蔬菜类商品的自动定价与补货决策

摘要

本文针对蔬菜类商品的自动定价与补货问题，通过相关系数、K-means 聚类、Prophet 与高斯过程回归预测、回归分析、TOPSIS 评价以及建立优化模型和 0-1 整数规划模型等方法，分析了单品与品类之间的分布规律及相互关系，并给出了品类与单品在不同条件下的最优定价策略和补货策略。

针对问题一，本题首先对数据进行了预处理，然后以销售量作为因变量，考虑时间因素和价格因素作为自变量对销售量的影响情况，并建立 Prophet 模型对销售量的总体趋势、年趋势、周趋势进行分析。然后使用 **K-means 聚类**对单品进行分类，并基于品类和单品的不同数据特征，分别使用 **Pearson 相关系数**和 **Spearman 相关系数**对品类和单品各自之间的相关性进行分析。最终得出可以将单品分为 24 类，且各品类和单品具有很强的季节性分布特征，价格与销售量之间有显著的负相关性，有 1 组品类之间具有强相关性，18 组单品之间具有强相关性。

针对问题二，本题首先探究了品类的销售总量与利润率的关系，得出相关性并不显著的结论，因此转到品类销售量与价格之间关系的探究，发现存在显著的相关性，建立销售量与销售价格的双对数模型进行回归；然后在对比了时间序列预测法和 7 种机器学习预测方法的基础上，建立基于 **Prophet 和高斯过程回归**的预测模型，对各品类未来一周的批发价格和需求量范围进行预测。最后根据成本加成定价法：销售价=单位成本（1+成本利润率），结合预测的需求量范围建立以收益最大为目标的收益优化模型，给出各品类未来 7 天的最优定价和补货量，具体见表 8 和表 9。

针对问题三，本题首先根据题目所给时间要求，筛选出满足时间条件的可售单品共 49 种。然后以净利润和损耗率为指标建立 **TOPSIS 评价模型**对可售单品进行打分排序，初步筛选出前 40 个单品。对筛选出的可售商品同样采用回归+预测+优化的模式，基于回归与预测的结果，最终建立以收益最大为目标的**双层优化模型**，第一层求解出各单品的最优销售价格和补货量，第二层添加 0-1 决策变量建立 **0-1 整数规划模型**，采用**带惩罚的随机扰动粒子群优化算法**得出可进货包括西兰花在内的共 **33 种**单品，以及每种单品对应的销售价格和补货量，最大总净利润为 **2971.27**,具体见表 11。

针对问题四,在问题三的基础上创新性提出货架容量限制,假设商超货架容量为 250，求解**动态规划**问题，得到 29 种商品，最大净利润为 **2845.24**；此外在 Kaggle 网站上搜集数据集并进行探索性数据分析，引入进货量、商品性质数据，从供给侧、需求侧以及供求关系角度结合微观经济学知识及运筹学进行简要分析。

关键词：TOPSIS 评价模型、K-means 聚类、Prophet、带惩罚的随机扰动粒子群优化算法、动态规划

一、 问题重述

1.1 问题背景

在生鲜商超蔬菜进货时，商品保鲜期是至关重要的考虑因素。大多数蔬菜商品保鲜期都较短，由于其品质随时间增加而下降，若当日未售出，将会进行打折销售或者直接销毁，不会留到第二天销售。所以，商超会从各商品历史销售及需求量制定每日补货计划，采用“成本加成定价”方法设定商品销售价格。因为商超销售的商品种类众多、产地也不一定相同，并且一般在凌晨 3:00-4:00 进货交易，所以每日蔬菜供应品种、数量以及价格等都在波动。商超需要在不确切知道具体单品及价格的情况下，做出当日各蔬菜补货计划。从需求侧考量，商品销售量与时间往往存在相互关联关系；从供给侧考量，由于蔬菜供应在 4 月及 10 月较丰富且商超空间有限，对于销售组合的决策变得更为重要。综上考量，合理可靠的市场分析，对生鲜商超补货及定价决策尤为重要。

1.2 待求解问题

问题一：根据附件 1 及附件 2 相关数据，分析统计蔬菜各品类及单品销售量的分布规律及可能存在的相互关联关系。

问题二：依据附件 2 销售流水明细及附件 3 商品批发价格，以品类为单位，分析各品类蔬菜销售总量与成本加成定价的关系，得出各蔬菜品类成本利润率。同时以商超收益最大为目标，制定各蔬菜品类未来一周(2023 年 7 月 1-7 日)补货及定价策略。

问题三：由于商超销售空间限制，题目要求销售单品总数在 27-33 个范围内，且为了满足最小陈列量要求，各单品订购量至少为 2.5 千克。并依据 2023 年 6 月 24-30 日的可售品种，在尽量满足各类商品需求且商超利润收益最大的前提下，进一步制定 7 月 1 日单品的补货计划及定价策略。

问题四：为了更好地制定生鲜商超蔬菜商品的补货计划及订货决策，使商超收益最大化。除了题目给出的考量因素，分析商超进行市场调研时，还可以收集哪些相关信息数据且这些数据对解决上述问题有何帮助。

二、 问题分析

2.1 问题一分析

问题一第一小问要求分析销售量分布规律。因此，首先需要对数据进行预处理，进行数据的完整性检查以便确保后续数据分析结果的准确性。观察附件 1 和附件 2 数据，分别从品类和单品两个方面分析销售量的分布规律，以销售量为因变量，考虑时间因素和价格因素对销售量的影响，分析其占比情况、年分布情况、季度分布情况、月分布情况以及价格对销售量的影响情况等。同时使用 Prophet 模型对销售量的总体趋势、年趋势、周趋势进行分析。通过上述对数据的分析，得出各品类单品的分布规律，同时可得到消费者的需求偏好及价格偏好。

第二小问需要分析各品类及单品销售量的相互关系。可以使用 K-means 聚类法对单品进行分类，同时可选取不同的相关系数分析相关性，如 Pearson 相关系数、Spearman 相关系数。

2.2 问题二分析

问题二首先要求分析品类销售总量与成本加成定价的关系，对此需要由附件 3 批发价格作为成本及附件 2 销售流水明细中给出的销售价格。可根据成本加成定价法：销售价格=单位成本（1+成本利润率），求解出各品类利润率与品类销售量关系，分析商家历史销售方案。

其次要求制定未来一周的日补货总量和定价策略。具体来说，1.对于定价策略：首先，以销售价格作为自变量，销售量作为因变量，建立一元线性回归模型，得到销售价

格与销售总量的关系。然后，建立 Prophet 模型得到未来一周品类的批发价预测值和销售总量预测范围，将预测得到的销售总量范围作为需求量。最后，通过调整利润率及预测得到的批发价格得出销售价格，带入回归模型，得到销售量。以收益最大为目标，以利润率为变量找到使商超收益最大的定价策略。2.对于补货策略：为满足供应需求，销售量需要在预测得到的需求量范围内。通过上述已进行的利润率优化所得到的销售量加上损耗数量求得补货量。

2.3 问题三分析

问题三要求给出单品的补货计划，是在问题二基础上的进一步分析。首先可以根据题目要求，先初步筛选出 2023 年 6 月 24 日-30 日的可售单品数量，然后选取合适的指标使用 TOPSIS 评价模型对筛选出的单品进行打分排序选出评分最高的前 40 个单品。针对剩下的单品， 可以建立双层优化模型，第一层是对独立的每个单品沿用第二问的思路求解最优销售量和补货量，第二层添加 0-1 决策变量，对可售单品总数进行约束，考虑到陈列单品并不一定是完好的，可以有损耗的存在，因此用补货量建立最小陈列量约束，最终使用带惩罚的随机扰动 PSO（粒子群优化算法）求解出使商超收益最大的补货量和定价策略。

2.4 问题四分析

问题四需要寻找可收集的数据，分析影响补货和定价决策的因素。可以结合探索性数据分析和经济学分析两个角度。通过搜集补充对于进货量、整体市场情况、产品全面质量信息及供应需求情况相关数据，具体分析所应用数据对于问题分析的影响及帮助。

三、 模型假设

为了更好地进行模型建立与求解，本文根据实际情况给出以下合理假设与约束条件：

（1）假设当日补货蔬菜将在当日销售或者销毁，不会留到第二天出售，即没有存储成本；

（2）假设商品损耗率在一段时间内保持一定；

（3）假设打折销售量比较稳定。

四、 符号系统

符号	说明	单位
x_{ij}	预测的第 i 种品类第 j 天的批发价格	元
t_{ij}	第 i 种品类第 j 天的利润率	/
p_{ij}	第 i 种品类第 j 天的销售价格	元
q_{ij}	第 i 种品类第 j 天的销售量	千克
p'_{ij}	第 i 种品类第 j 天的打折价格	元
k_i	第 i 种品类/单品的损耗率	/
m_i	第 i 种品类/单品的打折率	/
n_i	第 i 种品类/单品的折扣率	/
r_{ij}	第 i 种品类第 j 天的补货量	千克
a_{ij}	第 i 种品类第 j 天的需求量上界	千克
b_{ij}	第 i 种品类第 j 天的需求量下界	千克
w	各品类 7.1-7.7 的总收益	元

v_i	第 i 种单品 7.1 的收益	元
x_i	预测的第 i 种单品的批发价格	元
t_i	第 i 种单品的利润率	/
p_i	第 i 种单品的销售价格	元
q_i	第 i 种单品的销售量	千克
p'_i	第 i 种单品的打折价格	元
r_i	第 i 种单品的补货量	千克
v	进货单品总收益	元
α_i	0-1 决策变量，取 1 表示选择此单品，否则不选	/

五、 模型建立

5.1 问题一的模型建立与分析

对于各品类和单品销售量分布规律的分析，本题先对数据进行了预处理，然后以销售量作为因变量，考虑时间因素和价格因素作为自变量对销售量的影响情况，并建立 Prophet 模型对销售量的总体趋势、年趋势、周趋势进行分析。对于各品类及单品销售量之间的相互关系，本部分首先采用聚类分析对单品进行分类，分析单品之间的关系，然后基于品类和单品的数据特征，使用 Pearson 相关系数和 Spearman 相关系数分别对品类和单品各自之间的相关性进行分析。

5.1.1 数据准备

（1）数据描述性统计

附件 2 共计 878503 条数据，查看其基本统计信息。

表 1 销量描述性统计结果

指标	非缺失值数量	平均值	标准差	最小值	上四分位数	中位数	下四分位数	最大值
值	878503.000	0.536	0.400	-9.082	0.290	0.435	0.742	160.000

（2）数据预处理

本部分首先对附件 2 销售量数据进行了数据清洗，发现数据存在异常值，因此对数据的异常值进行剔除。

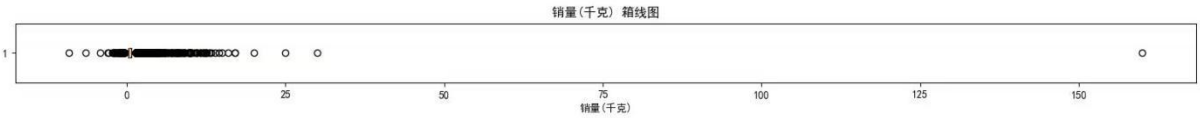


图 1 附件 2 销量箱线图

观察上图，可以观察到示例数据整体呈现右偏分布，主要集中在 0-1 之间，存在异常值。其中，最右侧值为 160 的离群点对整体均值和标准差的影响很大，可采取单独剔除的方式。剩下的异常值再按照 3σ 原则，找到异常值上下界，对异常数据进行剔除，共剔除 5114 个异常数据，剔除数据率为 0.582%，数据剩余 873389 条。

上述**先单独剔除较大离群点，再对剩余异常值点进行剔除**的方式，更能有效处理数据异常值，增强数据的准确性，这也是本文数据处理的一个思考点。

然后对数据进行完整性检查，每条数据记录都符合预期的规则和完整性要求，完整性良好。

观察附件 2 数据，每个单品并非在每一日都存在销售量，针对当日没有销售量的单

品，采取 0 值填充，保证时间上销售量的连续性，以便后续数据的处理与分析。

5.1.2 销量可视化分析

（1）各品类总销量分析

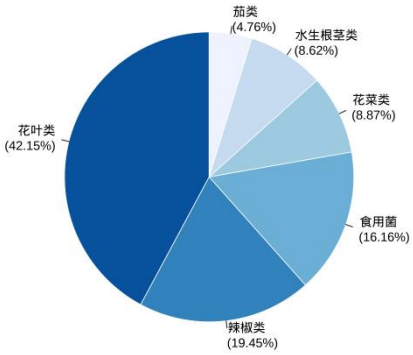


图 2 各品类总销量占比饼状图

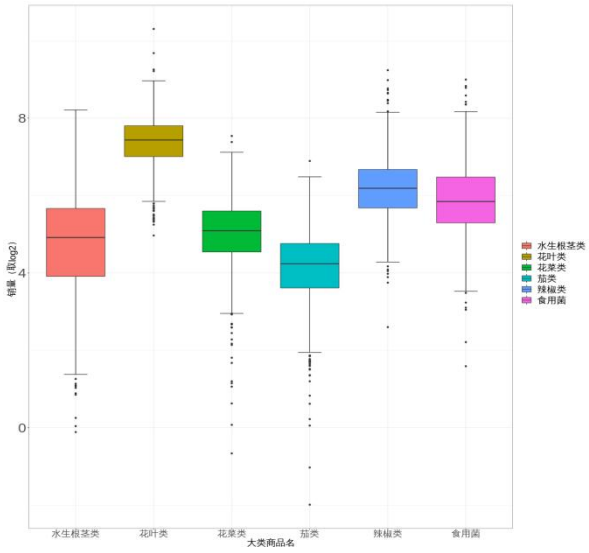


图 3 各品类总销量箱线图

对各品类的总销量占比和分布情况绘图分析，可以观察到：各品类销售总量分布不均，差异较大。其中花叶类的总销量占比第一，大约占全部品类总销量的一半，其次销量较大的为辣椒类和食用菌类，销量最小的为茄类，即消费者对花叶类的需求量最高，对茄类的需求量最低。

（2）各品类销售量的年分布、季度分布及月分布情况

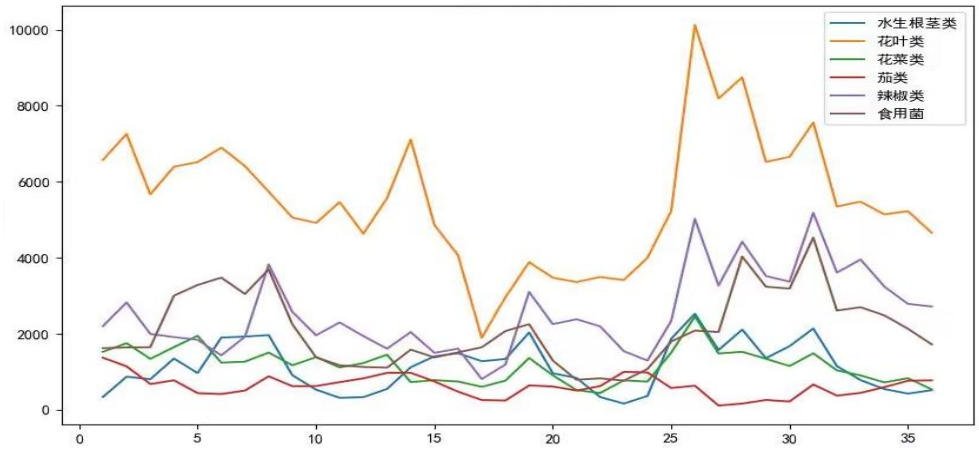


图 4 各品类销量 2020 年 7 月-2023 年 6 月销量折线图

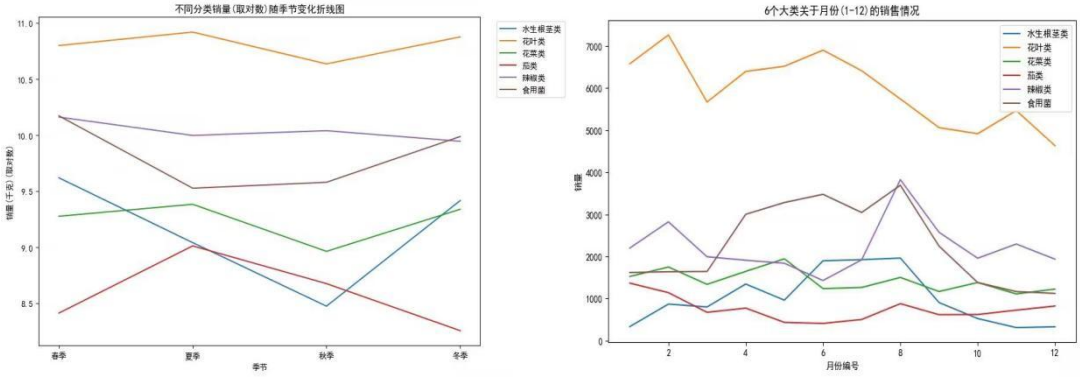


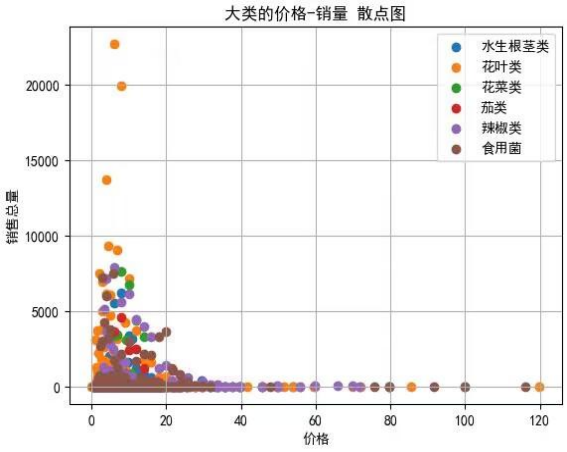
图 5 各品类销量季节变化折线图　　图 6 各品类月份销量折线图

图 3 的销量折线图横坐标为 2020 年 7 月-2023 年 6 月共计 36 个月的编号，依次从 0 到 35，纵坐标为各品类在不同年份不同月份的总销量。从折线图中可以观察到：各品类销量呈不规则波动，2020 年 7 月-2022 年 5 月各品类的总销量总体呈现下降趋势，2022 年 6 月后，各品类的总销量开始呈现上升趋势。2022 年 6 月到 9 月除茄类外，其他五类的区间分段相关性很高。

图 4 以季节作为自变量分析各品类总销量的变化情况，可以观察到：各品类总销量具有很强的季节性特征，其中花叶类、花菜类具有相同的变化趋势，具有很强的区间分段相关性，辣椒类正好与之相反。总体来看，还可以得出大部分品类夏冬两季销量较大，而秋季销量较小的结论。

图 5 细化到每个季节的每个月，可以更细致地观察各品类总销量在不同月份下的分布规律，可以观察到：各个品类在不同月份的总销量波动性较大，除花叶类外，其余品类总销量在 6 月-8 月均呈上升趋势，区间分段相关性很高，需求量高。

（3）不同价格下各品类销售量的分布情况



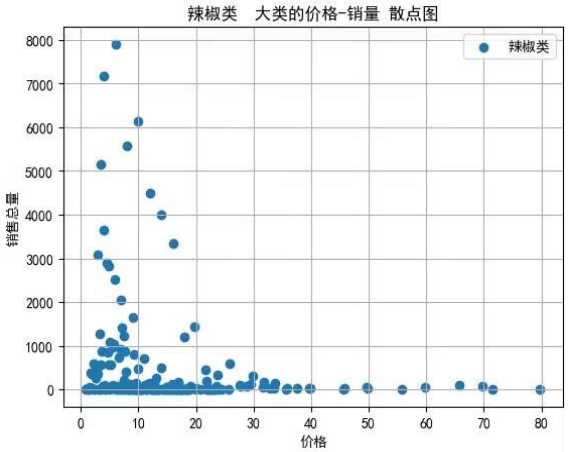


图 7 各品类价格-销量散点图　　图 8 辣椒类价格-销量散点图

以单品价格为 x 轴，销售总量为 y 轴绘制散点图，可以观察分析得出大部分单品价格都集中在 0-20 区间，这区间销量占比最多；其余价格越大，销量越接近 0。由此，容易得出价格越高销量越低。以辣椒类单品举例，同样可以发现此规律。

（4）各品类销售量趋势分析

本部分建立 Prophet 模型，通过做出各品类商品在总体趋势、周趋势、年趋势的变化折线图，对各品类在不同时间维度上的变化规律进行分析。

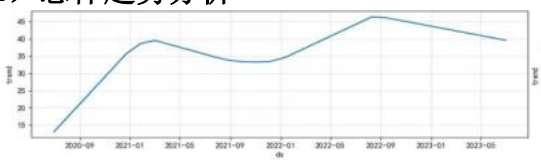
1）建立 Prophet 模型

Prophet 是一种由趋势项、季节项、节假日项和误差项组成的加法模型：

$$y(t) = g(t) + s(t) + h(t) + \varepsilon$$

其中，$y(t)$是某一时刻的预测值；$g(t)$为趋势项，表示非周期的变化趋势；$s(t)$为季节项，或者称为周期项，一般以周或年为单位；$h(t)$即节假日项，表示时间序列中潜在的、具有非周期性的节假日对预测值的影响；ε为误差项，或称为剩余项，表示模型未预测到的波动，误差项ε服从高斯分布。

2）总体趋势分析



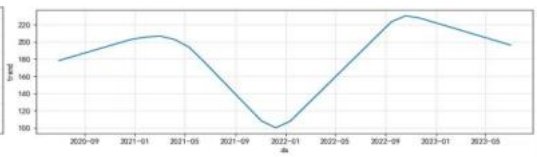


图 9 水生根茎类总体趋势折线图　　图 10 花叶类总体趋势折线图

上图给出部分趋势折线图。通过观察各品类销售量随在 2020 年 7 月至 2023 年 6 月范围内随时间变化的总体趋势图发现，除茄类及食用菌外，其余品类商品在 2021 年 3 月到 2023 年 6 月区间分段相关性很高。除茄类商品外，其余品类蔬菜均在 2021 年末 2022 年初时达到极小值，推测 2021 年下半年蔬菜市场销量整体呈收缩趋势。水生根茎类总体上大致呈上升趋势，并在 2021 年 3 月及 2022 年 9 月达到极大值。茄类销量一致保持下降趋势，可推测出顾客对于茄类商品喜爱度随时间显著下降。食用菌销量随时间呈先下降后增加趋势，从 2022 年初起，销量一致呈持续向好的增长趋势。

