern = r"\(.*\)"  # 正则表达式，匹配括号及其内部内容
df_merged['单品名称'] = df_merged['单品名称'].str.replace(pattern, '', regex = True)
'''
# 将销售日期转换为时间类型
df_merged['销售日期'] = pd.to_datetime(df_merged['销售日期'])

#删除退货数据
```

```
# 找到匹配的“退货”数据
returned_data = df_merged[df_merged['销售类型'] == '退货']

# 找到符合条件的“销售”数据
sales_data = df_merged[df_merged['销售类型'] == '销售']
# 创建一个空的列表，用于存储匹配到的销售数据
matching_sales_data =  pd.DataFrame(columns=sales_data.columns)

# 遍历退货数据
empty_num = 0
for index, return_row in returned_data.iterrows():
    # 使用条件筛选来找到与退货数据具有相同四个参数的销售数据
    matches = sales_data[(sales_data['是否打折销售'] == return_row['是否打折销售'])
                         & (abs(sales_data['销售单价(元/千克)'] - return_row['销售单价(元/千克)']) < 0.5)
                         & (sales_data['销量(千克)'] == -1 * return_row['销量(千克)'])
                         & (sales_data['单品编码'] == return_row['单品编码'])
                         & (sales_data['销售日期'] == return_row['销售日期'])]
    # 将匹配到的销售数据添加到列表中
    # 提取第一行
    if not matches.empty:
        matching_sales_data = matching_sales_data.append(matches.iloc[0], ignore_index = False)
        sales_data = sales_data.drop(matches.iloc[0].name)
    else:
        empty_num += 1
        print(empty_num)

print(matching_sales_data)
# 删除匹配的数据
df_merged = df_merged.drop(returned_data.index)
df_merged = df_merged.drop(matching_sales_data.index)

df_merged.to_excel('combaine.xlsx', index=False)
```

用途：问题 1 的 FP-growth 模型

语言：Python

```
'''--------------------------------FP-growth-------------------------------'''
os.chdir("C:\\Users\\14716\\Desktop\\2023 国赛\\C 题\\附件")
#合并表导入
data = openpyxl.load_workbook('modified_dataset.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
```

```
header = next(rows)  # 获取第一行数据作为列索引
df_merged = pd.DataFrame(rows, columns=header)
print(df_merged.columns)
# 将销售日期转换为时间类型
df_merged['销售日期'] = pd.to_datetime(df_merged['销售日期'])
# 将扫码销售时间转换为时间类型
df_merged['扫码销售时间'] = pd.to_datetime(df_merged['扫码销售时间'])

# 计算相邻时间的差值，并将大于 120 秒的时间间隔分割为不同的订单
df_merged['订单编号'] = (df_merged['扫码销售时间'].diff() > pd.Timedelta(seconds=120)).cumsum()

# 按照订单编号分组，并将每个订单的分类名称放入列表中，在此处
result = df_merged.groupby('订单编号')['分类名称'].agg(list).tolist()

# 将数据转化为 one-hot 编码形式
te = TransactionEncoder()

# 使用编码器转换交易数据
te_ary = te.fit(result).transform(result)

# 将转换后的数据放入 DataFrame
df_test = pd.DataFrame(te_ary, columns=te.columns_)


# 使用 FP-Growth 找出频繁项集
frequent_itemsets = fpgrowth(df_test, min_support=0.05, use_colnames=True)


# 生成关联规则
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.05)

# 输出的 rules DataFrame 包含了关联规则以及每条规则的支持度（support）、置信度（confidence）和提升度（lift）等度量。
print(rules)
print(frequent_itemsets)

# 定义一个函数，用于将 frozenset 转换为字符串
def frozenset_to_string(fset):
    return ' '.join(list(fset))  # 使用空格分隔元素

# 应用函数到指定列，并重新赋值给该列
rules['antecedents'] = rules['antecedents'].apply(frozenset_to_string)
```

```
rules['consequents'] = rules['consequents'].apply(frozenset_to_string)
frequent_itemsets['itemsets'] = frequent_itemsets['itemsets'].apply(frozenset_to_string)