3）周趋势分析

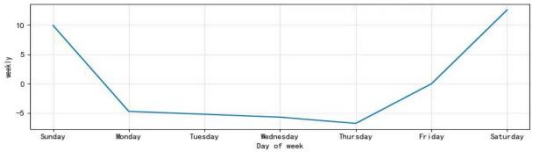


图 11 水生根茎类周趋势折线图

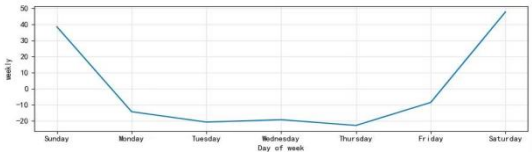


图 12 花叶类周趋势折线图

上图给出部分周趋势折线图。针对各品类蔬菜销量周趋势变化，可以看出各品类商品销量均在周末两天达到最大值，在周一到周五均表现出下降趋势。由此可以推测，顾客偏好在周末囤购蔬菜，所以商超需要在周末两天适当多采购蔬菜以满足销售需求。

4）年趋势分析

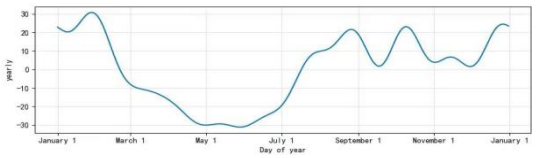


图 13 水生根茎类年趋势折线图

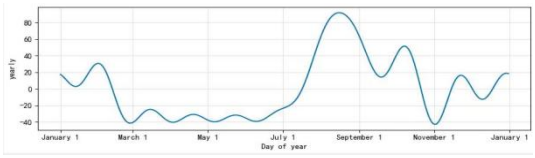


图 14 花叶类年趋势折线图

上图给出部分年趋势折线图。分析各品类年趋势变化可得，在每年 2 月达到第一个极大值，可以推测出是由于春节的影响导致各类商品销量均增加。同时在上半年，除茄类外，销量大致呈现下降趋势，可以推测这段时间为淡季。同时结合趋势图及题目给出的供应品种在 4 月至 10 月较为丰富的信息，这段时间蔬菜销量波动较大，可能是由于蔬菜上市导致。在各品类中，茄类与其余品类变化趋势明显不同，在 2 月、3 月及 9 月呈增长趋势，其余为下降趋势。

（5）各单品总销量分析

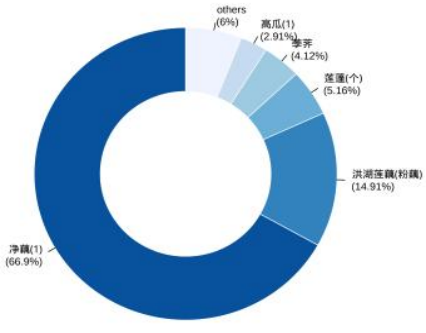


图 15 水生根茎类单品分布图

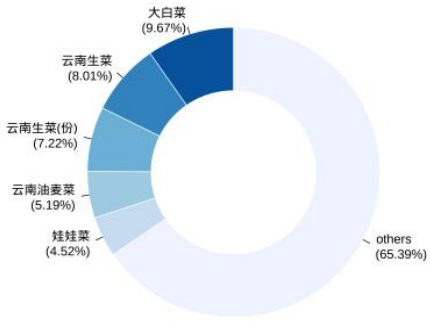


图 16 叶类单品分布图

提取各品类不同单品销售量进行汇总，制作占比环图。上图展示其中部分占比环图分析同一品类下不同单品销售分布及顾客对于各品类下单品的购买偏好。根据各品类蔬菜单品的总销量数据分析可得：

水生根茎类中共计 19 种单品，产地一净藕的销售情况最为显著，其总销量显著高于品类下其他单品，可以推测该产地是净藕的主要产地，且商品品质较好，因此受到绝

大多数顾客青睐。该品类下，净藕（1）与洪湖莲藕分别占总销量的 66.9%跟 14.91%，其销量远高于其他单品。综上所述，可以认为这两种单品是商超水生根茎类补货的主要商品。

花叶类中单品种类最为多样，共计 100 个蔬菜品种。且各类单品的销售分布较为分散，其中单品销量前三的大白菜、云南生菜、云南生菜（份）分别占品类总销量的 9.67%、8.01%。及 7.22%。因此，生鲜商超在采购花叶类商品时需要考虑更加全面，为了提供顾客更多选择，销售单品种类应尽量多样化。其余四种品类分析详见附录。

5.1.3 K-means 聚类分析

（1）数据预处理

K-means 聚类是一种无监督机器学习的分类方法，此算法适用于连续变量，因此本部分首先需要对附件 2 数据进行处理，例如对于扫码销售时间，这种精确到某天的毫秒的数据，直接计算它离 0 点的时间距离（单位为秒或者毫秒）；对销售日期，计算它离 2020-7-1（整个数据的第一天）的距离，这样日期就处理成了连续变量，或者顺序变量。对于分类型数据，采取独热编码，将其转换为连续变量。

此外聚类前需要对数据进行标准化处理，这里选用最小-最大缩放（Min-Max Scaling）法，对原始数据的线性变换，使结果值映射到[0 - 1]之间。计算公式为：

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

其中 x_{max} 为样本数据的最大值，x_{min} 为样本数据的最小值，x_{new} 为归一化后的样本值。

（2）聚类

以销量(千克)、销售单价(元/千克)、损耗率(%)、连续型时间、类损耗值、批发价格(元/千克)、天数差七个连续型指标加上独热编码处理后的分类名称指标、销售形式指标、是否打折指标、季节指标四项指标，进行聚类分析，具体分类结果见文件聚类结果.csv。

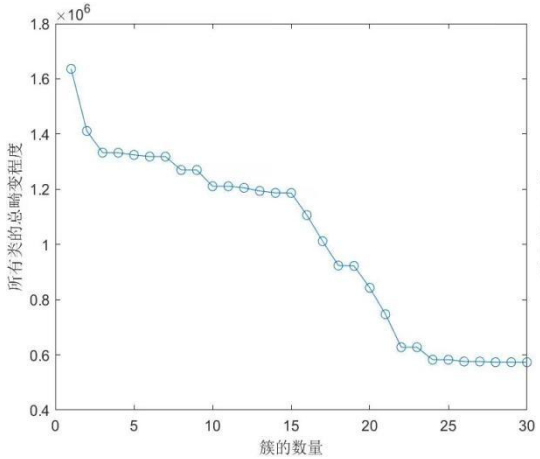


图 17 肘部图

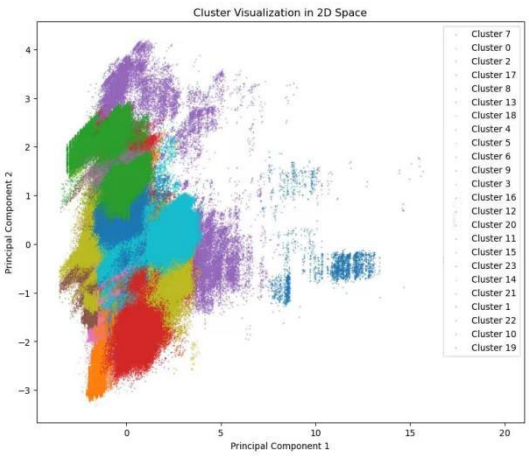


图 18 PCA 降维后各簇散点图

观察肘部图，说明簇的数量为 24 时聚类效果较好。

（3）聚类效果检验

使用方差分析和 kruskal-Wallis 检验来检验这个聚类是否合理，结果所有 p 值（见附录）都远小于 0.01，说明聚类结果十分合理。

5.1.3 基于 Pearson 相关系数的品类相关性分析

常用的相关系数有 Pearson 相关系数以及 Spearman 相关系数。当数据满足连续数据、正态分布以及线性关系三个条件时，两种系数均可使用，但是 Pearson 相关系数的效率更高。若其中一个条件没有满足，则可以使用 Spearman 相关系数。在对品类销售量散点图观察及正态分布的检验后，发现满足上述三个条件，因此选用效率更高的 Pearson

相关系数对品类之间的相关性进行分析。

（1）绘制散点图观察数据

Pearson 相关系数是衡量两个变量线性程度的指标，因此在计算前需要对其进行初步观察，因此需要绘制各品类散点图如下：

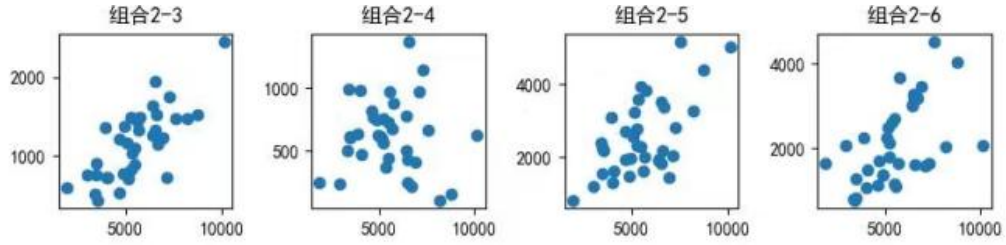


图 19 部分品类组合月销售量散点图

对水生根茎类、花叶类、花菜类、茄类、辣椒类、食用菌 6 类蔬菜品类进行编号，分别对应 1-6，两两绘制散点图，观察上图可以知道：部分品类组合月总销量具有正相关或负相关关系。

（2）正态分布检验

QQ 图是一种常用的检验是否服从正态分布的方法，是通过比较观察值的分位数与理论分布的分位数来进行比较的概率图方法。如果检验的随机变量服从正态分布，则其 QQ 图为一条直线。

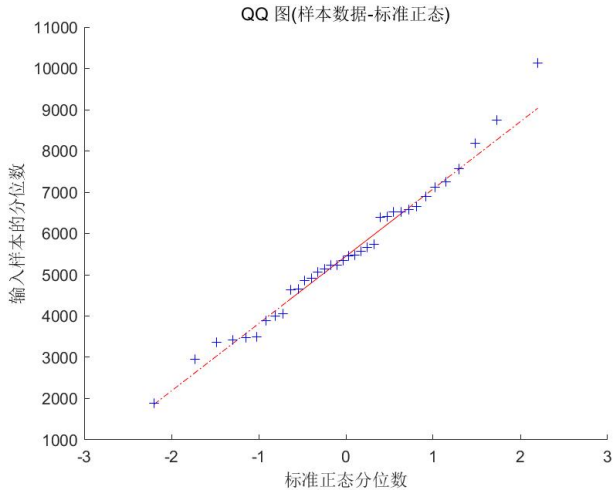


图 20 QQ 图检验结果

随机抽取花叶类的月度合成数据进行 QQ 图绘制，可以观察到结果近似为一条直线，得到符合正态分布的结论。

（3）Pearson 相关系数计算

基于上述散点图及 QQ 图的分析，可以得出：对于各品类月度销量数据之间的相关性，可以采用 Pearson 相关系数进行计算分析。

对 Pearson 相关系数大小的解释如下：定义相关系数在 0.7~1.0 或-0.7~-1.0 之间，才认为其具有强相关性。假设现在有两组品类月度销售量数据 X：$\{X_1、X_2.....X_n\}$和 Y：$\{Y_1、Y_2.....Y_n\}$，样本均值为：

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

$$\overline{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

样本协方差为：

$$\mathrm{Cov(X,Y)} = \frac{\sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{X} - \overline{X})(\mathrm{Y} - \overline{Y})}{\mathrm{n} - 1}$$

Pearson 相关系数：

$$r_{XY} = \frac{\mathrm{Cov(X,Y)}}{S_X S_Y}$$

其中，S_X, S_Y是样本标准差：

$$S_X = \sqrt{\frac{\sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{X} - \overline{X})^2}{\mathrm{n} - 1}}$$

$$S_Y = \sqrt{\frac{\sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{Y} - \overline{Y})^2}{\mathrm{n} - 1}}$$

基于上述公式，对相关系数矩阵计算如下图：

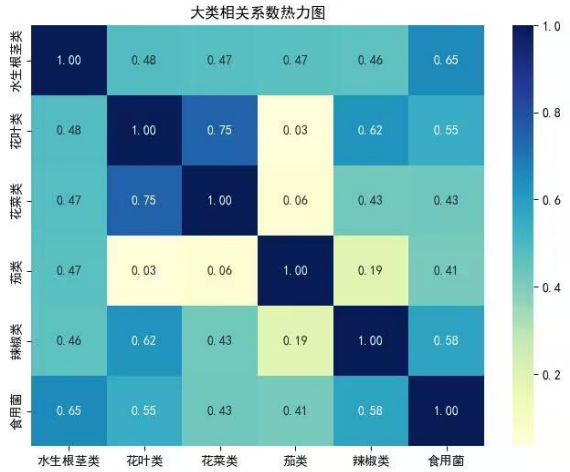


图 21 品类月度销量的 Pearson 相关系数矩阵

上图中，颜色越深表示品类间月度销量的相关性越强，基于本题定义的相关系数解释：相关系数在 0.7~1.0 或-0.7~-1.0 之间，才认为其具有强相关性，最终得到花菜类与花叶类具有强相关性。

（4）P 值判断法对 Pearson 相关系数 r 进行显著性检验

最后，本部分使用 P 值判断法对 Pearson 相关系数 r 是否显著异于 0 进行检验，提出了假设：

$$H_0: r = 0 \quad , \quad H_0: r \neq 0$$

花菜类与花叶类之间相关性 p 值为 0.000000167，p<<0.01，说明在 99%的置信水平上拒绝原假设，即二者间的相关性非常显著。

5.1.4 基于 Spearman 相关系数的单品相关性分析

（1）计算 Spearman 相关系数

由于单品种类多，单品之间不能证明其具有线性关系，因此对于单品销售量之间的相关性分析，本部分采用 Spearman 相关系数[1]进行分析：对 Spearman 相关系数大小的解释如下：定义相关系数在 0.8~1.0 或-0.8~-1.0 之间，才认为其具有强相关性。假设现在有两组单品月度销售量数据 X：$\{X_1、X_2.....X_n\}$和 Y：$\{Y_1、Y_2.....Y_n\}$，则 Spearman 相关系数计算公式为：

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

其中，d_i为X_i 和Y_i 之间的等级差，即将它的一列数据从小到大排序后，这个数所在的位

置。基于上述公式，计算两两单品之间 Spearman 相关系数如下图：

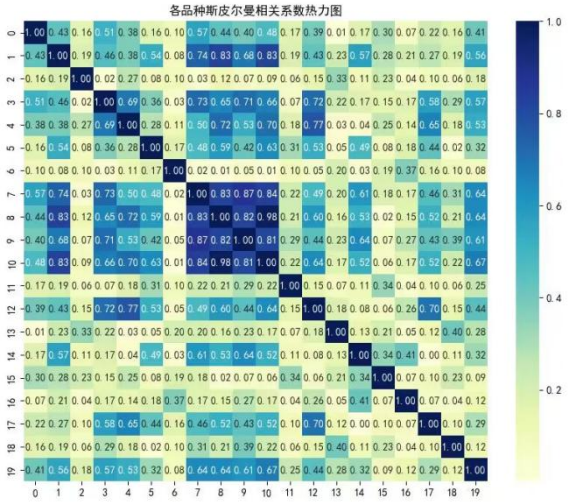


图 22 品类月度销量的 Pearson 相关系数矩阵

可以看出部分单品之间具有很强的相关性，筛选系数大于 0.8，且卖出时间超过 24 个月的单品组共 18 组，部分月度销量折线表如下：

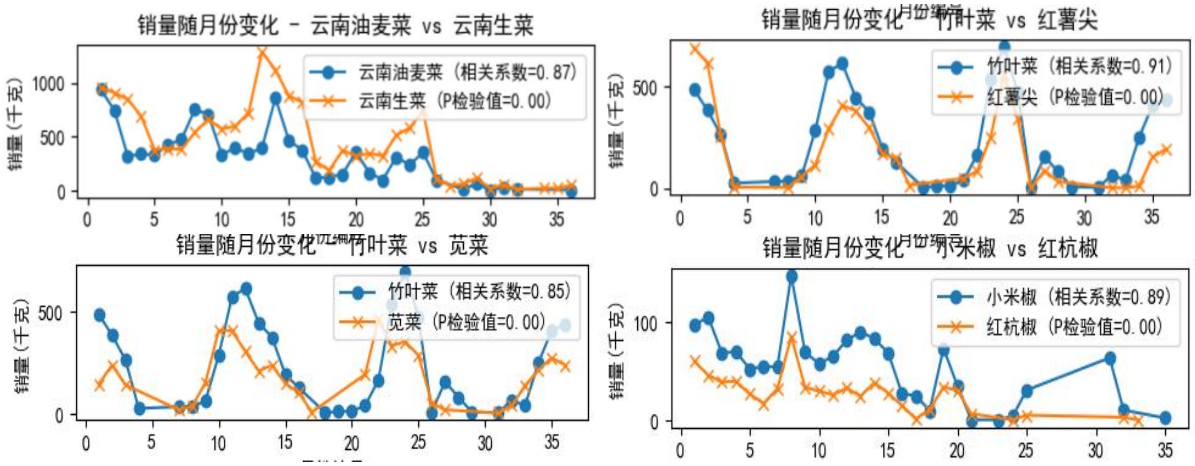


图 23 部分强相关性单品组合月度销量折线图

从上图可以很明显看出部分单品组合之间具有很明显的相关性，其中，竹叶菜和红薯尖的相关系数达到了 0.91，相关性最为明显。

（2）P 值判断法对 Spearman 相关系数进行显著性检验

最后，本部分使用 P 值判断法对 Pearson 相关系数r_s是否显著异于 0 进行检验，提出了假设：

$$H_0: r_s = 0 \quad , \quad H_0: r_s \neq 0$$

检验统计量为$r_s\sqrt{n-1}$，计算后求出对应的 p 值均<0.1，说明在 99%的置信水平上拒绝原假设，即上述对应 18 个单品组的相关性均非常显著。

5.2 问题二的模型建立与分析

问题二首先要求分析各蔬菜品类的销售总量与成本加成定价的关系，首先对于品类的价格、成本和销量，采用按月度求和平均的方式，根据成本加成定价法：销售价格=单位成本（1+成本利润率），求解出各品类利润率与品类销售量关系。其次要求给出各品类未来一周的日补货总量和定价策略，此部分采用回归+预测+优化的模式，建立一元线性回归模型和基于 Prophet 预测模型和高斯过程回归的融合模型以及优化模型进行求解。涉及品类的相关数据，均采用所包含单品的均值，具体求解步骤如下图：

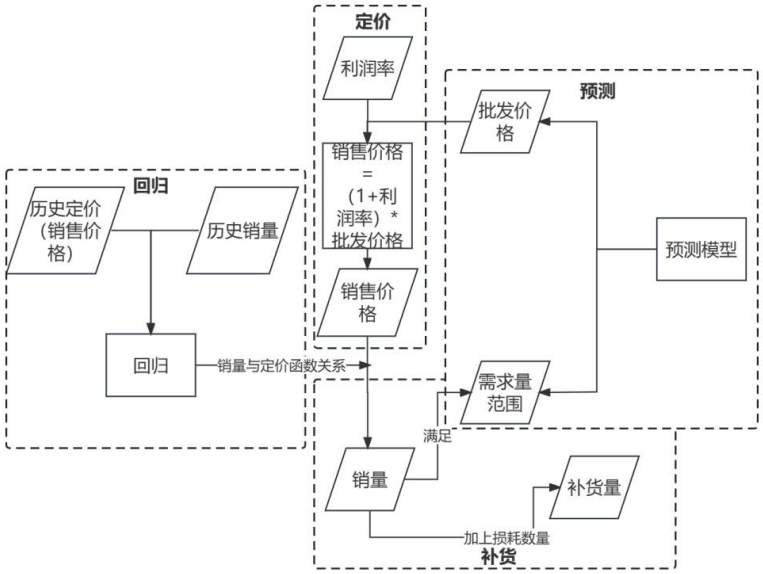


图 24 定价策略与补货策略制定流程图

最终以收益最大为目标建立目标函数，以利润率为变量找到使商超收益最大的定价策略。为满足供应需求，销售量需要在预测得到的需求量范围内。

5.2.1 数据准备

本部分首先对附件 2 各品类的价格数据分别进行了数据清洗，发现数据存在缺失值和异常值。由于缺失值很少，少于样本的 1%，并且当日批发进价和今日批发价格相似，所以直接采用最近的非空值填充。对于异常值的处理，采用 3σ 原则进行剔除：

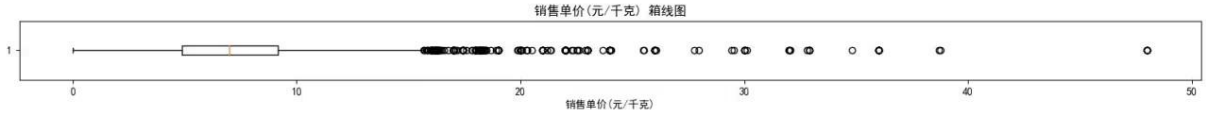


图 25 附件 2 销量箱线图

上图为水生根茎类的价格箱线图，可以观察到示例数据整体呈现右偏分布，主要集中在 6-7 之间，存在异常值。按照 3σ 原则，找到异常值上下界，对异常数据进行剔除。其余品类也按照此法进行数据处理。

5.2.2 探究分析销售量与成本加成定价的关系

成本加成定价法，是通过商家设定的利润率，将商品单位成本加上一定的利润制定产品价格的方法。由此，将各蔬菜品类销售总量与成本加成定价的关系转化为各品类商品销售量与利润率的关系。结合附件 3 的批发价格及附件 2 的销售价格，将各单品价格加权求均值得到品类价格，再通过成本加成定价法的公式：销售价格=单位成本（1 + 成本利润率），求得各品类商品利润率。以时间为 x 轴，做出利润率与总销量取对数处理后的变化折线图：

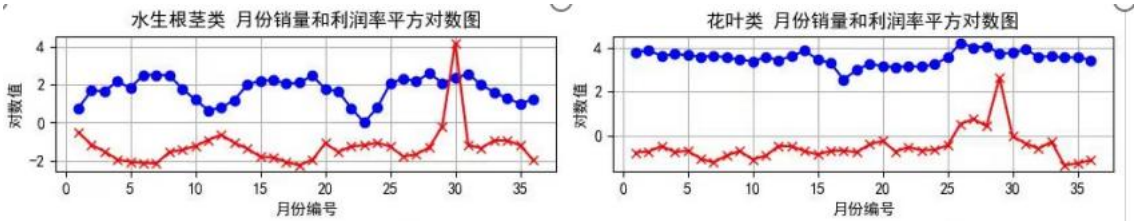


图 26 部分品类利润率-销量变化折线图

通过观察分析散点图，在各类商品与利润关系中，相关关系并不十分显著。其中，处理后的辣椒类与利润相关性 R 值为 0.6986，表现为中等相关关系。其余大部分在 2020 年 7 月到 2022 年末表现为部分相关，且除食用菌品类外，各品类在 2022 年末 2023 年

初利润率出现较明显增长，可推测是由于这期间需求增加导致商家对利润做出调整。

分析商家在定价时做出的定价决策，水生根茎类、食用菌大部分表现为负相关，可以推测出这两个品类商家采取的是薄利多销，采取降低利润来提高销量增加收益。其余品类大致呈现正相关关系，在销量（需求）增加时，提高利润来进一步增加收益。

由于以上得出的销量与利润的关系不明显，不能推出两者之间的函数关系。所以，根据销售价格=单位成本（1 + 成本利润率），转而去寻找销量与销售价格之间的关系。

5.2.3 建立一元线性回归模型

为观察品类销量与定价的变化趋势及相互关系，基于附件 2 数据单品销售数据，加权平均得到品类销量及价格，再对二者取对数，以时间为横轴，绘制折线图。

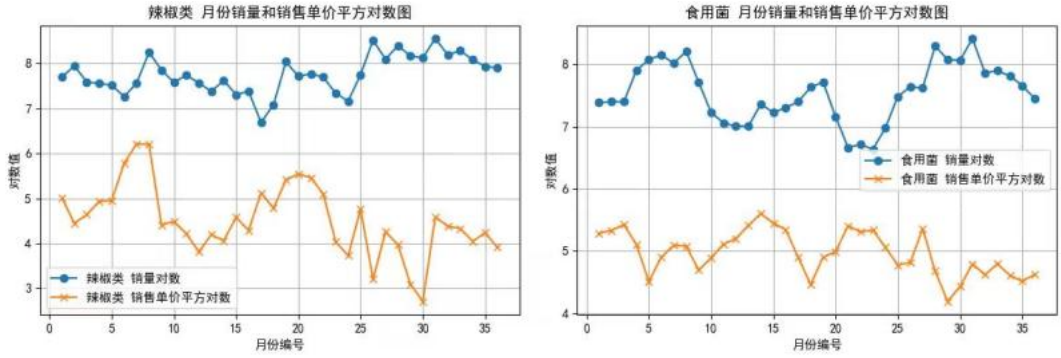


图 27 部分品类销量与定价变化趋势折线图

通过观察折线图给出的变化趋势，可以清晰地发现，销量与定价之间存在明显相反的变化趋势，大致呈负相关关系。以食用菌为例，进一步求得两者相关系数 R=-0.6076，表现为中等负相关；p 值为 0.0001，可在 0.01 的显著性水平下认为是负相关。

综上所述，分析两者之间的函数关系，研究不同品类的价格弹性。本部分选用计量经济学中广泛使用的双对数模型，该模型可将非线性关系转化为线性关系，且该模型应用于分析消费者对价格变化敏感程度、经济增长对投资影响及出口与汇率关系等等，常用于研究收入弹性、价格弹性等。对此，为了更精确地分析销量受定价变动的影响，将品类销量与定价各取对数，建立如下模型对两个函数关系进行回归分析：

$$\ln q = a + b \ln p$$

这里 q 表示品类销售量，p 表示品类价格。对两边求导可以得到：

$$\frac{\mathrm{dq}}{\mathrm{q}} = b \frac{\mathrm{dp}}{\mathrm{p}}$$

这里 $\frac{\mathrm{dp}}{\mathrm{p}}$ 即表示定价的相对变化，$\frac{\mathrm{dq}}{\mathrm{q}}$ 表示销量的相对变化，定价 p 的相对变化引起了销量 q 的相对变化，所以这里的 b 正表示 q 对 p 的弹性，具体而言，定价每增加 1%，将导致销量增加 b%。针对这里建立的模型可以分析得到，通过对因变量和自变量取对数可以降低数据波动对回归结果的影响，进一步提升模型的精确度，同时求导后得到的回归系数正代表了价格弹性，简化了计算[2]。

基于上述模型和附件 2 数据，以水生根茎类为例得到回归结果及方差分析如下：

表 2 双对数模型的回归结果

q	Coefficients	标准误差	t Stat	P>t	Lower 95%	Upper 95%
b	-1.380787	0.2861489	-4.83	0	-1.962311	-0.7992622
a	9.905795	0.6629778	14.94	0	8.558462	11.25313

表 3 双对数模型的方差分析

Source	SS	df	MS
回归	6.21559338	1	6.21559338

残差	9.07595747	34	0.266939926
Total	15.2915508	35	0.436901453

回归结果表明，得到销量与定价函数关系为，$\ln q = 9.906 - 1.381 \ln p$。其中，价格弹性系数为-1.381，表示价格与销量成反比。可以粗略地说，定价每增加 1%，将导致销量下降 1.381%。

5.2.4 建立基于 Prophet 和高斯过程回归的预测模型

本部分主要根据批发价格与销售量数据对未来一周的批发价格和需求量范围进行预测，常用的预测方法主要是时间序列预测以及机器学习。本部分在比较七种机器学习的模型后，选取了预测效果最好的高斯过程回归模型，和时间序列模型进行融合，取平均值作为最终的预测结果。

（1）Prophet 模型预测

Prophet 使用了一种基于加法分解的模型，可以高效地进行拟合和预测，尤其适用于大规模时间序列数据。Prophet 更加专注于时间序列分析，可以更好地捕捉和分析季节性趋势和假期效应等。本部分 Prophet 模型的建立与问题一相同，这里不再重复，列出一些重要控制参数如下：

表 4 Prophet 模型参数结果表

参数名称	参数值
changepoint_prior_scale	0.15
yearly_seasonality	TRUE
growth	linear
seasonality_prior_scale	multiplicative

changepoint_prior_scale 指设定自动突变点选择的灵活性；yearly_seasonality 代表时间序列的时序周期性，增加预测模型的拟合程度；growth 指增长趋势，分为“linear”与“logistic”，分别代表线性与非线性的增长，默认值为 linear；seasonality_prior_scale 代表改变周期性影响因素的强度，值越大，周期性因素在预测值中的影响程度越大。

1）未来一周（7.1-7.7）的需求量预测结果

以水生根茎类为例：

表 5 7.1-7.7 水生根茎类需求量预测表

	预测值	下界	上界
7.1	31.51403422	22.43397765	40.3940729
7.2	27.21930854	18.50555806	36.34357316
7.3	15.95047518	7.625425143	25.19527342
7.4	15.28787515	6.644053331	23.93793046
7.5	14.62375048	5.938928568	24.20963441
7.6	16.77276953	7.271027793	26.05307241
7.7	23.77090806	14.76173934	32.9966772

得到模型拟合及预测结果如下：

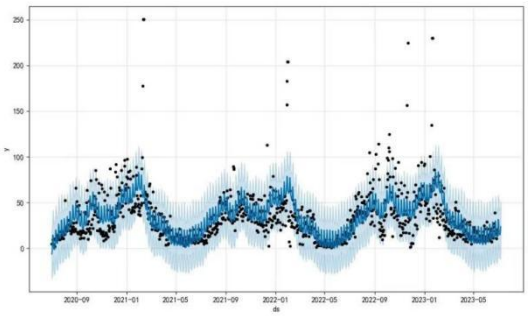


图 28 批发价的拟合情况及预测结果图

图中的黑色点表示原始数据中的每个数据点；蓝色线表示模型的拟合结果；浅蓝色区域表示模型的置信区间：置信区间越宽，表示模型的不确定性越大。折线尾部即预测结果与置信区间。可以观察到，模型的拟合效果良好，预测结果较好。

（2）高斯过程回归模型预测

为在机器学习的众多方法中选中合适的预测模型，本部分对决策树、线性回归、SVM、高斯过程回归、最小二乘法、集成学习——随机森林、深度学习——神经网络七种机器学习的方法进行测试。

1）评价指标：RMSE

RMSE（Root Mean Squared Error）是均方根误差的缩写。这是一种常用的回归模型性能评估指标，表示预测值与实际值之间的差异。RMSE 可以用以下公式计算：

$$\mathrm{RMSE}=\sqrt{\frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}(y_i-\hat{y}_i)^2}$$

其中，n 是样本数量，y_i(i=1,2……n)表示第 i 个样本的真实值，$\hat{y}_i$(i = 1,2.....n)表示第 i 个样本的预测值。RMSE 越小，说明模型预测结果与真实值之间的差距越小，表示模型的预测能力越好，所以 RMSE 的值一般越小越好。

计算上述其中模型的 RMSE，得到结果如下：

表 6 七种模型 RMSE 值

模型	决策树	线性回归	SVM	高斯过程回归	最小二乘法	集成学习——随机森林	深度学习——神经网络
RMSE	0.911	2.0417	2.0979	0.86283	2.0414	0.89094	1.4789

可以看出高斯过程回归的 RMSE 值最小，预测能力最好。为了评估提出的模型在不同数据集上的泛化性能，本部分采用了 5 折交叉验证的方法。在交叉验证过程中，整个数据集被随机分为 5 个不重叠的子集。在每一轮验证中，模型在 4 个子集上进行训练，然后在剩下的一个子集上进行测试。这一过程会重复 5 次，每次选择一个不同的子集作为测试集。最后，5 次测试的结果会被平均，以获得更稳健的模型性能估计。使用交叉验证，不仅能更准确地估计模型的泛化能力，还能有效地避免模型过拟合训练数据。

（3）基于 Prophet 和高斯过程回归的预测模型

此部分进行模型融合。本文对两个基础模型，Prophet 模型和高斯回归模型用平均法的融合方法进行模型融合，将两个模型的预测结果简单求取平均值作为最终的预测结果。由于高斯回归模型不能很好地预测需求量的范围，因此将 Prophet 模型的需求量结果视为最终预测结果，但是对批发价格取平均，得到最终结果如下：

表 7 7.1-7.7 批发价格预测表

	7 月 1 号	7 月 2 号	7 月 3 号	7 月 4 号	7 月 5 号	7 月 6 号	7 月 7 号
水生根茎类	10.80452644	10.75129109	10.64691782	10.50834972	10.39209108	10.16189466	10.12586851
花叶类	3.869657866	3.903072401	3.897680868	3.942031626	3.99631894	4.009600475	4.025418026
花菜类	8.319114375	8.299734407	8.314479943	8.305909985	8.329872893	8.369917782	8.396204832
茄类	4.024664956	4.057338233	4.026396251	3.999175207	4.042158884	4.05967175	4.056002689
辣椒类	6.451655946	6.463206946	6.45986809	6.468819013	6.562334879	6.576496178	6.643959664

食用菌	5.207228 6	5.244110 826	5.353391 197	5.382549 566	5.381475 588	5.415753 082	5.425308 461

5.2.5 建立收益优化模型

本部分根据预测的品类批发价格与需求量，结合求解得到的回归函数，建立以收益最大为目标的优化模型，得出最佳定价策略与补货策略。

（1）进行一些符号说明

x_{ij}为预测的第 i 种品类第 j 天的批发价格；t_{ij}为第 i 种品类第 j 天的利润率；p_{ij}为第 i 种品类第 j 天的销售价格；q_{ij}为第 i 种品类第 j 天的销售量；p'_{ij}为第 i 种品类第 j 天的打折价格；k_i为第 i 种品类的损耗率；m_i为第 i 种品类的打折率，即打折数占总数的比值；n_i为第 i 种品类的折扣率，即打折价格与原价的比值；r_{ij}为第 i 种品类第 j 天的补货量；a_{ij}为第 i 种品类第 j 天的需求量上界，b_{ij}为第 i 种品类第 j 天的需求量下界，i=1,2……6，j=1,2……7。

（2）确定决策变量

要获得最大收益的方案，需要确定最适合的利润率，由此解出价格和销量等数据，因此确定利润率为决策变量。

（3）确定目标函数

收益应同时考虑到正常售卖的销售额、打折售卖的销售额以及成本，即收益=正常售卖的销售额+打折售卖的销售额-成本，其中，由于损耗率的存在，成本考虑的不应是销售量而应是补货量，补货量与销售量的关系应为$r_{ij}(1-k_i)=q_{ij}$，i=1,2……6，j=1,2……7，设收益为 w，建立目标函数为：

$$\max w=\sum_{i=1}^{6}\sum_{j=1}^{7}[p_{ij}\,q_{ij}(1-m_i)]+\sum_{i=1}^{6}\sum_{j=1}^{7}m_i q_{ij}\,p'_{ij}-\sum_{i=1}^{6}\sum_{j=1}^{7}\frac{q_{ij}}{1-k_i}x_{ij}$$

（4）确定约束条件

销售价格采取“成本加成定价”的方式，即销售价格=（1+利润率）×批发价格，则建立约束条件为：

$$p_{ij}=(1+t_{ij})x_{ij}$$

销售量是通过已建立的回归函数得到的，建立约束条件为：

$$\ln q_{ij}=a+b\ln[(1+t_{ij})x_{ij}]$$

销售量需满足需求量的范围，即利润率需要在一定的范围内，建立约束条件为：

$$\ln a_{ij}\le a+b\ln[(1+t_{ij})x_{ij}]\le \ln b_{ij}$$

打折价格是在销售价格乘以一定折扣率得到的，建立约束条件为：

$$p'_{ij}=n_i(1+t_{ij})x_{ij}$$

（5）建立收益优化模型

$$\max w=\sum_{i=1}^{6}\sum_{j=1}^{7}[p_{ij}\,q_{ij}(1-m_i)]+\sum_{i=1}^{6}\sum_{j=1}^{7}m_i q_{ij}\,p'_{ij}-\sum_{i=1}^{6}\sum_{j=1}^{7}\frac{q_{ij}}{1-k_i}x_{ij}$$

$$\begin{cases} p_{ij}=(1+t_{ij})x_{ij} \\ \ln q_{ij}=a+b\ln[(1+t_{ij})x_{ij}] \\ \ln a_{ij}\le a+b\ln[(1+t_{ij})x_{ij}]\le \ln b_{ij} \\ p'_{ij}=n_i(1+t_{ij})x_{ij} \end{cases}$$

5.2.6 优化模型求解

上述所建模型中，批发价x_{ij}和需求量的上下界a_{ij}、b_{ij}已经由预测得出，a, b 值也在

回归模型中解出，打折率m_i和折扣率n_i均基于历史数据对各单品相应指标求和取平均得到，损耗率k_i也通过对附件 4 中各品类中各单品的损耗率简单求解平均值得到。则上述模型仅需要考虑利润率t_{ij}的变化对收益 w 的影响，使用 matlab 中的粒子群优化问题求解器得出最优的利润率。

由模型可以知道销售价格求解公式为$p_{ij}=(1+t_{ij})x_{ij}$，补货量与销售量的关系为$\mathrm{r_{ij}}\,(1-k_i)=q_{ij}$，即补货量求解公式为：$\mathrm{r_{ij}}=\frac{\mathrm{e}^{a+b\ln[(1+t_{ij})x_{ij}]}}{(1-k_i)}$，基于此，求解结果如下：

表 8 7.1-7.7 各品类销售价格表

	7 月 1 日	7 月 2 日	7 月 3 日	7 月 4 日	7 月 5 日	7 月 6 日	7 月 7 日
水生根茎类	11.68	13.43	25.52	28.20	30.59	26.42	15.82
花叶类	4.76	5.46	10. 58	11.72	11.27	11.45	9.19
花菜类	11.24	11.10	20.47	21.95	20.10	19.21	14.44
茄类	6.09	6.18	10.32	11.29	12.26	11.16	9.48
辣椒类	8.00	8.38	30.78	36.04	37.74	32.45	14.92
食用菌	10.58	11.11	14.24	15.24	14.10	13.86	11.91

表 9 7.1-7.7 各品类日补货总量表

	7 月 1 日	7 月 2 日	7 月 3 日	7 月 4 日	7 月 5 日	7 月 6 日	7 月 7 日
水生根茎类	40.39	36.34	25.19	23.93	24.2	26.05	32.99
花叶类	239.77	227.64	179.16	174.57	177.84	173.37	187.72
花菜类	44.57	45.35	33.47	32.6	33.24	34	39.94
茄类	31.22	30.59	24.55	22.82	22.94	23.58	24.75
辣椒类	127.28	119.53	94.8	91.75	89.82	93.77	105.19
食用菌	110.1	105.71	79.81	77.87	84.14	84.5	98.09

5.3 问题三的模型建立与分析

问题三是对单品 7 月 1 日补货量和定价策略的建模分析，本题首先根据题目所给要求，筛选出满足时间条件的可售单品。然后以净利润和损耗率为指标建立 TOPSIS（Technique for Order Preference by Similarity to Ideal Solution）评价模型对可售单品进行打分排序，初步筛选出前 40 个单品。对筛选出的可售商品沿用第二问的解题思路，同样采用回归+预测+优化的模式，因回归与预测的方法与上题思路一样，因此此题中仅简单说明计算结果不再对求解过程进行赘述。基于回归与预测的结果，在问题二的模型基础上，添加 0-1 整数决策变量，以收益最大为目标建立 0-1 整数规划模型求解结果。

5.3.1 建立 TOPSIS 评价模型

首先按照题目要求，筛选出 2023 年 6 月 24 日-30 日的可售品种，共 49 种。

1）评价指标

TOPSIS 的指标分为两种主要类型，即成本型和效益型，TOPSIS 法的主要目标是找到一个最佳的解决方案，使该方案在成本型指标上具有最小的值，同时在效益型指标上具有最大的值，因此本部分选取损耗率作为成本型指标，选取净利润作为效益型指标，建立指标体系，净利润定义为单品所有销量×（销售价格-当日批发进价）。

2）指标权重

根据经验给定损耗率指标的权重为 0.1，净利润指标的权重为 0.9。

3）模型建立与求解

对于模型求解步骤如下

Step1	对数据进行归一化和正向化处理
Step2	用定义的权重乘以归一化和正向化后的数据
Step3	采用余弦法找出有限方案中的最优方案和最劣方案
Step4	计算各评价对象与最优方案和最劣方案间的距离
Step5	计算各评价对象与最优方案的贴近程度评分。

最终求解出排名前 40 的单品种类，部分结果如下表：

表 10 排名前五的单品名称与得分情况

单品名称	净利润	损耗率(%)	TOPSIS 得分
西兰花	89951.68923	9.26	0.977995731
芜湖青椒(1)	61356.95698	5.7	0.682530047
净藕(1)	57448.83735	5.54	0.639285418
云南生菜	49860.65935	15.25	0.553727197
紫茄子(2)	39867.58729	6.07	0.444869459

5.3.2 预测与回归

本部分使用 Prophet 模型对每个单品 7.1 日的批发价格进行预测，并对每个单品的销售价格和销售量构建回归函数得到价格与销售量的关系。

5.3.5 建立双层优化模型

本部分采用双层优化模型，第一层针对独立的每个商品，根据预测的单品批发价格与补货量约束，结合求解得到的回归函数，建立以收益最大为目标的优化模型，得出独立的每个商品的最佳销售价格与补货量。第二层针对 40 种单品建立 0-1 整数规划模型。

（1）第一层收益优化模型

1）进行一些符号说明

v_i为第 i 种单品的收益；x_i为预测的第 i 种单品的批发价格；t_i为第 i 种单品的利润率；p_i为第 i 种单品的销售价格；q_i为第 i 种单品的销售量；p'_i为第 i 种单品的打折价格；k_i为第 i 种单品的损耗率；m_i为第 i 种单品的打折率；n_i为第 i 种单品的折扣率；r_i为第 i 种单品的补货量；i=1,2……33。

2）确定决策变量

要获得当日最大收益的方案，需要确定最适合的利润率，由此解出价格和销量等数据，因此确定利润率t_i为决策变量。

3）确定目标函数

收益应同时考虑到正常售卖的销售额、打折售卖的销售额以及成本，即收益=正常售卖的销售额+打折售卖的销售额-成本，其中，由于损耗率的存在，成本考虑的不应是销售量而应是补货量，补货量与销售量的关系为$r_i(1-k_i)=q_i$，设某单品当日收益为v_i，i=1,2,……40，则对于独立的每一个单品，可建立目标函数为：

$$\max v_i = p_i q_i (1-m_i)\alpha_i + m_i q_i p'_i \alpha_i - \frac{q_i}{1-k_i} x_i \alpha_i$$

4）确定约束条件

A.销售价格采取“成本加成定价”的方式，即销售价格=（1+利润率）×批发价格，则建立约束条件为：

$$p_i = (1+t_i)x_i$$

B.销售量是通过已建立的回归函数得到的，建立约束条件为：

$$\ln q_i = a + b\ln[(1+t_i)x_i]$$

C.各单品订购量要满足最小陈列量 2.5 千克，因此补货量需满足此要求，建立约束条件为：

$$r_i \geq 2.5$$

D.补货量与销售量之间存在损耗率，由补货量的范围即可得到销售量的约束条件，即利润率需要在一定的范围内，建立约束条件为：

$$\mathrm{r_i}(1-k_i)=q_i$$

$$\ln[2.5(1-k_i)] \leq a+b\ln[(1+t_i)x_i]$$

E.打折价格是在销售价格乘以一定折扣率得到的，建立约束条件为：

$$p'_i = n_i(1+t_i)x_i$$

5）建立第一层收益优化模型

$$\max \mathrm{v_i} = p_i q_i(1-m_i)\alpha_i + m_i q_i p'_i \alpha_i - \frac{q_i}{1-k_i}x_i\alpha_i$$

$$\begin{cases} p_i=(1+t_i)x_i \\ \ln q_i = a+b\ln[(1+t_i)x_i] \\ r_i \geq 2.5 \\ \mathrm{r_i}(1-k_i)=q_i \\ \ln[2.5(1-k_i)] \leq a+b\ln[(1+t_i)x_i] \\ p'_i = n_i(1+t_i)x_i \end{cases}$$

（2）第二层 0-1 整数规划模型

1） 符号说明

v 为进货单品总收益；α_i为 0-1 决策变量，取 1 表示选择此单品，否则不选，i=1,2,…… 40。

2）确定决策变量

要获得当日最大收益的方案，需要确定选择的单品种类，由此给出最优进货方案，因此确定和α_i为决策变量。

3）确定目标函数

第一层优化模型已经求出每一种单品的最优进销售价格与补货量，此层模型需要在此基础上得出使总收益最大的进货方案，因此建立目标函数为：

$$\max \mathrm{v} = \sum_{\mathrm{i}=1}^{40} \mathrm{v_i}$$

4）确定约束条件

售卖单品数量要控制在 27~33 个，建立约束条件为：

$$27 \leq \sum_{i=1}^{33} \alpha_i \leq 33$$

$$\alpha_i = \begin{cases} 1，选择此单品 \\ 0，不选择此单品 \end{cases}$$

5）建立第二层 0-1 整数规划模型

$$\max \mathrm{v} = \sum_{\mathrm{i}=1}^{40} \mathrm{v_i}$$

$$\begin{cases} 27 \leq \sum_{i=1}^{33} \alpha_i \leq 33 \\ \alpha_i = \begin{cases} 1，选择此单品 \\ 0，不选择此单品 \end{cases} \end{cases}$$