# 输出频繁项集到 xlsx 文件
frequent_itemsets.to_excel('品类/frequent_itemsets_120_0.05.xlsx', index=False, encoding='utf-8')
# 输出关联规则到 xlsx 文件
rules.to_excel('品类/rules_120_0.05.xlsx', index=False, encoding='utf-8')
```

用途：数据处理 2

语言：Python(Jupiter 环境)

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth, association_rules
from datetime import timedelta
import datetime
import seaborn as sns
import openpyxl
import re
import os
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  # 或者其它支持的字体
plt.rcParams['axes.unicode_minus'] = False
os.chdir("C:\\Users\\14716\\Desktop\\2023 国赛\\C 题\\附件")
#combaine
data = openpyxl.load_workbook('combaine.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df = pd.DataFrame(rows, columns=header)
```

用途：问题 2 绘制品类的时间分布图以及斯皮尔曼相关系数热力图

语言：Python(Jupiter 环境)

```
#对每天、每一类进行求和分组
data = df.groupby(['销售日期', '分类名称'])['销量(千克)'].sum().reset_index()
# 获得唯一的产品列表和日期范围
products = data['分类名称'].unique()
dates = pd.date_range(start = data['销售日期'].min(), end = data['销售日期'].max())
# 生成所有产品和日期的组合
product_date_combos = [(date, product) for date in dates for product in products]
# 创建新的 DataFrame
data_filled = pd.DataFrame(product_date_combos, columns = ['销售日期' , '分类名称'])
```

```
# 将新的 DataFrame 和原始 DataFrame 合并，对于缺失值，填充为 0
data_filled = pd.merge(data_filled, data, on=['销售日期', '分类名称'], how='left')
data_filled['销量(千克)'].fillna(0, inplace=True)

print(data_filled)

i = 0
colors = [ 'green', 'red', 'orange', 'blue' , 'lightgreen', 'purple'] #可以自定义更多颜色
plt.figure(figsize = (20, 5), dpi=200) #设置图形大小
#循环某一类别
for category in ['花叶类', '辣椒类', '食用菌']:
    data_category = data_filled[data_filled['分类名称'] == category]
    plt.plot(data_category['销售日期'], data_category['销量(千克)'],
             label = category, color = colors[i])
    i += 1
    plt.xlabel('销售日期',fontsize=20)
    plt.ylabel('销量(千克)',fontsize=20)
    plt.xticks(fontsize=20)   # 设置 x 轴刻度字体大小
    plt.yticks(fontsize=20)   # 设置 y 轴刻度字体大小
    plt.ylim(-10, 500)
    plt.legend(markerscale=2,prop={'size': 20})
plt.show()
plt.figure(figsize = (20, 5), dpi=200) #设置图形大小
#循环某一类别
for category in ['水生根茎类', '花菜类', '茄类']:
    data_category = data_filled[data_filled['分类名称'] == category]
    plt.plot(data_category['销售日期'], data_category['销量(千克)'],
             label = category, color = colors[i])
    i += 1
    plt.xlabel('销售日期',fontsize=20)
    plt.ylabel('销量(千克)',fontsize=20)
    plt.xticks(fontsize=20)   # 设置 x 轴刻度字体大小
    plt.yticks(fontsize=20)   # 设置 y 轴刻度字体大小
    plt.ylim(-10, 250)
    plt.legend(markerscale=2,prop={'size': 20})
plt.show()
# 设置 '日期' 列为索引
data_filled.set_index('销售日期', inplace=True)

# 创建一个空的 figure 对象
plt.figure(figsize=(10,5), dpi = 200)

for i in data_filled['分类名称'].unique():
```

```
    # 对于每种分类，筛选出它的数据，然后按月求和，最后提取出 '销量(千克)' 列
    df_class = data_filled[data_filled['分类名称'] == i].resample('M').sum()['销量(千克)']
    # 绘制折线图
    plt.plot(df_class.index, df_class.values, label=i)

plt.xticks(fontsize=10)  # 设置 x 轴刻度字体大小
plt.yticks(fontsize=10)  # 设置 y 轴刻度字体大小
plt.xlabel('月份',fontsize=10)
plt.ylabel('销量(千克)',fontsize=10)
plt.legend(markerscale=2,prop={'size': 10})
plt.show()

df['季度'] = df['销售日期'].dt.month % 12 // 3 + 1
grouped_df = df.groupby(['分类名称', '季度'])['销量(千克)'].sum().reset_index()
i = 0
colors = [ 'green', 'red', 'orange', 'blue' , 'lightgreen', 'purple'] #可以自定义更多颜色
plt.figure(figsize=(10, 6),dpi = 200)
for category in ['花叶类', '辣椒类', '食用菌', '水生根茎类', '花菜类', '茄类']:
    df_category = grouped_df[grouped_df['分类名称'] == category]
    plt.scatter(df_category['季度'], df_category['销量(千克)'], color = colors[i])
    plt.plot(df_category['季度'], df_category['销量(千克)'],label = category, color = colors[i])
    i += 1
plt.xlabel('季度',fontsize=12)
plt.ylabel('销量(千克)',fontsize=12)
plt.xticks(fontsize=12)  # 设置 x 轴刻度字体大小
plt.yticks(fontsize=12)  # 设置 y 轴刻度字体大小
seasons = ['春', '夏', '秋', '冬']
plt.xticks(range(1, 5, 1), seasons)
plt.legend(markerscale=2,prop={'size': 12}, loc=(0.8, 0.5))
plt.show()

import seaborn as sns

# 假设 df 是你的 DataFrame

# 将销售日期设为索引，分类名称设为列，值为销售量。日期需要按天聚合，这里我们选择的是求和。
pivot_table = pd.pivot_table(df, values='销量(千克)', index='销售日期', columns='分类名称', aggfunc='sum')

# 计算斯皮尔曼相关系数
```

```
spearman_corr = pivot_table.corr(method='spearman')

# 用 Seaborn 创建热力图
plt.figure(figsize=(10,8), dpi = 200)
ax = sns.heatmap(spearman_corr, annot=True, cmap='Reds', fmt='.2f', vmin=-1, vmax=1)
# 移除横轴和纵轴的标题
ax.set_xlabel('')
ax.set_ylabel('')
plt.show()
```