5.3.6 采用带惩罚的随机扰动 PS0（粒子群优化算法）求解

模型中 a,b 值在回归模型中解出，打折率m_i和折扣率n_i均基于历史数据统计得到。由于此问比较复杂，本部分采取带惩罚的随机扰动 PSO 对模型进行求解。

该问题中，以利润率和进货量为变量进行约束，均为连续性变量且上下界空间大，不论采用 Lingo 求解还是线性规划等传统算法均不能在可接受的时间内完成计算，同时商场进货可能面临各种约束，例如存储空间、资金、最小或最大进货量等。PSO 可以通过加入惩罚项来很方便地处理这些约束。故针对本部分采用 PSO 对模型进行求解，并在算法中引入了特定的惩罚项和随机扰动以提高其效果。

1）惩罚项

在本问题中，有一个重要约束：进货量（去除折损率影响后）必须不小于销量。为了在 PSO 算法中体现这一约束，在目标函数（净利润）中引入了一个惩罚项。当一个粒子的解不满足这一约束时，其目标函数值被设置为负无穷大，从而在后续迭代中被自然淘汰。

2）随机扰动

为了增加算法的全局搜索能力并避免陷入局部最优，我们在更新粒子速度和位置的过程中添加了随机扰动。这一操作虽小，但对提高算法性能有明显贡献，增加模型的稳定性。

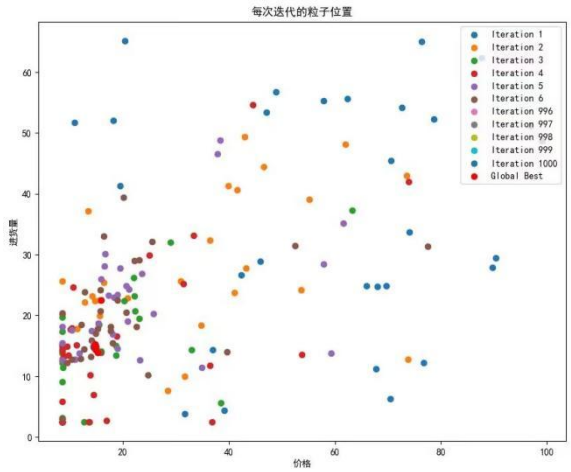


图 29 西兰花迭代前 5 代和后 5 代和最佳粒子位置

最终求解出 33 种进货单品、单品利润、销售价格及补货量如下：

表 11 种单品的最大利润、最佳价格及补货量

单品名称	最大利润	最佳定价	最佳采购量	单品名称	最大利润	最佳定价	最佳采购量
西兰花	79.94	14.74	14.88	红薯尖	40.95	10.92	5.83
芜湖青椒(1)	233.61	18	19.72	奶白菜	43.39	11.59	5.4
净藕(1)	101.35	19.79	9.68	长线茄	83.88	21.6	5.89
云南生菜	58.09	11.63	9.57	菠菜(份)	44.45	6.8	17.4
紫茄子(2)	165.16	18	12.39	双孢菇(盒)	43.25	6.9	15.87
螺丝椒	67.51	21.48	5.69	小皱皮(份)	27.77	6.81	7.58
云南油麦菜	67.35	16	6.59	苋菜	40.78	11.34	5.06
小米椒(份)	80.29	8.55	14.99	白玉菇(袋)	26.75	9.9	5.4
云南生菜(份)	143.73	6.9	47.78	小青菜(1)	38.5	12	17.41
上海青	77.71	25.8	3.79	洪湖藕带	40.17	32.57	3.17
金针菇(盒)	499.68	8.9	73.56	木耳菜	102.96	19.8	6.75
西峡花菇(1)	64.05	27.6	5.18	高瓜(1)	47.19	31.6	2.51

枝江青梗散花	55.45	14	14.76	青线椒(份)	28.37	8.9	6.71
菠菜	156.71	27.6	9.71	海鲜菇(包)	144.48	9.9	2.5
螺丝椒(份)	88.62	8.98	15.72	蟹味菇与白玉菇双拼(盒)	30.87	13.8	6.52
云南油麦菜(份)	127.87	11.8	16.76	高瓜(2)	52.18	29.6	3.88
竹叶菜	68.21	11.56	8.61				

最大总净利润为 2971.27。

5.4 问题四的模型建立与分析

在前面的问题分析与建模求解中，我们发现在题目给出数据中缺乏进货量、同类蔬菜商品市场情况、产品质量保质期等数据。综合文献调研获得的资料，我们还总结得出，要进一步完善对于蔬菜商品的定价计划及补货决策的分析，可以从供给侧分析历年产量情况、种植技术、运输情况等；对需求侧分析，顾客的购买行为的时间偏好、区域偏好、因天气、商超地域位置、交通情况等导致的人流量情况等以及考虑供求关系，并运用经济学、运筹学相关理论方法进行分析。本文还在 Kaggle 网站中搜寻相关数据，作为附件给出信息的补充，并对其进行探索性分析，帮助问题分析更加全面。

（1）货架容量

问题三对于销售容量的限制仅考虑的单品数量，缺乏对于单品货架陈列空间的限制。参考运筹学 0/1 背包问题，在问题三 0-1 规划单品组合的收益最大的基础上，增加单品组合的容量满足货架空间的限制，改为动态规划问题。这使得问题更接近于实际情况，促使商家对于补货决策更加准确，实用性更强。

这里假设货架容量为某固定值ρ要大于单品陈列值ρ_i总和，即满足$\sum_{i=1}^{33}\rho_i\,\alpha_i \le \rho$，随机生成各种单品的需要的容量空间数据，并存放于表格中。最后在问题三的基础上求出部分结果如下表：

表 12 考虑货架容量基础上的部分结果

单品名称	最大利润	最佳定价	最佳采购量
西兰花	79.94	14.74	14.88
芜湖青椒(1)	233.61	18	19.72
净藕(1)	101.35	19.79	9.68
云南生菜	58.09	11.63	9.57
紫茄子(2)	165.16	18	12.39

最大利润结果为 2845.24 元。

（2）进货量

进货量，即补货量，在前文的分析中，由于附件给出的数据只包括销售量及打折量，而给出的损耗率是运输过程的损耗，商超对运损商品及因销售时间增加而品质变差的商品进行打折：m*q≠k*r，打折销售数量不等于运损数量；且商品并不一定全部销售完，可能存在销毁的部分，即 r ≠（1-m）*q+m*q，补货量 r 不等于正常销售及打折销售数量，因此，每日的具体补货量即损耗数量并不明确。在引入历史进货数据后，可以统计得出每日具体收益、商品损耗情况及销售量与需求量比率，同时在问题 2、3 的求解补货决策时，弥补由于给出的打折量数据极少的缺点，由此推出历史补货决策来对问题分析提供借鉴参考。

（3）市场上同类商品价格，或者同类（竞争）商家的定价决策

市场上同类商品价格，或者同类（竞争）商家的定价决策将会影响顾客对于本商超商品的购买决策。在题目给出的数据中，均是有关该商超自身的销售补货数据，并没有涉及其他商家的销售情况。而在实际情况中，市场类型，包括完全竞争市场及垄断竞争

市场等，单品平均价格，其他商家促销活动等等都会影响商户对于补货和定价的决策。综合整体市场情况，可以有助于商家对商品做出定价决策更加准确，同时根据其他商家定价销售情况进行相应打折或者促销活动的调整。

一般蔬菜等农产品卖家和买家很多，产品几乎没有差异，属于完全竞争市场，价格由供给及需求决定，商家不能擅自调整商品价格。但也不排除少数蔬菜品种由于产量稀少等原因存在部分垄断的情况，这里主要分析完全竞争市场及垄断竞争市场的短期均衡情况。

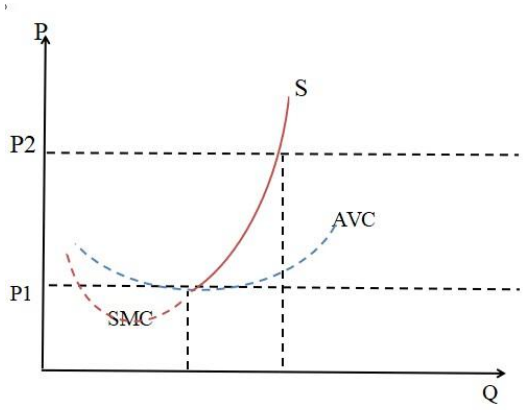


图 30 厂商短期均衡供给曲线

厂商短期均衡供给曲线呈现为从左下到右上的曲线，以斜率为边际成本，由完全竞争厂商的短期供给曲线可以分析在厂商在每一个价格水平的供给量都是能够给他带来最大利润或最小亏损的最优产量。

在垄断市场下不像完全竞争市场具备产量与价格一一对应关系，因此对于垄断厂商来说供给曲线并不存在。垄断竞争市场需求曲线向右下倾斜，卖家定价一定程度上可以控制产品价格，且受市场竞争限制较完全竞争市场小。

（4）产品质量包括蔬菜商品性质、保质期

产品质量包括蔬菜商品性质、保质期，直接影响顾客的购买决策，从而导致销售量波动。题目中只给出了运损及因时间导致的商品质量变差，且本文在分析中假设当日采购商品只能在一天内出售，不考虑存储成本。而在实际情况中，商品本身可能存在大小、形状、色泽等差异，可存储销售时间也跟具体商品性质有关。加入对于这个影响因素的考量，首先顾客存在异质性对于不同大小、色泽产品有不同的偏好与需求；其次，引入产品保质期可以延长销售时间，延长补货周期、减少补货量，同时顾客对于保质期更长的生鲜产品存在更高的偏好。并且研究显示，缩短补货周期能使蔬菜商品需求增加。在对于分析商超收益时需要增加对于存储成本、保鲜成本的考虑。

由于需求是随机的，这里的存储模型表现为随机型存储模型。结合题目情况考虑，商超可以结合商品保质期及近期批发价格，采取定期订货与定点订货结合的方法，每日检查一次存储量，若低于当日需求量必须补货，其余情况根据最低成本做出补货决策。

（5）从供给侧分析

收集历年蔬菜单品的产量数据、各品种蔬菜上市时间，可以分析产量对于市场平均价格及市场需求的影响及价格随上市时间的波动情况，对于一段时间内商品价格进行提前预估。例如，某一年某蔬菜商品销量不错，第二年农户可能因此增加对于该品种蔬菜的种植规模，导致输入市场的供给数量大增，导致产品价格下降；种植技术改进影响蔬菜产量，从而影响蔬菜价格及需求；蔬菜运输情况在本文中考虑到了运损，同时运输距离、运输方式等会影响蔬菜运输成本，进而影响蔬菜价格。

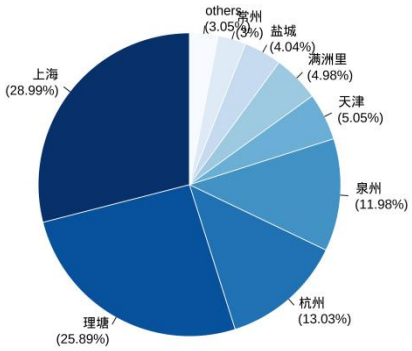


图 31 搜集数据集与附件 2 整合的产地占比

（6）从需求侧分析

时间偏好我们也在问题分析中得出在周末及春节期间销售量明显增长；不同区域居民对于某种商品可能存在明显的偏爱或者厌恶，通过前面调研收集以上数据可以帮助商超更精确地进行采购决策。同时商超地理位置，当日天气状况可能影响商超长期及当日人流量，进而影响需求量。部分商品需求量呈现价格刚性，如香料类的农产品，无论价格怎么变化，需求量均保持不变，对此商户对此类商品价格调节的自由度稍大。蔬菜商品大多具有重复购买特征，可以结合会员（忠诚顾客）销售信息，尝试从产品需求扩散模型分析需求及价格决策[3]。

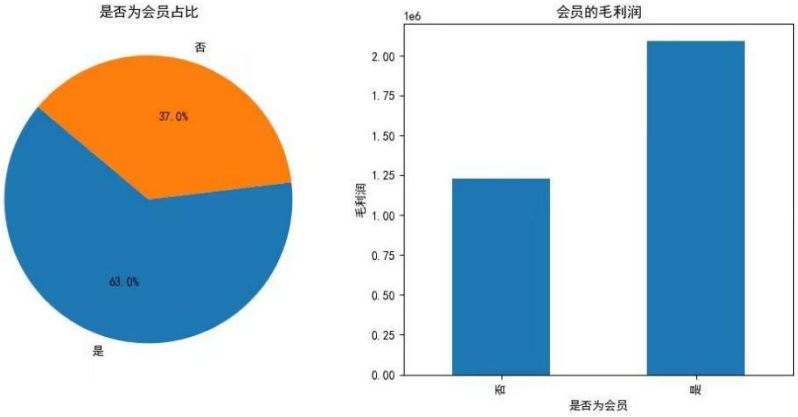


图 32 会员销售信息

（7）综合供求关系，探究市场是买方市场还是卖方市场

题目仅考虑商超补货需要满足顾客需求，而在整个市场上可能存在供大于求或者供不应求的情况。在这种情况下，商超采购及销售过程中，存在买家卖家身份的变化。与卖方市场情况刚好相反，在供大于求的买方市场，商品价格有下降趋势，买方在交易中处于有利地位。对此，商超在批发价格的讨价还价及定价的过程中可以参考不同市场理论进行具体分析调整。

借鉴报童问题[4]，忽略打折情况进行分析，商家每卖出一份商品赚（p-x）元，若未售出每份亏 x 元，每日售出 i 份商品的概率为 P(i)，在这里若引入进货量，可以由历史数据推导 P(i)的公式，这里假设 P(i)已知。这里分析订货量为何值时，期望收益最大，设售出商品数为 q，补货量为 r，供过于求（q≤r）时：商品由于部分不能销售出去而承受损失，具体期望值为 $\sum_{q=0}^{r} x(r-q)P(q)$；供不应求（q＞r）时：商品由于补货不足而少赚钱，其期望值为 $\sum_{q=r+1}^{\infty}(p-x)(q-r)P(q)$。

综上，损失的期望值为：$C(r)=\sum_{q=0}^{r} x(r-q)P(q)+\sum_{q=r+1}^{\infty}(p-x)(q-r)P(q)$。通过求导方法求极值，得到最佳补货量 r。

六、 模型分析与评价

6.1 优点

问题一采用 Pearson 相关系数分析品类间相关关系及 Spearman 相关系数探究单品间关联关系多进行关联分析。由于两者应用范围不同，给出数据并不能证明单品间线性关系，转而采用 Spearman 相关系数分析，弥补了相互之间的缺漏，促进对于问题的分析更加全面和完善。

问题二结合时间序列及机器学习对需求量及批发价格进行预测，寻找最优模型。通过检测多个机器学习模型预测效果，对比得到高斯过程回归 RMSE 值最低，拟合效果也与 prophet 模型结果十分接近，进一步验证方法的准确度较高。在此基础上再进行两种方法的融合，使模型预测效果进一步优化，预测结果更精确。

问题二、三对于补货定价决策分析中，明确对于销售量、需求量及补货量概念的区分，商家根据历史销售量预测需求量范围，在补货时补货量首先需要满足需求量范围，再加上损耗数量得到最终补货数量。这使方法思路更加清晰，适用性更强。

问题三对于可售品种的推测，采用数据排序筛选，结合损耗率作为成本指标，利润率作为效益指标，两个指标一同分析，增加排序的准确性与精确度，且综合性更强。

问题三在问题二的基础上增加 0—1 整数规划模型，构成双层优化模型。首先求解出各单品最优收益方案，确定定价方案；再考虑销售单品数限制，通过 0-1 整数规划，对单品组合进行优化，得到最终结果。建立的模型思路清晰，简洁易懂，且得到了较好的决策结果。

问题二在单品组合决策分析时，采用粒子群优化算法，考虑到问题一得出的单品间相互关系，给单品赋上相应的惩罚项，使决策更加合理；增加随机扰动项，可以避免陷入局部搜索，提高算法的鲁棒性与收敛速度。

分析建模前，对数据进行清洗，检查是否存在缺失值、重复值、异常值，结合 3σ 原则剔除异常数据，为进一步数据分析做好准备，提高了数据的准确性。

6.2 缺点

题目对于打折及损耗的分析不够明确，附件 4 的损耗率数据假设为运输途中损耗率，而总的损耗率是包括运损及商品因时间而品质下降，对所有损耗商品打折销售。一方面，打折数据占总销售数据比例太低，分析时准确度过低；另一方面，打折商品并不一定销售完了，所以不能进一步分析得到总的损耗数量及补货量。

附件 2 的时间信息，除了给出了销售日期，还提供了扫码销售时间，通过分析销售的具体时间点，可以进一步对顾客购买行为进行分析。

6.3 改进

增加进货量数据的收集，可以更加精准地分析损耗率，完善对打折量的考量，同时分析损耗商品打折销售的具体改进策略。通过分析历史补货量数据，可以分析历史补货策略，帮助参考得到更优的补货计划。

七、 模型推广

本文分析商超蔬菜商品历史销售数据，并对蔬菜采购补货量及定价进行预测与决策。其中，对于品类及单品销量占比、不同时间、价格的分布规律的统计分析及关联分析的研究模型思路，对于其他类型商品探究潜在销售规律与内在关联，做出更可靠的销售方

案决策也具有参考借鉴价值。

基于历史销售数据，借助时间序列及对比多类机器学习方法结果，对产品需求及批发价格进行预测分析，对于处理大量时间序列数据具有高效性、高适应性且解释性较强。选用的 prophet 时间序列模型将时间序列分解为趋势项、季节项、节假日项和误差项，不仅可以从多角度分析数据趋势，而且分解后的时间序列数据更加简洁直观，可以避免因素变动导致彼此相互影响。该方法广泛应用于经济学、气象学、交通运输、医学分析等行业，对于各类商品历史销售数据、人口增长数据、气温变化等时间数据的分析都具有较高的利用价值。

双对数模型进行回归分析，根据取对数这里特点可以将非线性关系转化为线性关系，同时可以直接通过求回归系数得到变量间弹性，分析变量间的相互变化关系。此方法除了广泛应用于计量经济学分析需求弹性、收入弹性，还可以自然科学、人口增长等多个领域探索挖掘各种变量之间的关系。

问题二、三采用的回归+预测+优化的求解最优补货定价策略的思路，通过调整利润率的定价过程，同时需要相应销售价格所对应的销售量满足预测得到的需求量范围，做出补货决策。这可以对大多数商品的补货及定价决策提供参考借鉴。

八、 结论

经过对数据的预处理后，本文首先对销售数据在时间、价格等方面统计分析分布规律，然后使用 K-means 聚类对单品进行分类，并结合 Pearson 相关系数和 Spearman 相关系数对品类和单品进行相关分析，分析商家销售方案，为题目定价补货决策做好准备。在具体决策分析时，基于 Prophet 和高斯过程回归的融合预测模型，得到了较好的效果。并结合对品类销量与定价的回归结果，从定价和补货两方面优化得到收益最优的决策方案。问题三对单品的补货决策前通过预测与 TOPSIS 评价模型排序打分，筛选出可售单品，再在问题二思路上采取 0-1 规划解决销售空间有限的问题。最后再梳理问题分析中可补充的数据，包括进货量、货架容量、供给需求情况等等，为问题后续完善提供思路借鉴，并具体增加货架容量限制，对问题三进行完善。

九、 参考文献

[1] 王晓燕,李美洲.浅谈等级相关系数与斯皮尔曼等级相关系数[J].广东轻工职业技术学院学报,2006(04):26-27.

[2] 魏勤,陈飞燕.基于双对数模型的我国机动车辆保险需求弹性分析[J].华北电力大学学报(社会科学版),2010(05):23-29.

[3] 张金隆,吴翔,徐浩轩.易变质新产品定价与补货联合决策模型[J].系统工程学报,2018,33(01):79-89.DOI:10.13383/j.cnki.jse.2018.01.008.

[4] 李德，钱颂迪，运筹学，北京：清华大学出版社，1982.

附录

附录 1：支撑材料清单

code
```
│       classfic.m
│       createFit.m
│       get2.m
│       inin.cpp
│       one_zero.cpp
│       person.m
│       predict_cost.m
│       PSO2.m
│       reg_single.m
│       sperman.m
│       tarr2.m
│       回归.do
│       数据观测.ipynb
│       数据观测获得阅读合成数据.ipynb
│       第三问.ipynb
│       第二问.ipynb
│       第二问优化.ipynb
│       第二问预测成本开始.ipynb
│       第四问.ipynb
│       聚类.ipynb
│
├──data
│   │   append4.xlsx
│   │   haed.xlsx
│   │   haed1.xlsx
│   │   y_values.csv
│   │   单品 P.xlsx
│   │   单品 R.xlsx
│   │   单品斯皮尔曼相关系数.csv
│   │   单品的销量.xlsx
│   │   大类 P.xlsx
│   │   大类 R.xlsx
│   │   按月合成数据.xlsx
│   │   按月合成数据 1_12.xlsx
│   │   正太分布检验 X.csv
│   │   正太分布检验 Y.csv
│   │   相关系数和 P 值表格.csv
│   │   箱线图.xlsx
│   │   计算相关系数的.xlsx
```

```
│   │   计算相关系数的 1to36.xlsx
│   │   附件 1.xlsx
│   │   附件 2 月度合成.xlsx
│   │   饼状图.csv
│   │   饼状图.xlsx
│   │
│   ├──第三问
│   │       ans.csv
│   │       tmp.csv
│   │       topsis 前 40 个商品.csv
│   │       单品销量价格关系.csv
│   │       单品销量价格关系.xlsx
│   │       最终选择的 33 个商品.csv
│   │
│   ├──第二问数据
│   │       价格.xlsx
│   │       回归结果.xlsx
│   │       回归输入数据.xlsx
│   │       大类商品每天价格.xlsx
│   │       大类在每一天的进货价格基于时间序列和机器学习.csv
│   │       时间序列预测销量.xlsx
│   │       答案.xlsx
│   │       高斯回归.csv
│   │
│   ├──第四问
│   │       文件.csv
│   │       第四问增加容量限制的动态规划求解答案.csv
│   │
│   └──聚类
│           聚类结果.csv
│
└──答案
    ├──第一问
    │       6 个大类皮尔逊相关系数.xlsx
    │       单品斯皮尔曼相关系数.xlsx
    │       聚类结果.csv
    │
    ├──第三问
    │       topsis 选择出前 40 商品.csv
    │       最终选择的 33 个商品.csv
    │
    ├──第二问
    │       价格.xlsx
    │       回归结果和方差检验.xlsx
```

```
│       时间序列成本价格预测.xlsx
│       时间序列预测销量.xlsx
│       第二问最终答案.xlsx
│       高斯回归的成本价格预测.csv
│       高斯回归融合时间序列成本价格预测.csv
│
└──第四问
        第四问增加容量限制的动态规划求解答案.csv
```

附录 2：Python on Visual Studio Code

编译环境 Anaconda 虚拟环境 python3.7.16

（1）数据观测

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

## ====================================  简单数据处理  ====================================
df1 = pd.read_excel('../data/附件 1.xlsx')
df2 = pd.read_csv('../data/附件 2.csv')
df3 = pd.read_excel('../data/附件 3.xlsx')
df4 = pd.read_excel('../data/附件 4.xlsx')# %%
df3.shape
df2_and_3 = df2.merge(df1, on='单品编码')
df2_and_3['销售量（暂时定义）'] = df2_and_3['销量(千克)'] * df2_and_3['销售单价(元/千克)']
df2_and_3.columns
df2_and_3_groupby = df2_and_3.groupby(['单品编码', '销售类型', '是否打折销售'])['销量(千克)'].sum().reset_index().merge(df1,
on='单品编码')
tmp = df2_and_3_groupby.groupby(['单品编码'])['销量(千克)'].sum().reset_index()
df2_and_3.groupby(['分类名称'])['销量(千克)'].sum()
df2_and_3.groupby(['分类名称'])['销量(千克)'].mean()
df2_and_3.groupby(['分类名称'])['销量(千克)'].std()
sales_table = pd.pivot_table(df2_and_3.groupby(['分类名称', '销售日期'])['销量(千克)'].sum().reset_index(),
                             values='销量(千克)', index='分类名称', columns='销售日期', aggfunc='sum').reset_index()
sales_table = sales_table.fillna(0)
sales_table.T.reset_index()
sales_table = sales_table.T.reset_index()
sales_table
sales_table.to_excel('../data/箱线图.xlsx', index=False)
df2_and_3.groupby(['分类名称'])
d2 = df2['单品编码'].unique()
d1 = df1['单品编码'].unique()
d3 = df3['单品编码'].unique()

## ====================================  完整性检验  ====================================
for i in d2:
    f = -1
    for j in d1:
        if i == j:
            f = 1
            break
    if f == -1:
        print(str(i) + '不存在')

for i in d3:
    f = -1
    for j in d1:
        if i == j:
            f = 1
            break
    if f == -1:
        print(str(i) + '不存在')

# ====================================  按照月合成数据  ====================================
df2['销售日期'] = pd.to_datetime(df2['销售日期'])
df2['月份'] = df2['销售日期'].dt.month
df2_month = df2.groupby(['单品编码', '月份编号'])['销量(千克)'].sum().reset_index()
df2_month.to_excel('../data/附件 2 月度合成.xlsx', index=False)
```

```
df2_month.columns
sales_table = pd.pivot_table(df2_month, values='销量(千克)', index='单品编码', columns='月份编号',
aggfunc='sum').reset_index()
sales_table = sales_table.fillna(0)
sales_table = pd.read_excel('../data/按月合成数据.xlsx')
sales_table = sales_table.merge(df1, on='单品编码')
sales_table.columns
cate1df = sales_table.groupby(['分类名称'])['月份 1', '月份 2', '月份 3', '月份 4', '月份 5', '月份 6', '月份 7', '月份 8', '
月份 9', '月份 10', '月份 11', '月份 12', '月份 13', '月份 14', '月份 15', '月份 16', '月份 17', '月份 18','月份 19', '月份 20', '
月份 21', '月份 22', '月份 23', '月份 24', '月份 25', '月份 26', '月份 27','月份 28', '月份 29', '月份 30', '月份 31', '月份 32',
'月份 33', '月份 34', '月份 35', '月份 36'].sum().reset_index()
df1
cate1df.head(1)

# ======================================    六个大类 以 1-36 月单位为横坐标 销量纵坐
标  ======================================
plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题
sample_names = cate1df['分类名称']
features = cate1df.drop('分类名称', axis=1)
plt.figure(figsize=(10, 6))  # 设置图形大小

for sample_name, row in zip(sample_names, features.values):
    plt.plot(range(1, 37), row, label=sample_name)

plt.xlabel('月份编号')  # 横坐标标签
plt.ylabel('销量')    # 纵坐标标签
plt.title('6 个大类关于月份(1-36)的销售情况')  # 图标题
plt.legend()  # 添加图例

plt.show()

# ======================================  简单处理数据 ====>>> 然后转到 matlab 求解相关系
数  ======================================
x = cate1df.iloc[1, 1:]
y = cate1df.iloc[2, 1:]
plt.figure(figsize=(5, 5))
plt.scatter(x, y)

x.to_csv('../data/正太分布检验 X.csv', index=False)
y.to_csv('../data/正太分布检验 Y.csv', index=False)
cate1df.T.to_excel('../data/计算相关系数的 1to36.xlsx', index=False)
fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(10, 6))
for ax in axes.flat:
    ax.set(xlabel='', ylabel='')
row_combinations = [(i, j) for i in range(6) for j in range(i + 1, 6)]

# ======================================  查看品类之间相关散点图  ======================================
for idx, (i, j) in enumerate(row_combinations):
    print(idx)
    x = cate1df.iloc[i, 1:]
    y = cate1df.iloc[j, 1:]
    ax = axes[idx // 5, idx % 5]
    ax.scatter(x, y)
    ax.set_title(f'组合{i+1}-{j+1}')
plt.tight_layout()

plt.show()


# ======================================  按照 1-12 月度合成的销量折线图  ======================================
df2_1_12 = df2.groupby(['单品编码', '月份'])['销量(千克)'].sum().reset_index()
df2_1_12
sales_table_1_12 = pd.pivot_table(df2_1_12, values='销量(千克)', index='单品编码', columns='月份',
aggfunc='sum').reset_index()
sales_table_1_12 = sales_table_1_12.fillna(0)
sales_table_1_12 = sales_table_1_12.merge(df1, on='单品编码')
sales_table_1_12 = pd.read_excel('../data/按月合成数据 1_12.xlsx')
sales_table_1_12.columns
cate1df_1_12 = sales_table.groupby(['分类名称'])['月份 1', '月份 2', '月份 3', '月份 4', '月份 5', '月份 6', '月份 7', '月份
8', '月份 9','月份 10', '月份 11', '月份 12'].sum().reset_index()
```

```
cate1df_1_12.columns
plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题

sample_names = cate1df_1_12['分类名称']
features = cate1df_1_12.drop('分类名称', axis=1)

plt.figure(figsize=(10, 6))  # 设置图形大小

for sample_name, row in zip(sample_names, features.values):
    plt.plot(range(1, 13), row, label=sample_name)

plt.xlabel('月份编号')  # 横坐标标签
plt.ylabel('销量')    # 纵坐标标签
plt.title('6 个大类关于月份(1-12)的销售情况')  # 图标题
plt.legend()  # 添加图例

plt.show()


## ======================================  相关系数看看  ======================================

x = cate1df_1_12.iloc[1]
y = cate1df_1_12.iloc[4]
plt.scatter(x, y)
cate1df_1_12 = cate1df_1_12.T
cate1df_1_12.to_excel('../data/计算相关系数的.xlsx', index=False)
```

（2）数据观测月度获得合成数据

```
from fbprophet import Prophet
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns


## ======================================  合成季节  ======================================
df2 = pd.read_csv('../data/附件 2.csv')
df1 = pd.read_excel('../data/附件 1.xlsx')
df2['销售日期'] = pd.to_datetime(df2['销售日期'])
df2['月份'] = df2['销售日期'].dt.month
df2.columns
month_to_season = {
    1: '冬季',
    2: '冬季',
    3: '春季',
    4: '春季',
    5: '春季',
    6: '夏季',
    7: '夏季',
    8: '夏季',
    9: '秋季',
    10: '秋季',
    11: '秋季',
    12: '冬季'
}
df2['季节'] = df2['月份'].map(month_to_season)
df2 = df2.merge(df1, on = '单品编码' )
tmp = df2.groupby(['分类名称', '季节'])['销量(千克)'].sum().reset_index()
tmp.columns


## ======================================  绘制季度合成销量  ======================================
plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题
```

```
categories = tmp['分类名称'].unique()
plt.figure(figsize=(10, 7))
for category in categories:
    subset = tmp[tmp['分类名称'] == category]
    x = subset['季节'].sort_values()
    y = subset['销量(千克)']
    plt.plot(x, np.log(y), label=category)

plt.title('不同分类销量(取对数)随季节变化折线图')
plt.xlabel('季节')
plt.ylabel('销量(千克)(取对数)')
plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1.0))
plt.show()

## 简单数据处理
tmp.groupby('分类名称')['销量(千克)'].sum().reset_index().to_excel('../data/饼状图.xlsx', index=False)
tmp = df2.groupby(['分类名称', '单品名称'])['销量(千克)'].sum().reset_index()
tmp['分类名称'].unique()
tmp1 = tmp[tmp['分类名称'] == '食用菌']
tmp1.drop(columns='分类名称').to_csv('../data/饼状图.csv', index=False)
df2g = df2.groupby(['分类名称', '销售日期'])['销量(千克)'].sum().reset_index()
df2g['分类名称'].unique()
df21 = df2g[df2g['分类名称'] == '水生根茎类']
df21 = df21.drop(columns='分类名称')
```

```
# ====================================  Prophet 预测  ====================================
all_dates = pd.date_range(start='2020-07-01', end='2023-06-30', freq='D')
df_all_dates = pd.DataFrame({'销售日期': all_dates})

# 把两个数据框合并
df21['销售日期'] = pd.to_datetime(df21['销售日期'])
df_merged = pd.merge(df_all_dates, df21, on='销售日期', how='left')
# 使用 0 填充缺失的销量数据
df_merged['销量(千克)'].fillna(0, inplace=True)
# 准备用于 Prophet 的数据
df_prophet = df_merged.rename(columns={'销售日期': 'ds', '销量(千克)': 'y'})
# 使用 Prophet 进行预测
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=0)  # 创建未来日期，这里预测未来 365 天
forecast = model.predict(future)
model.plot(forecast)
model.plot_components(forecast)
```

```
# ====================================  剔除多余数据  ====================================
df2.groupby(['销量(千克)']).count().reset_index()[['销量(千克)', '销售日期']]
# 绘制箱线图
plt.figure(figsize=(20, 1))
plt.boxplot(df2['销量(千克)'], vert=False)
# 设置图标题和横轴标签
plt.title('销量(千克) 箱线图')
plt.xlabel('销量(千克)')
# 显示图形
plt.show()
mean = df2['销量(千克)'].mean()
std = df2['销量(千克)'].std()
ub = mean + 3 * std
lb = mean - 3 * std
df2 = df2[df2['销量(千克)'] < ub]
df2 = df2[df2['销量(千克)'] > lb]
df2.to_csv('../data/附件 2.csv')
df2 = pd.read_csv('../data/附件 2.csv')
df2.columns
```

```
## ====================================  简单数处理  ====================================

tmp = df2.groupby(['单品名称', '月份编号']).agg({'销量(千克)':'sum'}).reset_index()
name = tmp['单品名称'].unique()

sales_table = pd.pivot_table(tmp, values='销量(千克)', index='月份编号', columns='单品名称',
aggfunc='sum').reset_index().fillna(0)
sales_table.to_csv('../data/单品斯皮尔曼相关系数.csv', index=False, encoding='utf-8-sig')

excel_file = "../data/单品 R.xlsx"
df = pd.read_excel(excel_file, header=None)
list_of_columns = df.values.tolist()
dfP = pd.read_excel("../data/单品 P.xlsx", header=None)
listP = dfP.values.tolist()

x = []
y = []

for level in [round(x, 2) for x in  [i/100 for i in range(80, 101)]]:
    count = 0
    for i in range(246):
        for j in range(i + 1, 246):
            if (abs(list_of_columns[i][j]) >= level):
                # print(i, j)
                count += 1
    x.append(level)
    y.append(count)




# ====================================  找出想关心较强的 18 个单品  ====================================

danpin = []

for i in range(246):
        for j in range(i + 1, 246):
            if (abs(list_of_columns[i][j]) >= 0.8):
                if tmp[tmp['单品名称'] == name[i]].shape[0] >= 24 and tmp[tmp['单品名称'] == name[j]].shape[0] >= 24:
                    danpin.append([name[i], name[j], list_of_columns[i][j], listP[i][j]])
pd.DataFrame(danpin, columns=['x', 'y', 'R', 'P']).to_csv('../data/相关系数和 P 值表格.csv', index=False,
encoding='utf-8-sig')
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
fig, axes = plt.subplots(9, 2, figsize=(12, 18))
fig.subplots_adjust(hspace=0.5)

for i in range(len(danpin)):
    t1 = tmp[tmp['单品名称'] == danpin[i][0]]
    t2 = tmp[tmp['单品名称'] == danpin[i][1]]
    spR = danpin[i][2]
    spP = danpin[i][3]
    x_tmp = t1['月份编号']
    y_tmp = t1['销量(千克)']
    x_tmp2 = t2['月份编号']
    y_tmp2 = t2['销量(千克)']
    row = i // 2
    col = i % 2
    axes[row, col].plot(x_tmp, y_tmp, label=f'{danpin[i][0]} (相关系数={spR:.2f})', marker='o', linestyle='-')
    axes[row, col].plot(x_tmp2, y_tmp2, label=f'{danpin[i][1]} (P 检验值={spP:.2f})', marker='x', linestyle='-')
    axes[row, col].set_title(f'销量随月份变化 - {danpin[i][0]} vs {danpin[i][1]}')
    axes[row, col].set_ylabel('销量(千克)')

    axes[row, col].legend()

plt.show()

## ====================================  相关系数  ====================================
## ====================================  由 matlab 导入  ====================================
R = [[1.0000,    0.4841 ,   0.4721 ,  -0.4660 ,   0.4589  ,  0.6528],
```

```
    [0.4841 ,   1.0000 ,   0.7469 ,  -0.0345  ,  0.6238 ,   0.5461],
    [0.4721,    0.7469,    1.0000    ,0.0581   , 0.4255 ,   0.4287],
   [-0.4660  , -0.0345  ,  0.0581  ,  1.0000  , -0.1914 , -0.4089],
    [0.4589   , 0.6238  ,  0.4255 ,  -0.1914   , 1.0000 ,   0.5755],
    [0.6528  ,  0.5461  ,  0.4287 ,  -0.4089  ,  0.5755 ,   1.0000]]
R = [[1.0000,    0.4841 ,   0.4721 ,  0.4660 ,   0.4589  ,  0.6528],
    [0.4841 ,   1.0000 ,   0.7469 ,  0.0345  ,  0.6238 ,   0.5461],
    [0.4721,    0.7469,    1.0000    ,0.0581   , 0.4255 ,   0.4287],
   [0.4660  , 0.0345  ,  0.0581  ,  1.0000  , 0.1914 , 0.4089],
    [0.4589   , 0.6238  ,  0.4255 ,  0.1914   , 1.0000 ,   0.5755],
    [0.6528  ,  0.5461  ,  0.4287 ,  0.4089  ,  0.5755 ,   1.0000]]
labels = ['水生根茎类', '花叶类', '花菜类', '茄类', '辣椒类', '食用菌']

## ======================================  绘制热力图  ======================================
plt.figure(figsize=(8, 6))
sns.heatmap(R, annot=True, cmap="YlGnBu", fmt=".2f", xticklabels=labels, yticklabels=labels)
plt.title('大类相关系数热力图')
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
# 创建热力图
plt.figure(figsize=(10, 8))
sns.heatmap(R, annot=True, cmap="YlGnBu", fmt=".2f")
plt.title('各品种斯皮尔曼相关系数热力图')
plt.show()

R = [[abs(x) for x in row[:20]] for row in list_of_columns[:20]]
```

（3）第二问

```
from fbprophet import Prophet
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns
df2 = pd.read_csv('../data/附件 2.csv')
df3 = pd.read_excel('../data/附件 3.xlsx')
df4 = pd.read_excel('../data/附件 4.xlsx')
className = ['水生根茎类', '花叶类', '花菜类', '茄类', '辣椒类', '食用菌']

## ======================================  绘制所有大类的散点图  ======================================
## ======================================  价格与销量  ======================================
for name in className:
    tmp = df2[df2['分类名称'] == name].groupby(['分类名称', '销售单价(元/千克)'])['销量(千克)'].sum().reset_index()
    x = tmp['销售单价(元/千克)']
    y = tmp['销量(千克)']
    plt.scatter(x, y, label=name)
plt.legend()
plt.title('大类的价格-销量 散点图')
plt.xlabel('价格')
plt.ylabel('销售总量')
plt.grid(True)
plt.show()


## ======================================  同上 单品 ======================================
className = ['水生根茎类', '花叶类', '花菜类', '茄类', '辣椒类', '食用菌']
name = className[4]
tmp = df2[df2['分类名称'] == name].groupby(['分类名称', '销售单价(元/千克)'])['销量(千克)'].sum().reset_index()
x = tmp['销售单价(元/千克)']
y = tmp['销量(千克)']
plt.scatter(x, y, label=name)
plt.title(f'{name}  大类的价格-销量 散点图')
plt.xlabel('价格')
plt.ylabel('销售总量')
plt.grid(True)
plt.show()
```

```
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
writer = pd.ExcelWriter("../data/第二问数据/回归输入数据.xlsx")

# ======================================  绘图检验销量和价格关系 ======================================
fig, axs = plt.subplots(3, 2, figsize=(12, 12))

# 循环遍历每个类别并绘制数据
for i, name in enumerate(className):
    tmp = df2[df2['分类名称'] == name]
    tmp = tmp.groupby(['月份编号']).agg({'销售单价(元/千克)': 'mean', '销量(千克)': 'sum'}).reset_index()
    x = tmp['月份编号']
    y1 = np.log(tmp['销量(千克)'])
    y2 = np.log(tmp['销售单价(元/千克)'] * tmp['销售单价(元/千克)'])
    row = i // 2
    col = i % 2
    axs[row, col].plot(x, y1, label=f'{name} 销量对数', marker='o', linestyle='-')
    axs[row, col].plot(x, y2, label=f'{name} 销售单价平方对数', marker='x', linestyle='-')
    axs[row, col].legend()
    axs[row, col].set_title(f'{name} 月份销量和销售单价平方对数图')
    axs[row, col].set_xlabel('月份编号')
    axs[row, col].set_ylabel('对数值')
    axs[row, col].grid(True)
    dftmp = tmp[['销售单价(元/千克)', '销量(千克)']]
    dftmp['p'] = np.log(dftmp['销售单价(元/千克)'])
    dftmp['q'] = np.log(dftmp['销量(千克)'])
    dftmp = dftmp[['p', 'q']]
    dftmp.to_excel(writer, sheet_name=f'{name}')
plt.tight_layout()
plt.show()
writer.save()
writer.close()

## ======================================  数据处理  ======================================

dftmp = tmp[['销售单价(元/千克)', '销量(千克)']]
dftmp['p'] = np.log(dftmp['销售单价(元/千克)'])
dftmp['q'] = np.log(dftmp['销量(千克)'])
y1 = np.log(tmp['销量(千克)'])
y2 = np.log(tmp['销售单价(元/千克)'] * tmp['销售单价(元/千克)'])
data = pd.DataFrame({'月份编号': tmp['月份编号'], '销量对数': y1, '销售单价平方对数': y2})
data.to_csv('../data/y_values.csv', index=False)
plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题
plt.figure(figsize=(20, 1))
plt.boxplot(df2['销售单价(元/千克)'], vert=False)
plt.title('销售单价(元/千克) 箱线图')
plt.xlabel('销售单价(元/千克)')
plt.show()
df2['销售单价(元/千克)'].describe()
mean = df2['销售单价(元/千克)'].mean()
std = df2['销售单价(元/千克)'].std()
df1 = pd.read_excel('../data/附件 1.xlsx')
df3 = pd.read_excel('../data/附件 3.xlsx')
df3 = df3.merge(df1, on='单品编码').reset_index()
df3 = df3[['单品名称', '日期', '批发价格(元/千克)']]
df3 = df3.rename(columns={'日期':'销售日期'})
df3['销售日期'] = pd.to_datetime(df3['销售日期'])
df22 = df2[['销售日期','分类名称', '单品名称', '销量(千克)', '销售单价(元/千克)', '是否打折销售', '月份编号']]
df22['销售日期'] = pd.to_datetime(df22['销售日期'])
merged_df = pd.merge(df22, df3, on=['单品名称', '销售日期'], how='left')
merged_df['销售金额'] = merged_df['销量(千克)'] * merged_df['销售单价(元/千克)']
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['利润'] / merged_df['批发价格(元/千克)']
tt = merged_df.groupby(['分类名称', '月份编号']).agg({'销量(千克)':'sum', '利润率':'mean'}).reset_index()


## ======================================  绘制利润和销量关系  ======================================

plt.rcParams["font.sans-serif"] = ["SimHei"]
```

```
plt.rcParams["axes.unicode_minus"] = False
fig, axes = plt.subplots(3, 2, figsize=(20, 12))
for i, name in enumerate(className):
    row, col = i // 2, i % 2
    ax = axes[row, col]
    tmp = tt[tt['分类名称'] == name]
    x = tmp['月份编号']
    y1 = np.log(tmp['销量(千克)']) - 5
    y2 = np.log(tmp['利润率'] * tmp['利润率'])
    ax.plot(x, y1, label='销量对数', marker='o', linestyle='-', color='blue')
    ax.plot(x, y2, label='利润率平方对数', marker='x', linestyle='-', color='red')
    ax.set_title(f'{name} 月份销量和利润率平方对数图')
    ax.set_xlabel('月份编号')
    ax.set_ylabel('对数值')
    ax.grid(True)
    ax.tick_params(axis='x', labelsize=10)
    ax.tick_params(axis='y', labelsize=10)

plt.tight_layout()
plt.show()

name = className[5]
tmp = tt[tt['分类名称'] == name]
y1 = tmp['销量(千克)']
y2 = tmp['利润率']
data = pd.DataFrame({'月份编号': tmp['月份编号'], '销量对数': y1, '销售单价平方对数': y2})
data.to_csv('../data/y_values.csv', index=False)
```

（4）第二问预测成本