用途：问题 2 绘制单品的时间分布规律图

语言：Python(Jupiter 环境)

```
# 将销售日期转换为时间类型
df['销售日期'] = pd.to_datetime(df['销售日期'])
# 将扫码销售时间转换为时间类型
df['扫码销售时间'] = pd.to_datetime(df['扫码销售时间'])
# 根据单品名称和销售时间分组，然后对每个组应用指定的计算公式求赚取额度和进货额度
df_single = df.groupby(['单品名称', '分类名称', df['销售日期'].dt.date]).apply(lambda x: pd.Series({'赚取额度': sum(x['销量(千克)']*x['销售单价(元/千克)']),

'进货额度': sum(x['销量(千克)']/(1-0.01 * x['损耗率(%)'])*x['批发价格(元/千克)']),

'总销量': sum(x['销量(千克)'])}))
# 重置索引，使其变回普通列
df_single = df_single.reset_index()

# 创建一个日期范围
date_range = pd.date_range(start=df_single['销售日期'].min(), end=df_single['销售日期'].max())

# 为每种商品分别创建一个产品/日期网格
grid = pd.MultiIndex.from_product([df_single['单品名称'].unique(), date_range],
                                  names=['单品名称', '销售日期'])

# 创建一个完整的 DataFrame，包括所有商品和日期
full_df = pd.DataFrame(index = grid).reset_index()

full_df['销售日期'] = pd.to_datetime(full_df['销售日期'])
df_single['销售日期'] = pd.to_datetime(df_single['销售日期'])

# 合并原始数据和完整数据集
merged_df = pd.merge(full_df, df_single, on=['单品名称', '销售日期'], how='left')
```

```
# 用 0 填充 NaN 值
merged_df['总销量'].fillna(0, inplace=True)

# 计算每个单品的总销量
sales_per_product = merged_df.groupby('单品名称')['总销量'].sum()

# 找出销量最大的四种单品
top_four_products = sales_per_product.nlargest(4).index

# 提取这些单品的销售数据
top_four_data = merged_df[merged_df['单品名称'].isin(top_four_products)]

# 绘制画布和子图
fig, axes = plt.subplots(2, 2, figsize=(15,10), dpi = 200)

# 扁平化 axes 用于迭代
axes = axes.flatten()

plt.rcParams['font.size'] = 12

colors = ['green', 'skyblue', 'red', 'black']
num = 0

# 对于每个单品，以自己的子图显示其日销量
for i, product in enumerate(top_four_products):
    # 提取单品数据
    data = top_four_data[top_four_data['单品名称'] == product]

    # 绘制线图
    axes[i].plot(data['销售日期'], data['总销量'], color = colors[num])
    num += 1
    axes[i].set_xlabel('销售日期')  # 给 x 轴添加标题
    axes[i].set_ylabel('日销量')  # 给 y 轴添加标题
    # 设置标题
    axes[i].set_title(product)

# 优化子图间距
plt.tight_layout()

# 显示图形
plt.show()
```