```
from fbprophet import Prophet
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns
df3 = pd.read_excel('../data/附件 3.xlsx')
df2 = pd.read_csv('../data/附件 2.csv')
df1 = pd.read_excel('../data/附件 1.xlsx')

df3 = df3.merge(df1, on='单品编码')
className = ['水生根茎类', '花叶类', '花菜类', '茄类', '辣椒类', '食用菌']



## ====================================  六个大类预测  ====================================
cost = []
for name in className:
    df33 = df3[df3['分类名称'] == name]

    plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
    plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题

    # 绘制箱线图
    plt.figure(figsize=(20, 1))
    plt.boxplot(df33['批发价格(元/千克)'], vert=False)

    # 设置图标题和横轴标签
    plt.title('销售单价(元/千克) 箱线图')
    plt.xlabel('销售单价(元/千克)')

    # 显示图形
    plt.show()

    # 提出右侧异常值

    mean = df33['批发价格(元/千克)'].mean()
    std = df33['批发价格(元/千克)'].std()
    df33 = df33[df33['批发价格(元/千克)'] < mean + 3 * std]
```

```
    # 聚合取均值
    df33 = df33.rename(columns={'日期':'销售日期'})
    df33 = df33.groupby(['销售日期'])['批发价格(元/千克)'].mean().reset_index()

    # 创建所有日期范围的数据框
    all_dates = pd.date_range(start='2020-07-01', end='2023-06-30', freq='D')
    df_all_dates = pd.DataFrame({'销售日期': all_dates})

    # 把两个数据框合并
    df33['销售日期'] = pd.to_datetime(df33['销售日期'])
    df_merged = pd.merge(df_all_dates, df33, on='销售日期', how='left')

    # 使用非空值填充空值
    df_merged['批发价格(元/千克)'].fillna(method='ffill', inplace=True)
    df_merged.to_csv(f'../data/每个大类的时间序列/{name}.csv', index=False, encoding='utf-8-sig')
    # 准备用于 Prophet 的数据
    df_prophet = df_merged.rename(columns={'销售日期': 'ds', '批发价格(元/千克)': 'y'})

    # 使用 Prophet 进行预测
    model = Prophet()
    model.fit(df_prophet)
    future = model.make_future_dataframe(periods=7)  # 创建未来日期，这里预测未来 365 天
    forecast = model.predict(future)

    # 绘制预测结果
    fig = model.plot(forecast)

    # 显示图形
    plt.show()

    cost.append(forecast['yhat'].tail(7).tolist())



cost_df = pd.DataFrame(cost, columns=['7 月 1 号', '7 月 2 号', '7 月 3 号', '7 月 4 号', '7 月 5 号', '7 月 6 号', '7 月 7 号'],
index=className)
cost_dfcost_mat = pd.read_excel('../data/每个大类的时间序列/价格.xlsx')


## ======================================  由 matlab 导出而来  ======================================
cost_mat = [[11.59591,  11.46544,   11.31933,11.17536,  11.04113    ,10.91863   ,10.80754],
[3.485820518    ,3.489162824,   3.513023887 ,3.543306939    ,3.573742151    ,3.602011255,   3.62749577],
[7.809982889    ,7.745292003,   7.72008064  ,7.70673506 ,7.695140599    ,7.682512679    ,7.66843736],
[4.479476232    ,4.492624554,   4.520529236,    4.555180137 ,4.59155921,    4.627059069 ,4.660530044],
[5.979963478    ,5.970563944,   5.974027474 ,5.984954813    ,5.999035271    ,6.013829038,   6.028185061],
[4.808871556    ,4.845635107,   4.882472499 ,4.916591318,   4.947386956 ,4.975071537    ,5.000048583]
]

pd.DataFrame(cost_mat, columns=['7 月 1 号', '7 月 2 号', '7 月 3 号', '7 月 4 号', '7 月 5 号', '7 月 6 号', '7 月 7 号'],
index=className).to_csv('../data/第二问数据/高斯回归.csv', encoding='utf-8-sig')

## ======================================  计算都得到融合成本  ======================================

len(cost_mat)
cost_mean = []
for i in range(len(cost)):
    cost_mean.append([])
    for j in range(len(cost[0])):
        cost_mean[i].append((cost[i][j] + cost_mat[i][j]) / 2)

cost_mean

cost_df = pd.DataFrame(cost_mean, columns=['7 月 1 号', '7 月 2 号', '7 月 3 号', '7 月 4 号', '7 月 5 号', '7 月 6 号', '7 月 7 号
'], index=className)
cost_df.to_csv('../data/第二问数据/大类在每一天的进货价格基于时间序列和机器学习.csv', encoding='utf-8-sig')
```

```
## ====================================  预测销量  ====================================
writer = pd.ExcelWriter("../data/第二问数据/时间序列预测销量.xlsx")

for i in range(len(className)):
    name = className[i]
    df22 = df2[df2['分类名称'] == name]

    plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
    plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题

    # 绘制箱线图
    plt.figure(figsize=(20, 1))
    plt.boxplot(df22['销量(千克)'], vert=False)

    # 设置图标题和横轴标签
    plt.title(f'{name}  销量(千克) 箱线图')
    plt.xlabel('销量(千克)')

    # 显示图形
    plt.show()



    # 提出右侧异常值

    mean = df22['销量(千克)'].mean()
    std = df22['销量(千克)'].std()
    df22 = df22[df22['销量(千克)'] < mean + 3 * std]
    df22 = df22[df22['销量(千克)'] > mean - 3 * std]
    # 绘制箱线图
    plt.figure(figsize=(20, 1))
    plt.boxplot(df22['销量(千克)'], vert=False)

    # 设置图标题和横轴标签
    plt.title(f'{name}  销量(千克) 箱线图')
    plt.xlabel('销量(千克)')

    # 显示图形
    plt.show()

    # 聚合取均值
    df22 = df22.groupby(['销售日期'])['销量(千克)'].sum().reset_index()

    # 创建所有日期范围的数据框
    all_dates = pd.date_range(start='2020-07-01', end='2023-06-30', freq='D')
    df_all_dates = pd.DataFrame({'销售日期': all_dates})

    # 把两个数据框合并
    df22['销售日期'] = pd.to_datetime(df22['销售日期'])
    df_merged = pd.merge(df_all_dates, df22, on='销售日期', how='left')

    # 使用非空值填充空值
    df_merged['销量(千克)'].fillna(method='ffill', inplace=True)
    # df_merged.to_csv(f'../data/每个大类的时间序列/{name}.csv', index=False, encoding='utf-8-sig')
    # 准备用于 Prophet 的数据
    df_prophet = df_merged.rename(columns={'销售日期': 'ds', '销量(千克)': 'y'})

    # 使用 Prophet 进行预测
    model = Prophet()
    model.fit(df_prophet)
    future = model.make_future_dataframe(periods=7)  # 创建未来日期，这里预测未来 365 天
    forecast = model.predict(future)

    # 绘制预测结果
    fig = model.plot(forecast)

    # 显示图形
    plt.show()
```

```
    #  ====================================  计算上下界  ====================================
    pre_ans = forecast[['yhat', 'yhat_lower', 'yhat_upper']].tail(7)
    pre_ans['lower_delta'] = (pre_ans['yhat'] - pre_ans['yhat_lower']) / 3
    pre_ans['upper_delta'] = (pre_ans['yhat_upper'] - pre_ans['yhat']) / 3
    pre_ans['yhat_lower'] = pre_ans['yhat'] - pre_ans['lower_delta']
    pre_ans['yhat_upper'] = pre_ans['yhat'] + pre_ans['upper_delta']
    pre_ans = pre_ans.drop(columns=['upper_delta', 'lower_delta'])

    pre_ans.to_excel(writer, sheet_name=f'{name}')

writer.save()
writer.close()
```

（5）第二问优化

```
from fbprophet import Prophet
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns

## ====================================  计算优化模型需要使用的上下界等等  ====================================
beita = [
        [9.905795, -1.380787],
        [9.391347, -.4751426],
        [8.767719, -.8079629],
        [7.718134, -.656666],
        [8.401708, -.2846255],
        [11.55355, -1.612381]
]

# 给出第 i 个大类的价格推出销售量
def func(i, x):
    x = np.log(x)
    p0 = beita[i][0]
    p1 = beita[i][1]
    return np.exp(p0+p1*x) / 30

# 根据销量上下界反推价格:
def getp(i, x):
    x = np.log(x * 30)
    p0 = beita[i][0]
    p1 = beita[i][1]
    return np.exp((x - p0) / p1)

# 分类名称
className = ['水生根茎类', '花叶类', '花菜类', '茄类', '辣椒类', '食用菌']

# 每个商品每天的价格
cost = pd.read_excel('../data/第二问数据/大类商品每天价格.xlsx').values.tolist()

# 每日销量上下界
sales_lower = []
sales_upper = []
for i in range(len(className)):
    name = className[i]
    tmp = pd.read_excel('../data/第二问数据/时间序列预测销量.xlsx', sheet_name=i)
    sales_lower.append(tmp['yhat_lower'].to_list())
    sales_upper.append(tmp['yhat_upper'].to_list())

# 初始定价上下界
confirm_cost_lw = []
confirm_cost_up = []
for i in range(len(className)):
    confirm_cost_lw.append([])
    confirm_cost_up.append([])
    for j in range(7):
        confirm_cost_lw[i].append(getp(i, sales_upper[i][j]))
```

```
        confirm_cost_up[i].append(getp(i, sales_lower[i][j]))

# 修正定价上下界
confirm_cost_up[4][0] = 8
for i in range(6):
    for j in range(7):
        confirm_cost_lw[i][j] = max(confirm_cost_lw[i][j], cost[i][j])

# 利润率上下界
lower_p_margin = []
upper_p_margin = []

for i in range(6):
    lower_p_margin.append([])
    upper_p_margin.append([])
    for j in range(7):
        lower_p_margin[i].append((confirm_cost_lw[i][j] - cost[i][j]) / cost[i][j])
        upper_p_margin[i].append((confirm_cost_up[i][j] - cost[i][j]) / cost[i][j])

## ====================================  计算打折率  ====================================
df2 = pd.read_csv('../data/附件 2.csv').drop(columns='Unnamed: 0')
print(df2.columns)
df2.head()
discount_rate = [] # 有多少打折
discount = [] # 打折的优惠程度
for name in className:

    tmp = df2[df2['分类名称'] == name]
    tmp = tmp[tmp['销售类型'] == '销售']

    tmp1 = tmp[tmp['是否打折销售'] == '是']
    print(tmp1.shape[0], tmp.shape[0])
    discount_rate.append(tmp1.shape[0] / tmp.shape[0])

    tmp = tmp.groupby(['是否打折销售'])['销售单价(元/千克)'].mean().reset_index()['销售单价(元/千克)']
    discount.append(min(tmp[0], tmp[1]) / max(tmp[0], tmp[1]))
```

（6）第三问

```
from fbprophet import Prophet
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns
import random
from sklearn.linear_model import LinearRegression
from scipy.stats import spearmanr

df2 = pd.read_csv('../data/附件 2.csv')
df1 = pd.read_excel('../data/附件 1.xlsx')
df2 = df2.sort_values(by='销售日期', ascending=True)
df2['销售日期'] = pd.to_datetime(df2['销售日期'])



## ====================================  选择售卖时间大于 6.24 的  ====================================
df22 = df2.iloc[869888:]
className = df22['单品名称'].unique().tolist()

df3 = pd.read_excel('../data/附件 3.xlsx')
df3 = df3.rename(columns={'日期':'销售日期'})
df3['销售日期'] = pd.to_datetime(df3['销售日期'])
dfname = df22.groupby(['单品名称'])['销量(千克)'].sum().reset_index()['单品名称']
dfnew = pd.merge(dfname, df2, on='单品名称', how='left')
print(dfnew['单品名称'].unique().shape)
dfnew[dfnew['单品名称'] == '鲜木耳(份)']
dfnew = dfnew.merge(df3, on=['单品编码', '销售日期'])
```

```
# ======================================  用 topsis 提取在 6-24~6-30 在售的前 40 个商品  ======================================

dfnew = dfnew[['单品名称', '销售日期', '销量(千克)', '销售单价(元/千克)','销售类型', '是否打折销售', '月份编号', '月份','批发价格(元/千克)']]
df4 = pd.read_excel('../data/附件 4.xlsx', sheet_name=1)
dfnew['净利润'] = dfnew['销量(千克)'] * (dfnew['销售单价(元/千克)'] - dfnew['批发价格(元/千克)'])
SCmoney = dfnew.groupby(['单品名称'])['净利润'].sum().reset_index()
SCmoney = SCmoney.merge(df4, on='单品名称').drop(columns=['单品编码'])
features = SCmoney[['净利润', '损耗率(%)']].values
norm_features = features / np.linalg.norm(features, axis=0)

# 应用权重 [0.9, 0.1]
weights = np.array([0.9, 0.1])
weighted_features = norm_features * weights

# 计算理想解和负理想解
# 对于“净利润”，越高越好，所以理想解是最大值，负理想解是最小值
# 对于“损耗率(%)”，越低越好，所以理想解是最小值，负理想解是最大值
ideal_solution = np.array([np.max(weighted_features[:, 0]), np.min(weighted_features[:, 1])])
nadir_solution = np.array([np.min(weighted_features[:, 0]), np.max(weighted_features[:, 1])])
# 计算与理想解和负理想解的欧氏距离
dist_ideal = np.linalg.norm(weighted_features - ideal_solution, axis=1)
dist_nadir = np.linalg.norm(weighted_features - nadir_solution, axis=1)
# 计算相似度得分
similarity_score = dist_nadir / (dist_ideal + dist_nadir)
SCmoney['TOPSIS 得分'] = similarity_score
SCmoney_sorted = SCmoney.sort_values(by='TOPSIS 得分', ascending=False)
nameClassDF = SCmoney_sorted[:40]
nameClass = SCmoney_sorted[:40]['单品名称'].to_list()
SCmoney_sorted[:40].to_csv('../data/第三问/topsis 前 40 个商品.csv', index=False, encoding='utf-8-sig')
df2 = df2.merge(df3, on=['单品编码', '销售日期'])



# ======================================  Prophet 预测销量  ======================================
cost = []
for i in range(len(nameClass)):
    name = nameClass[i]
    name
    df22 = df2[df2['单品名称'] == name]
    df22 = df22[['销售日期', '批发价格(元/千克)']].groupby(['销售日期'])['批发价格(元/千克)'].mean().reset_index()
    plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
    plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题
    all_dates = pd.date_range(start='2020-07-01', end='2023-06-30', freq='D')
    df_all_dates = pd.DataFrame({'销售日期': all_dates})
    df22['销售日期'] = pd.to_datetime(df22['销售日期'])
    df_merged = pd.merge(df_all_dates, df22, on='销售日期', how='left')
    df_prophet = df_merged.rename(columns={'销售日期': 'ds', '批发价格(元/千克)': 'y'})
    model = Prophet(changepoint_prior_scale=1)
    model.fit(df_prophet)
    future = model.make_future_dataframe(periods=1)
    forecast = model.predict(future)
    fig = model.plot(forecast)
    plt.show()
    cost.append(forecast.tail(1)['yhat'].values[0])
costcp = cost.copy()
cost



## ======================================  画散点图  ======================================
for i in range(len(nameClass)):
    name = nameClass[i]
    name
    df22 = df2[df2['单品名称'] == name]
    df22 = df22[['销售日期', '批发价格(元/千克)']].groupby(['销售日期'])['批发价格(元/千克)'].mean().reset_index()
    plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
    plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题
    all_dates = pd.date_range(start='2020-07-01', end='2023-06-30', freq='D')
    df_all_dates = pd.DataFrame({'销售日期': all_dates})
    df22['销售日期'] = pd.to_datetime(df22['销售日期'])
    df_merged = pd.merge(df_all_dates, df22, on='销售日期', how='left')
```

```
    df_prophet = df_merged.rename(columns={'销售日期': 'ds', '批发价格(元/千克)': 'y'})
    print(name)
    plt.figure(figsize=(20, 4))
    plt.scatter(df_prophet['ds'], df_prophet['y'])
    plt.show()


## ====================================  探索价格和销量关系  ====================================
tmp = nameClassDF.merge(df2, on='单品名称')[['销售日期', '销售单价(元/千克)', '销量(千克)']].groupby(['销售日期']).agg({'
销售单价(元/千克)': 'mean', '销量(千克)': 'sum'}).reset_index()

plt.figure(figsize=(20, 8))
plt.scatter(tmp['销售单价(元/千克)'], tmp['销量(千克)'])
date_mask = (tmp['销售日期'] >= '2022-07-01') & (tmp['销售日期'] <= '2023-06-30')
selected_data = tmp[date_mask]
plt.figure(figsize=(20, 8))
plt.plot(selected_data['销售日期'], selected_data['销售单价(元/千克)'], label='销售单价')
plt.plot(selected_data['销售日期'], np.log(selected_data['销量(千克)']), label='exp(销量)')
plt.title('销售单价和销量趋势 (2022-07-01 到 2023-06-30)')
plt.legend()
plt.xlabel('销售日期')
plt.ylabel('值')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
selected_data[['销售单价(元/千克)', '销量(千克)']].to_excel('../data/第三问/单品销量价格关系.xlsx', index=False)
price = selected_data['销售单价(元/千克)'].values.reshape(-1, 1)
quantity = selected_data['销量(千克)'].values.reshape(-1, 1)

# ====================================  创建线性回归模型  ====================================

## ======================================  计算  并且 回归  价格和需求的  方
程  ======================================

xishu = []
for i in range(len(nameClass)):
    name = nameClass[i]
    tmp = df22[df22['单品名称'] == name]
    tmp = tmp.groupby(['销售日期']).agg({'销售单价(元/千克)': 'mean', '销量(千克)': 'sum'}).reset_index()

    # ================================================ 散点图 ================================================ #

    # plt.figure(figsize=(10, 4))
    # plt.scatter(tmp['销售单价(元/千克)'], tmp['销量(千克)'])
    # # 添加标题
    # plt.title(f'{name}散点图')
    # plt.show()

    # ================================================ 折线图 ================================================ #

    plt.figure(figsize=(20, 2))
    plt.plot(tmp['销售日期'], np.exp(tmp['销量(千克)']))
    plt.plot(tmp['销售日期'], np.log(tmp['销售单价(元/千克)']))
    # 添加标题
    plt.title(f'{name}查看直线关系')
    plt.show()
    # 选择销售单价和销量数据
    price = tmp['销售单价(元/千克)'].values.reshape(-1, 1)
    quantity = tmp['销量(千克)'].values.reshape(-1, 1)
    # 创建线性回归模型
    model = LinearRegression()

    # 拟合模型
    model.fit(price, quantity)

    # 预测销量
    predicted_quantity = model.predict(price)
    # ================================================ 折线图 ================================================ #
    # 创建一个 20x8 大小的图形
    plt.figure(figsize=(10, 4))
```

```
    plt.scatter(price, quantity, label='实际销量')
    plt.plot(price, predicted_quantity, color='red', label='拟合线')
    plt.title(f'{name}销售单价和销量关系拟合')
    plt.legend()
    plt.xlabel('销售单价(元/千克)')
    plt.ylabel('销量(千克)')
    plt.show()

    # ================================================ 写函数 ================================================ #
    # 获取线性回归模型的系数和截距
    slope = model.coef_[0]
    intercept = model.intercept_


    print("线性回归方程: y = {}x + {}".format(slope[0], intercept[0]))
    xishu.append([slope[0], intercept[0]])

## ======================================  计算销量上届  ======================================
inputd_up = []
for i in range(len(nameClass)):
    name = nameClass[i]
    tmp = df22[df22['单品名称'] == name]
    tmp = tmp.groupby(['销售日期'])['销量(千克)'].sum().reset_index()
    inputd_up.append(tmp['销量(千克)'].quantile(0.975))
# ======================================  计算打折率 和 打折量  ======================================

df22 = nameClassDF.merge(df2, on='单品名称')[['单品名称', '损耗率(%)', '销售日期', '销量(千克)', '销售单价(元/千克)', '批发价格(元/千克)', '是否打折销售']]
disrate = []
for i in range(len(nameClass)):
    name = nameClass[i]
    tmp = df22[df22['单品名称'] == name]
    # tmp = tmp.groupby(['销售日期', '是否打折销售'])['销售单价(元/千克)'].mean().reset_index()
    t1 = tmp[tmp['是否打折销售'] == '是'].shape[0]
    disrate.append(t1 / tmp.shape[0])

dis = []
for i in range(len(nameClass)):
    name = nameClass[i]
    tmp = df22[df22['单品名称'] == name]
    tmp = tmp.groupby(['是否打折销售'])['销售单价(元/千克)'].mean().reset_index()
    dis.append(min(tmp['销售单价(元/千克)'][0], tmp['销售单价(元/千克)'][1]) / max(tmp['销售单价(元/千克)'][0], tmp['销售单价(元/千克)'][1]))

## ======================================  计算价格上届  ======================================
price_up = []
for name in nameClass:
    tmp = df22[df22['单品名称'] == name]

    price_up.append(tmp['销售单价(元/千克)'].max())

## ======================================  粒子群优化算法求解得到答案  ======================================
for id in range(len(nameClass)):
    # id = 0

    class Particle:
        def __init__(self, dim, lower_bound, upper_bound):
            self.position = np.random.uniform(lower_bound, upper_bound, dim)
            self.velocity = np.random.uniform(-0.1, 0.1, dim)
            self.best_position = np.copy(self.position)
            self.best_score = -float('inf')

    def objective_function(x, cost, p0, p1, disrate, dis):
        price, inputd = x
        q = p0 * price + p1
        gross_profit = price * dis * q * disrate + price * q * (1 - disrate)
        total_cost = inputd * cost
        net_profit = gross_profit - total_cost
        return net_profit
```

```
    # 增加惩罚项 确保 inputd 不少于需求
    def objective_function_with_constraint(x, cost, p0, p1, disrate, dis):
        price, inputd = x
        q = p0 * price + p1
        constraint = inputd - 1.05 * q

        # 满足约束条件
        if constraint >= 0:
            gross_profit = price * dis * q * disrate + price * q * (1 - disrate)
            total_cost = inputd * cost
            net_profit = gross_profit - total_cost
        else:
            # 违反约束条件，施加一个大的负惩罚
            net_profit = -float('inf')

        return net_profit

    def update_velocity(particle, global_best_position, w=0.5, c1=1.5, c2=1.5):
        inertia = w * particle.velocity
        cognitive = c1 * np.random.random() * (particle.best_position - particle.position)
        social = c2 * np.random.random() * (global_best_position - particle.position)
        new_velocity = inertia + cognitive + social
        return new_velocity

    # 无用的东西
    def check(x):
        return x * (1.05)

    # 参数
    costid = cost[id]
    price = price_up[id]
    p0 = xishu[id][0]
    p1 = xishu[id][1]
    disrateid = disrate[id]
    disid = dis[id]
    inputd_upid = inputd_up[id]
    n_particles = 30
    n_iterations = 1000

    # 粒子群初始化
    particles = [Particle(dim=2, lower_bound=np.array([costid, 2.5]), upper_bound=np.array([price, inputd_upid])) for _
in range(n_particles)]

    # 全局最优
    global_best_score = -float('inf')
    global_best_position = None


    # 粒子群算法主循环
    for iteration in range(n_iterations):
        for particle in particles:
            score = objective_function_with_constraint(particle.position, costid, p0, p1, disrateid, disid)

            # 更新个体最优
            if score > particle.best_score:
                particle.best_score = score
                particle.best_position = np.copy(particle.position)

            # 更新全局最优
            if score > global_best_score:
                global_best_score = score
                global_best_position = np.copy(particle.position)

        # 更新速度和位置
        for particle in particles:
            particle.velocity = update_velocity(particle, global_best_position)

            # 添加一些随机扰动
            particle.velocity += np.random.uniform(-0.1, 0.1, particle.velocity.shape)

            # 更新位置
```

```
            particle.position += particle.velocity

            # 边界检查
            particle.position = np.clip(particle.position, [costid, 2.5], [price, inputd_upid])

        # 输出当前最优解
        # print(f"Iteration {iteration + 1}: Global best score = {global_best_score}, best position =
{global_best_position}")


    # 结果
    print(f"{nameClass[id]}: 最大利润 = {global_best_score}, 最佳定价 = {global_best_position[0]}, 最佳采购量 =
{global_best_position[1]}")




## ====================================  可视化  ====================================
id = 0



class Particle:
    def __init__(self, dim, lower_bound, upper_bound):
        self.position = np.random.uniform(lower_bound, upper_bound, dim)
        self.velocity = np.random.uniform(-0.1, 0.1, dim)
        self.best_position = np.copy(self.position)
        self.best_score = -float('inf')

def objective_function_with_constraint(x, cost, p0, p1, disrate, dis):
    price, inputd = x
    q = p0 * price + p1
    constraint = inputd - 1.05 * q

    if constraint >= 0:
        gross_profit = price * dis * q * disrate + price * q * (1 - disrate)
        total_cost = inputd * cost
        net_profit = gross_profit - total_cost
    else:
        net_profit = -float('inf')

    return net_profit

def update_velocity(particle, global_best_position, w=0.5, c1=1.5, c2=1.5):
    inertia = w * particle.velocity
    cognitive = c1 * np.random.random() * (particle.best_position - particle.position)
    social = c2 * np.random.random() * (global_best_position - particle.position)
    new_velocity = inertia + cognitive + social
    return new_velocity

def check(x):
    return x * (1.05)

particle_positions = []
global_best_positions = []

costid = cost[id]
p0 = xishu[id][0]
p1 = xishu[id][1]
disrateid = disrate[id]
disid = dis[id]
inputd_upid = inputd_up[id]
n_particles = 30
n_iterations = 1000

particles = [Particle(dim=2, lower_bound=np.array([costid, 2.5]), upper_bound=np.array([100, inputd_upid])) for _ in
range(n_particles)]
global_best_score = -float('inf')
global_best_position = None
for iteration in range(n_iterations):
    iter_particle_positions = []
```

```
    for particle in particles:
        score = objective_function_with_constraint(particle.position, costid, p0, p1, disrateid, disid)
        if score > particle.best_score:
            particle.best_score = score
            particle.best_position = np.copy(particle.position)
        if score > global_best_score:
            global_best_score = score
            global_best_position = np.copy(particle.position)
        iter_particle_positions.append(np.copy(particle.position))
    particle_positions.append(np.array(iter_particle_positions))
    global_best_positions.append(np.copy(global_best_position))

    for particle in particles:
        particle.velocity = update_velocity(particle, global_best_position)
        particle.velocity += np.random.uniform(-0.1, 0.1, particle.velocity.shape)
        particle.position += particle.velocity
        particle.position = np.clip(particle.position, [costid, 2.5], [100, inputd_upid])
print(f"{nameClass[id]}: 最大利润 = {global_best_score}, 最佳定价 = {global_best_position[0]}, 最佳采购量 =
{global_best_position[1]}")

# 可视化
particle_positions = np.array(particle_positions)
global_best_positions = np.array(global_best_positions)

plt.figure(figsize=(8, 8))

for i in range(len(particle_positions)):
    if i > 5 and i < 995:
        continue
    plt.scatter(particle_positions[i, :, 0], particle_positions[i, :, 1], label=f'Iteration {i+1}')

plt.scatter(global_best_positions[:, 0], global_best_positions[:, 1], c='red', label='Global Best')
plt.title('每次迭代的粒子位置')
plt.xlabel('价格')
plt.ylabel('进货量')
plt.legend(loc='upper right')
plt.show()

## ====================================  到处答案  ====================================
data = {
    '商品名称': ['西兰花', '芜湖青椒(1)', '净藕(1)', '云南生菜', '紫茄子(2)', '螺丝椒', '云南油麦菜', '小米椒(份)', '云南生菜
(份)', '上海青', '金针菇(盒)', '西峡花菇(1)', '枝江青梗散花', '菠菜', '螺丝椒(份)', '娃娃菜', '云南油麦菜(份)', '竹叶菜', '菜
心', '红薯尖', '奶白菜', '长线茄', '菠菜(份)', '青茄子(1)', '双孢菇(盒)', '小皱皮(份)', '莴菜', '白玉菇(袋)', '小青菜(1)', '
洪湖藕带', '木耳菜', '高瓜(1)', '青线椒(份)', '红椒(2)', '海鲜菇(包)', '姜蒜小米椒组合装(小份)', '蟹味菇与白玉菇双拼(盒)', '
七彩椒(2)', '圆茄子(2)', '高瓜(2)'],
    '最大利润': [79.94, 233.61, 101.35, 58.09, 165.16, 67.51, 67.35, 80.29, 143.73, 77.71, 499.68, 64.05, 55.45, 156.71,
88.62, 18.44, 127.87, 68.21, 18.20, 40.95, 43.39, 83.88, 44.45, 26.23, 43.25, 27.77, 40.78, 26.75, 38.50, 40.17, 102.96,
47.19, 28.37, 12.50, 144.48, 12.99, 30.87, 14.99, 1.84, 52.18],
    '最佳定价': [14.74, 18.0, 19.79, 11.63, 18.0, 21.48, 16.0, 8.55, 6.9, 25.8, 8.9, 27.6, 14.0, 27.6, 8.98, 6.52, 11.8,
11.56, 12.00, 10.92, 11.59, 21.6, 6.8, 12.77, 6.9, 6.81, 11.34, 9.9, 12.0, 32.57, 19.8, 31.6, 8.9,  19.32, 9.9, 4.9, 13.8,
33.6, 11.89, 29.6],
    '最佳采购量': [14.88, 19.72, 9.68, 9.57, 12.39, 5.69, 6.59, 14.99, 47.78, 3.79, 73.56, 5.18, 14.76, 9.71, 15.72, 11.77,
16.76, 8.61, 2.88, 5.83, 5.40, 5.89, 17.40, 3.02, 15.87, 7.58, 5.06, 5.40, 17.41, 3.17, 6.75, 2.51, 6.71, 6.71, 2.50, 21.93,
6.52, 3.67, 2.50, 3.88]
}

df = pd.DataFrame(data)
df.to_csv('../data/第三问/ans.csv', index=False, encoding='utf-8-sig')

## df = pd.DataFrame(data)  0-1 规划模型权责前 33 个商品  df = pd.DataFrame(data)
from pulp import LpMaximize, LpProblem, LpVariable
max_profits = [5, 7, 9, 6, 8, 10, 12, 15, 20, 17,
               11, 13, 16, 18, 22, 25, 19, 21, 24, 27,
               26, 28, 30, 31, 33, 35, 36, 40, 42, 45,
               46, 48, 50, 51, 52, 55, 56, 57, 58, 60]

max_profits = df['最大利润'].tolist()
prob = LpProblem(name="Maximize_Total_Profit", sense=LpMaximize)
x_vars = [LpVariable(name=f"x{i}", cat='Binary') for i in range(40)]
prob += sum(max_profits[i] * x_vars[i] for i in range(40)), "Total_Profit"
prob += sum(x_vars) >= 27, "Min_items"
```

```
prob += sum(x_vars) <= 33, "Max_items"
prob.solve()
selected_items = [i for i in range(40) if x_vars[i].varValue == 1]

print(f"Selected items: {selected_items}")
print(f"Total Profit: {prob.objective.value()}")
df.iloc[selected_items].to_csv('../data/第三问/final_ans.csv', index=False, encoding='utf-8-sig')
selected_items =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 20, 21, 22, 23, 24, 26, 29, 30, 31, 34,
39]
df.iloc[selected_items].to_csv('../data/第四问/final_ans.csv', index=False, encoding='utf-8-sig')
```

（7）聚类

```
from fbprophet import Prophet
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns

## ======================================  数据处理特征构造  ======================================
df2 = pd.read_csv('../data/附件 2.csv')
df2['销售日期'] = pd.to_datetime(df2['销售日期'])
df2['扫码销售时间'] = pd.to_datetime(df2['扫码销售时间'])
df2.head()
df2.to_csv('../data/附件 2.csv', index=False, encoding='utf-8-sig')
df2 = pd.read_csv('../data/附件 2.csv')
df2['销售日期'] = pd.to_datetime(df2['销售日期'])
df2['扫码销售时间'] = pd.to_datetime(df2['扫码销售时间'])
df41 = pd.read_excel('../data/附件 4.xlsx', sheet_name=0)
df42 = pd.read_excel('../data/附件 4.xlsx', sheet_name=1)
df42 = df42.drop(columns=['单品编码'])
df2 = df2.merge(df42, on='单品名称')
df2.head()
df2['损耗率(%)'].fillna(df2['损耗率(%)'].mean(), inplace=True)
df2.head()
df2['hour'] = df2['扫码销售时间'].dt.hour
df2['minute'] = df2['扫码销售时间'].dt.minute
df2['second'] = df2['扫码销售时间'].dt.second
df2['millisecond'] = df2['扫码销售时间'].dt.microsecond // 1000
df2['连续型时间'] = df2['hour'] * 3600 + df2['minute'] * 60 + df2['second'] + (df2['millisecond'] / 1000)
df41.drop(columns=['小分类编码'], inplace=True)
df41.rename(columns={'小分类名称': '分类名称', '平均损耗率(%)_小分类编码_不同值': '类损耗值'}, inplace=True)
df2 = df2.merge(df41, on=['分类名称'])
df3 = pd.read_excel('../data/附件 3.xlsx')
df3.head()
df3.rename(columns={'日期': '销售日期'}, inplace=True)
df3['销售日期'] = pd.to_datetime(df3['销售日期'])
df2 = pd.merge(df2, df3, on=['销售日期', '单品编码'], how='left')
df2['批发价格(元/千克)'].fillna(df2['批发价格(元/千克)'].mean(), inplace=True)
df2.drop(columns=['Unnamed: 0', '扫码销售时间', '单品编码', '分类编码', 'hour', 'minute', 'second', 'millisecond'],
inplace=True)
df2.columns
one_hot = pd.get_dummies(df2['分类名称'], prefix='分类名称')
df2 = pd.concat([df2, one_hot], axis=1)
df2.drop('分类名称', axis=1, inplace=True)
one_hot = pd.get_dummies(df2['销售类型'], prefix='销售类型')
df2 = pd.concat([df2, one_hot], axis=1)
df2.drop('销售类型', axis=1, inplace=True)
one_hot = pd.get_dummies(df2['是否打折销售'], prefix='是否打折销售')
df2 = pd.concat([df2, one_hot], axis=1)
df2.drop('是否打折销售', axis=1, inplace=True)
one_hot = pd.get_dummies(df2['季节'], prefix='季节')
df2 = pd.concat([df2, one_hot], axis=1)
df2.drop('季节', axis=1, inplace=True)
df2.drop(columns=['月份编号'], inplace=True)
base_date = pd.Timestamp('2020-07-01')
df2['天数差'] = (df2['销售日期'] - base_date).dt.days
df22 = df2.copy()
df2.drop('销售日期', axis=1, inplace=True)
```

```
df2.head()
df2.drop('月份', axis=1, inplace=True)
df2.head()
df2.drop('单品名称', axis=1, inplace=True)
df2.head()
df2.columns

# 现在有一个处理好的 df2: 描述如下:
#
# Index(['销量(千克)', '销售单价(元/千克)', '损耗率(%)', '连续型时间', '类损耗值', '批发价格(元/千克)',
#        '分类名称_水生根茎类', '分类名称_花叶类', '分类名称_花菜类', '分类名称_茄类', '分类名称_辣椒类', '分类名称_食用菌',
#        '销售类型_退货', '销售类型_销售', '是否打折销售_否', '是否打折销售_是', '季节_冬季', '季节_夏季', '季节_春季',
#        '季节_秋季', '天数差'],
#       dtype='object')
#
# 其中 ['销量(千克)', '销售单价(元/千克)', '损耗率(%)', '连续型时间', '类损耗值', '批发价格(元/千克)'] 是连续型数据
#      ['分类名称_水生根茎类', '分类名称_花叶类', '分类名称_花菜类', '分类名称_茄类', '分类名称_辣椒类', '分类名称_食用菌'] 是
独热编码处理后的分类名称数据
#      ['销售类型_退货', '销售类型_销售'] 是独热编码处理后的销售形式数据
#      ['是否打折销售_否', '是否打折销售_是'] 是独热编码处理后的是否打折数据
#      ['季节_冬季', '季节_夏季', '季节_春季','季节_秋季'] 是独热编码处理后的季节数据
#      ['天数差'] 是日期被处理成了顺序数据
#
# 现在请你聚类分析


## ======================================  Kmeans 聚类  ======================================

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
# 对连续型数据进行标准化
scaler = StandardScaler()
continuous_features = ['销量(千克)', '销售单价(元/千克)', '损耗率(%)', '连续型时间', '类损耗值', '批发价格(元/千克)', '天数
差']
df2[continuous_features] = scaler.fit_transform(df2[continuous_features])
n_clusters = 24  # 假设我们想要将数据分为 3 个类簇，这个数字可能需要通过多次尝试或使用肘部法则来确定
kmeans = KMeans(n_clusters=n_clusters, random_state=0)
df2['cluster'] = kmeans.fit_predict(df2)
print(df2['cluster'].value_counts())



df2.to_csv('../data/聚类/data.csv', index=False, encoding='utf-8-sig')

## ======================================  PCA 降维 然后绘图  ======================================
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
df2_pca = pca.fit_transform(df2.drop('cluster', axis=1))
df2_pca = pd.DataFrame(df2_pca, columns=['PC1', 'PC2'])
df2_pca['cluster'] = df2['cluster']
plt.figure(figsize=(10, 8))
for cluster in df2_pca['cluster'].unique():
    mask = df2_pca['cluster'] == cluster
    plt.scatter(df2_pca[mask]['PC1'], df2_pca[mask]['PC2'], label=f'Cluster {cluster}', s=0.1)
plt.legend()
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Cluster Visualization in 2D Space')
plt.show()
```

附件 3：C++ on Visual Studio Code
编译环境 MinGW g++ version 8.1.0

```
#include <iostream>
#include <vector>
#include <algorithm>

const int n = 40, m = 250;
const double w[] = {0, 79.94, 101.35, 233.61, 58.09, 165.16, 67.51, 67.35, 80.29, 143.73, 77.71, 499.68, 64.05, 55.45,
156.71, 88.62, 18.44, 127.87, 68.21, 18.2, 40.95, 43.39, 83.88, 44.45, 26.23, 43.25, 27.77, 40.78, 26.75, 38.5, 40.17,
102.96, 47.19, 28.37, 12.5, 144.48, 12.99, 30.87, 14.99, 1.84, 52.18};
```

```
const int v[] = {0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 12, 11, 10, 9, 8, 10, 9, 8, 8, 7, 8, 9, 10, 11, 5, 4, 11, 10, 9, 8,
8, 7, 6, 11, 10, 9, 8, 8, 7, 6, 6};

struct ItemInfo {
    double value;
    std::vector<int> items;
};

int main () {
    std::vector<ItemInfo> dp(m + 1, {0.0, {}});

    for (int i = 1; i <= n; i++) {
        for (int j = m; j >= v[i]; j--) {
            if (dp[j - v[i]].value + w[i] > dp[j].value) {
                dp[j].value = dp[j - v[i]].value + w[i];
                dp[j].items = dp[j - v[i]].items;
                dp[j].items.push_back(i);
            }
        }
    }