用途：计算问题 2 中蔬菜各品类销售量与成本加成率及定价的斯皮尔曼相关系数
计算每一品类每天的日均价、利润和利润率

语言：Python(Jupiter 环境)

```
# 将销售日期转换为时间类型
df['销售日期'] = pd.to_datetime(df['销售日期'])
# 将扫码销售时间转换为时间类型
df['扫码销售时间'] = pd.to_datetime(df['扫码销售时间'])
# 根据单品名称和销售时间分组，然后对每个组应用指定的计算公式求赚取额度和进货额度
df_single = df.groupby(['单品名称', '分类名称', df['销售日期'].dt.date]).apply(lambda x: pd.Series({'赚取额度': sum(x['销量(千克)']*x['销售单价(元/千克)']),

'进货额度': sum(x['销量(千克)']/(1-0.01 * x['损耗率(%)'])*x['批发价格(元/千克)']),

'总销量': sum(x['销量(千克)'])}))
# 重置索引，使其变回普通列
df_single = df_single.reset_index()

df_single['销售日期'] = pd.to_datetime(df_single['销售日期']) # 转换为日期类型

grouped = df_single.groupby(['分类名称', df_single['销售日期'].dt.date])[['赚取额度', '进货额度', '总销量']].sum().reset_index()

# 计算利润，添加为新的一列
grouped['利润'] = grouped['赚取额度'] - grouped['进货额度']

grouped['利润率'] = grouped['利润'] / grouped['进货额度']

grouped['日均价'] = grouped['赚取额度'] / grouped['总销量']

for category in grouped['分类名称'].unique():
    sub_df = grouped[grouped['分类名称'] == category]

    # 计算斯皮尔曼相关系数
    correlation = sub_df['日均价'].corr(sub_df['总销量'], method='spearman')
    profit = sub_df['利润率'].corr(sub_df['总销量'], method='spearman')

    print(f'{category} 的日均价与总销量的斯皮尔曼相关系数为: {correlation},利润率与总销量的斯皮尔曼相关系数为: {profit}')

print(grouped)

grouped.to_excel('money.xlsx', index=False)
```

用途：问题 2 计算单品以及品类日买入量以及进货额度

语言：Python(Jupiter 环境)

```
# 将销售日期转换为时间类型
df['销售日期'] = pd.to_datetime(df['销售日期'])
```

```
# 将扫码销售时间转换为时间类型
df['扫码销售时间'] = pd.to_datetime(df['扫码销售时间'])

# 确保损耗率，销量，进货价，售价是数字型的
df[['损耗率(%)', '销量(千克)', '批发价格(元/千克)', '销售单价(元/千克)']] = df[['损耗率(%)', '销量(千克)', '批发价格(元/千克)', '销售单价(元/千克)']].apply(pd.to_numeric)

df['买入量'] = df['销量(千克)']/(1-0.01 * df['损耗率(%)'])
# 计算买入量乘以批发价格
df['买入额度'] = df['买入量'] * df['批发价格(元/千克)']

# 根据单品名称和销售时间分组，然后对每个组应用指定的计算公式求赚取额度和进货额度
df_single_buy = df.groupby(['单品名称', '分类名称', df['销售日期'].dt.date]).apply(lambda x: pd.Series({'日买入量': sum(x['买入量']),

'日买入额度': sum(x['买入额度'])}))
# 重置索引，使其变回普通列
df_single_buy = df_single_buy.reset_index()
print(df_single_buy)
df_single_buy['销售日期'] = pd.to_datetime(df_single_buy['销售日期']) # 转换为日期类型

df_combaine_buy = df_single_buy.groupby(['分类名称', df_single_buy['销售日期'].dt.date]).apply(lambda x: pd.Series({'品类日买入量': sum(x['日买入量']),

'品类日买入额度': sum(x['日买入额度'])}))
# 重置索引，使其变回普通列
df_combaine_buy = df_combaine_buy.reset_index()

# 进货价加权平均
df_combaine_buy['加权进货价'] = df_combaine_buy['品类日买入额度'] / df_combaine_buy['品类日买入量']

print(df_combaine_buy)

df_combaine_buy.to_excel('average_money.xlsx', index=False)

# 将数据集通过"Category"列划分
groups = df_combaine_buy.groupby('分类名称')

# 为每个分组的数据写一个 Excel 文件
for name, group in groups:
```

```
        filename = str(name) + '.xlsx'
        group.to_excel(filename, index=False)
```