    std::cout << "Max Value: " << dp[m].value << std::endl;
    std::cout << "Selected items: ";
    for (int item : dp[m].items) {
        std::cout << item - 1 << " ";
    }
    std::cout << std::endl;
    std::cout << (dp[m].items).size() << std::endl;
    return 0;
}
```

附录 4：正文补充结果表

（1）问题二未来一周（7.1-7.7）各品类的销售量预测表

7.1-7.7 水生根茎类销售量预测表

	预测值	下界	上界
7.1	31.51403422	22.43397765	40.3940729
7.2	27.21930854	18.50555806	36.34357316
7.3	15.95047518	7.625425143	25.19527342
7.4	15.28787515	6.644053331	23.93793046
7.5	14.62375048	5.938928568	24.20963441
7.6	16.77276953	7.271027793	26.05307241
7.7	23.77090806	14.76173934	32.9966772

7.1-7.7 花叶类销售量预测表

	预测值	下界	上界
7.1	215.8944203	190.2909671	239.7783828
7.2	202.7954785	178.3610395	227.6479174
7.3	154.882718	130.21414	179.1617402
7.4	148.6899247	124.0471077	174.5737885
7.5	151.7064607	126.3925275	177.8489863
7.6	149.0039038	125.4388636	173.3755064
7.7	164.6969315	139.2140604	187.7229009

7.1-7.7 水生根茎类销售量预测表

	预测值	下界	上界

7.1	31.51403422	22.43397765	40.3940729
7.2	27.21930854	18.50555806	36.34357316
7.3	15.95047518	7.625425143	25.19527342
7.4	15.28787515	6.644053331	23.93793046
7.5	14.62375048	5.938928568	24.20963441
7.6	16.77276953	7.271027793	26.05307241
7.7	23.77090806	14.76173934	32.9966772

7.1-7.7 花菜类销售量预测表

	预测值	下界	上界
7.1	37.51996125	30.30919796	44.57249602
7.2	38.05579645	30.63138576	45.35663574
7.3	26.15942873	18.67744689	33.47110952
7.4	25.23514472	17.65289488	32.60756178
7.5	26.2128648	18.95797763	33.24584119
7.6	26.77522264	19.66445325	34.00173148
7.7	32.19422171	24.76201285	39.94650394

7.1-7.7 茄类销售量预测表

	预测值	下界	上界
7.1	27.12031199	22.89607667	31.22346231
7.2	26.75332744	22.65803916	30.59074104
7.3	20.58611952	16.18319156	24.55115788
7.4	19.06649841	15.25878407	22.82836119
7.5	18.85687575	14.45875963	22.94482734
7.6	19.40170619	15.37307842	23.58607376
7.7	21.27189833	17.11496355	24.75892712

7.1-7.7 辣椒类销售量预测表

	预测值	下界	上界
7.1	108.8191043	90.77492386	127.2708361
7.2	100.6433493	81.07704362	119.5283698
7.3	75.16628216	55.99023839	94.79606952
7.4	70.96316513	53.52892679	91.75215313
7.5	71.44440193	52.8309617	89.81699525
7.6	74.44604891	55.15261791	93.77815475
7.7	87.79404014	68.80414482	105.1971331

7.1-7.7 食用菌销售量预测表

	预测值	下界	上界
7.1	94.16997113	77.38906603	110.1016467
7.2	88.09515501	71.50501829	105.7076963
7.3	63.83835419	47.94329531	79.81466835
7.4	60.50086686	42.98420196	77.86696412
7.5	66.96520527	48.69912262	84.14040538
7.6	19.40170619	15.37307842	23.58607376
7.7	21.27189833	17.11496355	24.75892712

（2）问题二的模型利润率及利润结果

7.1-7.7 各品类利润率表

	7月1日	7月2日	7月3日	7月4日	7月5日	7月6日	7月7日
水生根茎类	0.0812	0.2491	1.3971	1.6835	1.9433	1.5996	0.5621
花叶类	0.2307	0.3983	1.7151	1.9732	1.8195	1.8553	1.284
花菜类	0.3515	0.337	1.462	1.6427	1.4125	1.2946	0.7197
茄类	0.5123	0.5242	1.5641	1.8234	2.0321	1.7499	1.3373
辣椒类	0.24	0.2968	3.7641	4.5716	4.7514	3.9342	1.2456
食用菌	1.0317	1.1189	1.6596	1.8305	1.6202	1.5597	1.1947

7.1-7.7 各品类利润表

	7月1日	7月2日	7月3日	7月4日	7月5日	7月6日	7月7日
水生根茎类	17.7577	62.4418	367.9583	444.1855	513.3021	421.8449	145.7491
花叶类	92.569	215.8427	1180	1370	1260	1290	97.8
花菜类	411.5482	394.1539	1750	1970	1690	1550	855
茄类	60.3688	61.7936	186.4474	217.5357	242.5556	208.725	159.2715
辣椒类	505.1816	634.1978	8520	10400	10800	8900	2790
食用菌	1280	1390	2070	2280	2020	1940	1480

（3）问题三 TOPSIS 模型评分排序表

TOPSIS 模型评分排序前 40 位的评分表

单品名称	净利润	损耗率(%)	TOPSIS 得分
西兰花	89951.68923	9.26	0.977995731
芜湖青椒(1)	61356.95698	5.7	0.682530047
净藕(1)	57448.83735	5.54	0.639285418
云南生菜	49860.65935	15.25	0.553727197
紫茄子(2)	39867.58729	6.07	0.444869459
螺丝椒	28180.63217	10.18	0.315192468
云南油麦菜	27872.28247	12.81	0.311115767
小米椒(份)	27344.59	9.43	0.306216389
云南生菜(份)	25538.54	9.43	0.286405237
上海青	23141.48718	14.43	0.258598687
金针菇(盒)	20861.43	0.45	0.239471295
西峡花菇(1)	20986.84182	10.8	0.236149196
枝江青梗散花	19033.85341	9.43	0.215452278
菠菜	19260.50415	18.51	0.214755593
螺丝椒(份)	17679.15	9.43	0.200795303
娃娃菜	15824.66	2.48	0.185090935
云南油麦菜(份)	15190.41	9.43	0.174046104
竹叶菜	14182.44351	13.62	0.161001575
菜心	12762.05122	13.7	0.145659209
红薯尖	12072.21196	8.42	0.141758744
奶白菜	11293.39952	15.68	0.128795702

长线茄	10230.69717	6.9	0.124164452
菠菜(份)	10131.73	9.43	0.120935658
青茄子(1)	8531.15191	5.01	0.10948939
双孢菇(盒)	7853.86	0.2	0.109279205
小皱皮(份)	8570.75	9.43	0.105194456
莴菜	9150.42491	18.52	0.104185886
白玉菇(袋)	7886.26	6.57	0.101618472
小青菜(1)	7922.00643	10.33	0.09787971
洪湖藕带	6923.58263	24.05	0.07738815
木耳菜	4946.0424	7.61	0.074238567
高瓜(1)	6593.09279	29.25	0.072700086
青线椒(份)	4843.27	9.43	0.070644035
红椒(2)	4814.08202	9.43	0.070401996
海鲜菇(包)	1236.62	0	0.068353213
姜蒜小米椒组合装(小份)	4404.69	9.43	0.067078199
蟹味菇与白玉菇双拼(盒)	958.65	0.84	0.065932085
七彩椒(2)	4208.46592	9.43	0.065535641
圆茄子(2)	3104.34315	6.71	0.062650402
高瓜(2)	2821.638	9.43	0.055809462

（4）问题三聚类指标 p 值

	1 方差分析	2 克鲁斯卡尔-沃利斯检验(Kruskal-Wal
1 指标1	0	0
2 指标2	0	0
3 指标3	0	0
4 指标4	0	0
5 指标5	0	0
6 指标6	0	0
7 指标7	0	0
8 指标8	0	0
9 指标9	0	0
10 指标10	0	0
11 指标11	0	0
12 指标12	0	0
13 指标13	1.1360e-28	1.1453e-28
14 指标14	1.1360e-28	1.1453e-28
15 指标15	0	0
16 指标16	0	0
17 指标17	0	0
18 指标18	0	0
19 指标19	0	0
20 指标20	0	0
21 指标21	0	0
22		

（5）问题四全部答案

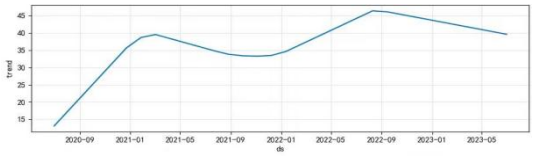
表 第四问增加容量限制的动态规划求解答案

单品名称	最大利润	最佳定价	最佳采购量
西兰花	79.94	14.74	14.88
芜湖青椒(1)	233.61	18	19.72

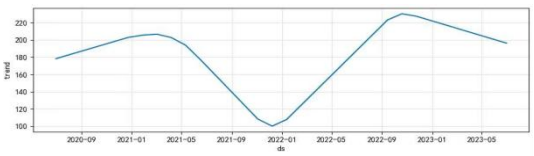
净藕(1)	101.35	19.79	9.68
云南生菜	58.09	11.63	9.57
紫茄子(2)	165.16	18	12.39
螺丝椒	67.51	21.48	5.69
云南油麦菜	67.35	16	6.59
小米椒(份)	80.29	8.55	14.99
云南生菜(份)	143.73	6.9	47.78
上海青	77.71	25.8	3.79
金针菇(盒)	499.68	8.9	73.56
西峡花菇(1)	64.05	27.6	5.18
枝江青梗散花	55.45	14	14.76
菠菜	156.71	27.6	9.71
螺丝椒(份)	88.62	8.98	15.72
云南油麦菜(份)	127.87	11.8	16.76
竹叶菜	68.21	11.56	8.61
红薯尖	40.95	10.92	5.83
奶白菜	43.39	11.59	5.4
长线茄	83.88	21.6	5.89
菠菜(份)	44.45	6.8	17.4
青茄子(1)	26.23	12.77	3.02
双孢菇(盒)	43.25	6.9	15.87
苋菜	40.78	11.34	5.06
洪湖藕带	40.17	32.57	3.17
木耳菜	102.96	19.8	6.75
高瓜(1)	47.19	31.6	2.51
海鲜菇(包)	144.48	9.9	2.5
高瓜(2)	52.18	29.6	3.88
最佳答案	2845.24		

附录 5：补充的图

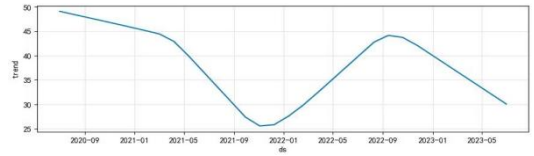
（1）问题一中全部品类总体趋势图



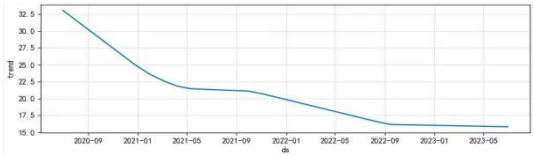
水生根茎类总体趋势折线图



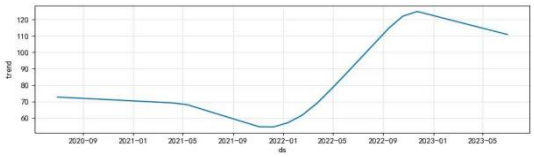
花叶类总体趋势折线图



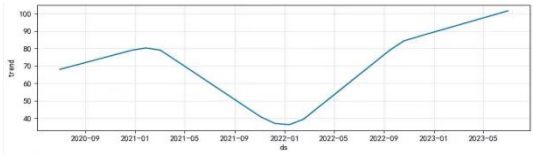
花菜类总体趋势折线图



茄类总体趋势折线图

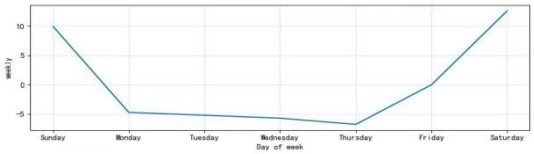


辣椒类总体趋势折线图

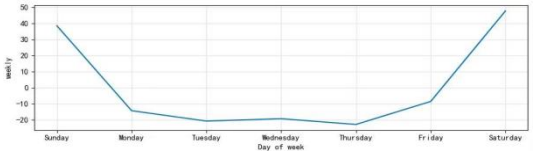


食用菌总体趋势折线图

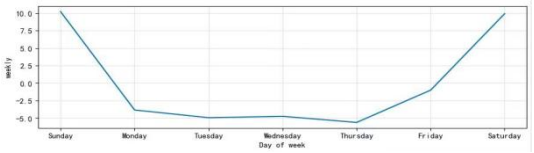
（2）问题一中全部品类周趋势图



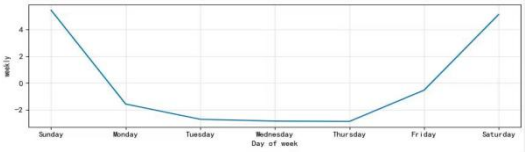
水生根茎类周趋势折线图



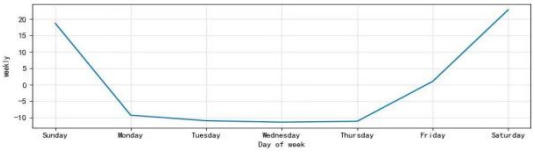
花叶类周趋势折线图



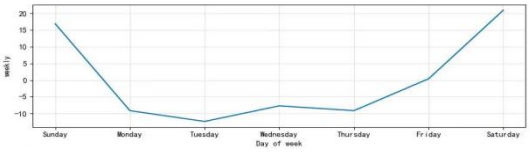
花菜类周趋势折线图



茄类周趋势折线图

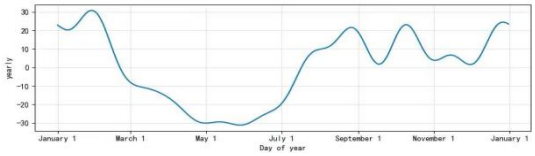


辣椒类周趋势折线图

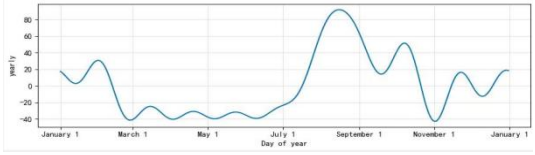


食用菌周趋势折线图

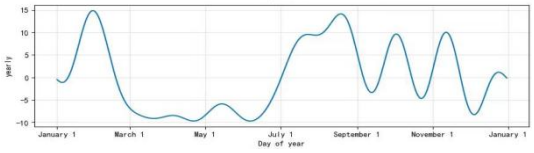
（3）问题一中全部品类年趋势



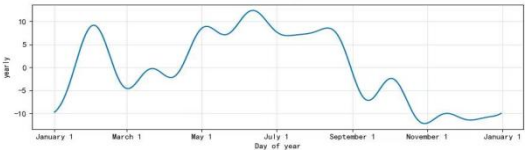
水生根茎类年趋势折线图



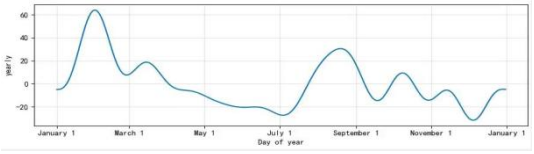
花叶类年趋势折线图



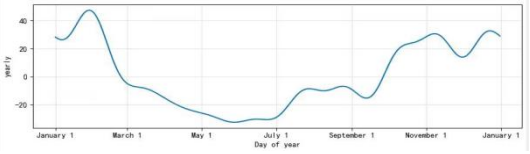
花菜类年趋势折线图



茄类年趋势折线图

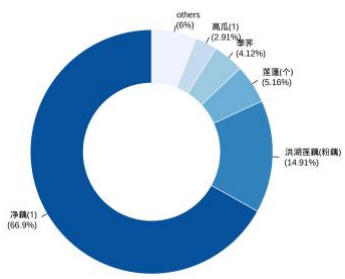


辣椒类年趋势折线图

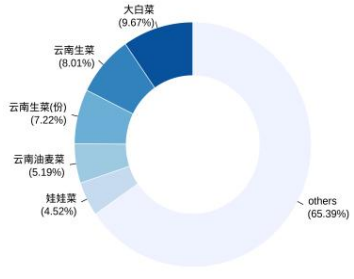


食用菌年趋势折线图

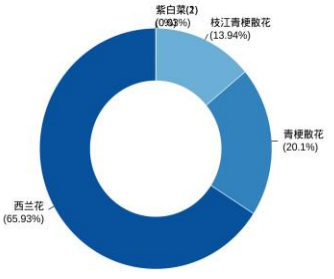
（4）问题一各单品总销量占比环图



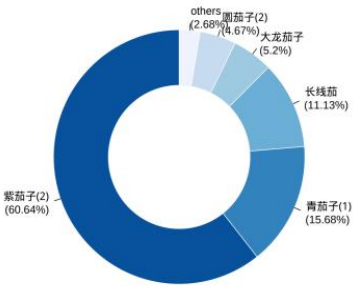
水生根茎类单品分布图



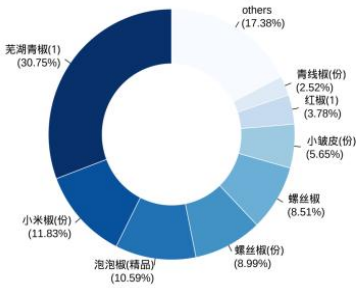
叶类单品分布图



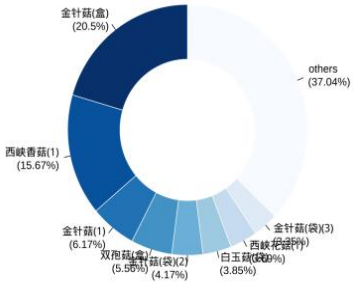
花菜类单品分布图



茄类单品分布图

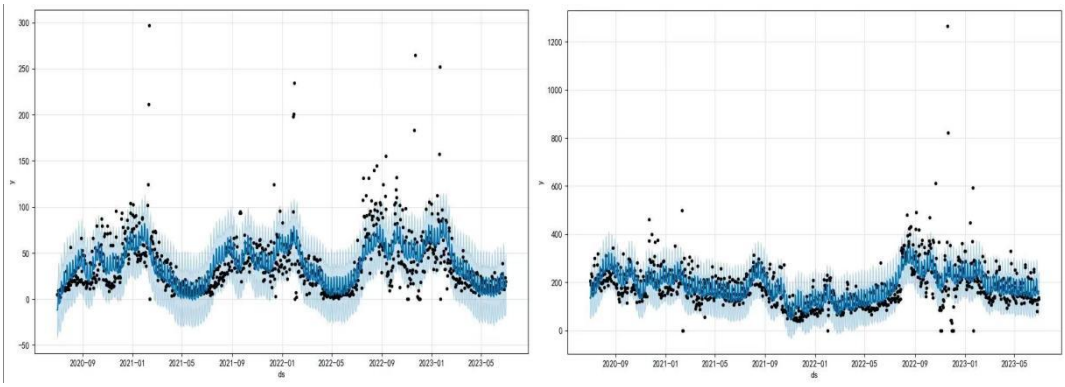


辣椒类单品分布图

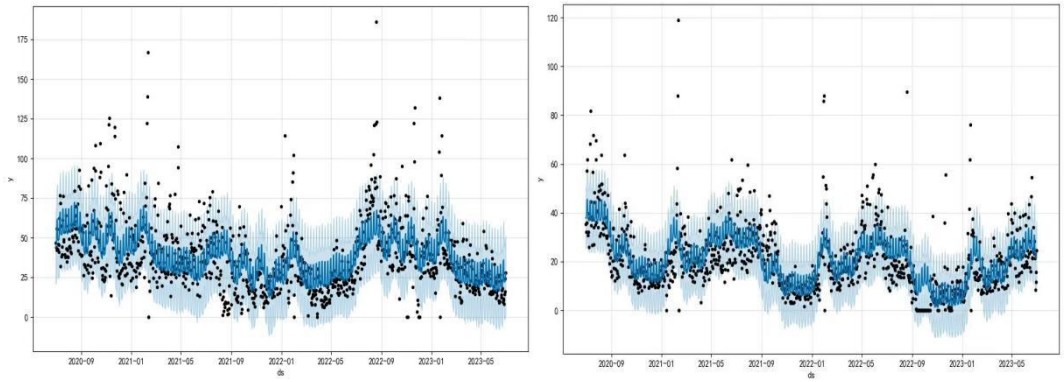


食用菌单品分布图

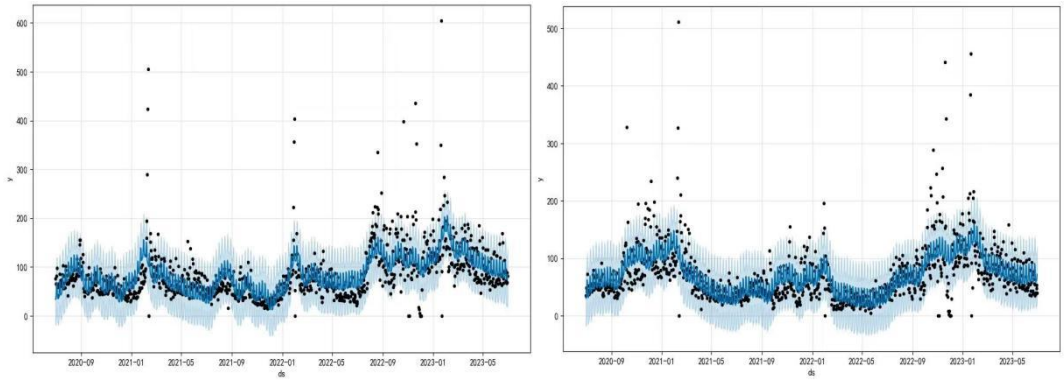
（5）问题一 6 种品类总体趋势及拟合情况图



水生根茎类　　花叶类

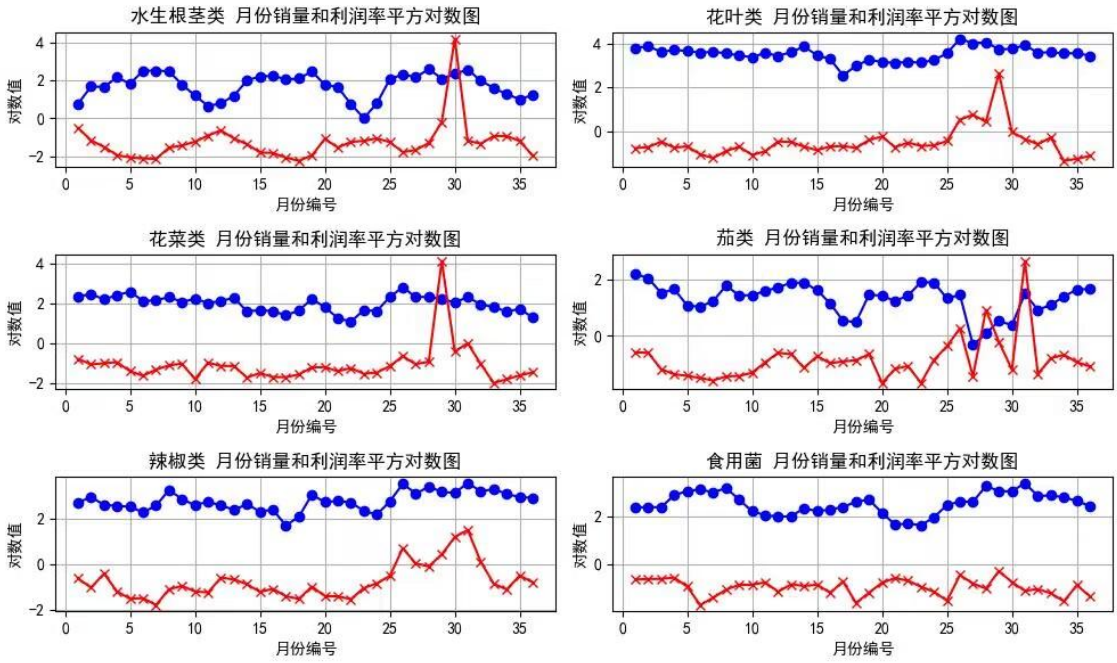


花菜类　　茄类

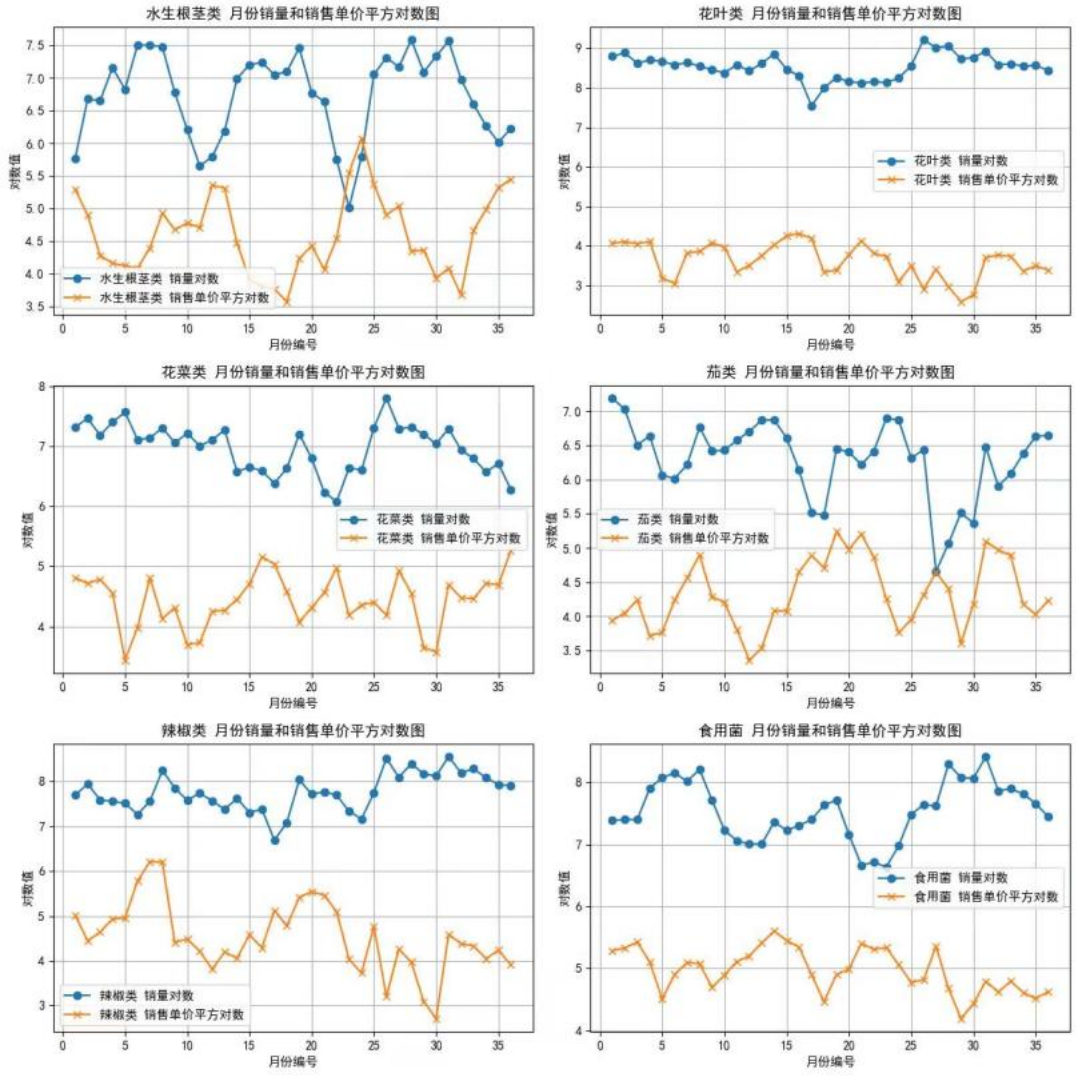


辣椒类　　食用菌

（6）问题二中各品类利润率-销量变化折线图



（7）问题二各品类销量与定价变化趋势折线图



（8）问题二 18 种具有强相关性单品组合的月度销售量折线图