用途：问题 2 求解

语言：Python

```
import openpyxl
import re
import os
import math

os.chdir("C:\\Users\\14716\\Desktop\\2023 国赛\\C 题\\附件")

data = openpyxl.load_workbook('预测需求量.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df_xql = pd.DataFrame(rows, columns=header)

data = openpyxl.load_workbook('预测成本.xlsx', read_only=True)
sheet = data.active
rows = sheet.iter_rows(values_only=True)
header = next(rows)  # 获取第一行数据作为列索引
df_cb = pd.DataFrame(rows, columns=header)

cla_li = [0.0]*7
all_li = np.array([cla_li]*6)
alpha_li = np.array([cla_li]*6)

for i in range(7):
    all_li[0,i]=math.log(((df_xql.iloc[0,i]-2.43)/157.63))*(-6.43)
    all_li[1,i]=math.log(((df_xql.iloc[1,i]+208.15)/549.49))*(-16.49)
    all_li[2,i]=math.log(((df_xql.iloc[2,i]-49.61)/202.37))*(-5.31)
    all_li[3,i]=math.log(((df_xql.iloc[3,i]-6.98)/488.86))*(-4.19)
    all_li[4,i]=math.log(((df_xql.iloc[4,i]-1.94)/215.91))*(-6.00)
    all_li[5,i]=math.log(((df_xql.iloc[5,i]-1.43)/149.40))*(-6.10)
    alpha_li[0,i]=all_li[0,i]/df_cb.iloc[0,i]-1
    alpha_li[1,i]=all_li[1,i]/df_cb.iloc[0,i]-1
    alpha_li[2,i]=all_li[2,i]/df_cb.iloc[0,i]-1
    alpha_li[3,i]=all_li[3,i]/df_cb.iloc[0,i]-1
    alpha_li[4,i]=all_li[4,i]/df_cb.iloc[0,i]-1
    alpha_li[5,i]=all_li[5,i]/df_cb.iloc[0,i]-1

print(all_li,alpha_li)

df1 = pd.DataFrame(all_li)
```

```
df2 = pd.DataFrame(alpha_li)
with pd.ExcelWriter('output.xlsx') as writer:
    df1.to_excel(writer, sheet_name='Sheet1')
    df2.to_excel(writer, sheet_name='Sheet2')
```

用途：输出第三问所需数据，即 2023-06-24 到 2023-06-30 的订单

语言：Python(Jupiter 环境)

```
# 确保你的销售日期已经是 datetime 类型，如果不是，需要先转换
df['销售日期'] = pd.to_datetime(df['销售日期'])
# 将扫码销售时间转换为时间类型
df['扫码销售时间'] = pd.to_datetime(df['扫码销售时间'])

# 设置你每需要的开始和结束日期
start_date = '2023-06-24'
end_date = '2023-06-30'

# 利用日期过滤器来提取数据
mask = (df['销售日期'] >= start_date) & (df['销售日期'] <= end_date)

df_new = df.loc[mask].reset_index(drop=True)

df_new.to_excel('lim_data.xlsx', index=False)
```

用途：计算第三问所需的平均售价、平均销售量以及平均进货价

语言：Python(Jupiter 环境)

```
# 计算第三问所需的平均售价、平均销售量以及平均进货价
df['销售日期'] = pd.to_datetime(df['销售日期'])

df_week = df[(df['销售日期'] >= '2023-06-24') & (df['销售日期'] <= '2023-06-30')]

df_week['加权售价'] = df_week['销量(千克)'] * df_week['销售单价(元/千克)']

df_week['购买量'] = df_week['销量(千克)'] / (1 - 0.01 * df_week['损耗率(%)'])

df_week['加权价格'] = df_week['购买量'] * df_week['批发价格(元/千克)']

total_weighted_price = df_week.groupby('单品名称')['加权价格', '购买量', '加权售价', '销量(千克)'].sum()

# 重置索引，使其变回普通列
total_weighted_price = total_weighted_price.reset_index()

total_weighted_price['平均销售量'] = total_weighted_price['销量(千克)'] / 7
total_weighted_price['平均售价'] = total_weighted_price['加权售价'] / total_weighted_price['销量(千克)']
```

```
total_weighted_price['平均进货价'] = total_weighted_price['加权价格'] / total_weighted_price['购买量']

total_weighted_price.to_excel('price.xlsx', index=False)
```