

基于核密度估计与改进遗传算法的蔬菜销售决策模型

摘要

本文一方面通过**核密度估计**计算各品类蔬菜的销量与价格的概率密函数，为蔬菜补货与定价决策提供可靠评价指标，另一方面通过**非参数回归分析**对蔬菜销量与价格变换的供需关系进行宏观描述。结合以上两个模型对**遗传算法**进行改进，使蔬菜销售决策利益最大化。

针对问题一，本文先将清洗后的数据进行可视化分析。使用**Pearson相关性分析**指出蔬菜品类之间关系。对个别特征显著单品使用**核密度估计**的概率密度函数进行描述，所得回归方程刻画了其分布规律。

针对问题二，本文在分析销量与成本加成定价的相关性后，使用**改进XGBoost模型**对进货价格进行时间序列预测。进一步以销量与价格的分布特征为依据，构造各品类销量与价格的**含参数指数函数**。通过**非参数回归**推导商超利益模型，进而对其最优化求得最优策略。

针对问题三，本文沿用问题二的商超利益模型，结合销量分布引入销量与价格此对二维变量的概率密度函数，**改进商超利益模型**，提升模型**信息反馈量与适用范围**。进而利用较可信的含概率利益模型定义**遗传算法**的进化规则，借助**基因突变与基因重组**等启发式特征，求得在制约因素下的最优策略。

针对问题四，本文在对不同蔬菜品类的销售量进行线性判断后，使用**典型相关分析**判断爬取的天气数据中的温度和降雨量与不同蔬菜品类销售量之间的相互关系。计算出的典型变量，可用来对未来的数据进行**特征选取**，以帮助商家进行补货和定价决策的优化，进一步完善蔬菜的**数据驱动模型**。

从整体上看，本文借助**核密度估计**提供的**概率密度函数**，结合**非参数回归**引入的**期望函数**，将其作用于改进策略后的**遗传算法**，利用启发式算法涵盖的**隐性特征**达到降低搜索空间的突出优点，最终整合为**空间复杂度低及可信度高的蔬菜销售决策模型**。

关键词：遗传算法 核密度估计 非参数回归 XGBoost时间序列预测

一、问题重述

1.1 问题背景

随着我国社会的不断发展，人民对于美好生活需要的日益增长，对于食品的要求也水涨船高。生鲜市场中，由于蔬菜类商品保鲜期较短且品相容易变差，对于未售出的商品，往往无法隔日售卖。如今，绝大多数商场都选择根据商品的历史销售和需求进行每日补货，以满足国家与人民对于优质菜品的需求。

同时，由于当今商场的蔬菜品类丰富、产地各异，且蔬菜的进货交易往往发生于凌晨，各商家必须在单品与进价未知的情况下做出当天各蔬菜品类的补货决策。目前主流的蔬菜定价方法为“成本加成定价”法，商场对有运损和品相较差的商品打折销售。商家往往需要通过一定的市场需求分析，调整补货和定价策略。已知蔬菜的销量与时间通常存在关联关系且供应品在 4 至 6 月较丰富，但由于商场的销售空间有限，销售组合的合理性尤为重要。

1.2 数据分析

本题一共给出了四个附件材料，此处对这些附件数据做初步解读。

附件 1：某商场经营销售的 6 个蔬菜品类的商品信息

附件 2：2020 年 7 月 1 日至 2023 年 6 月 30 日的销售流水明细

附件 3：2020 年 7 月 1 日至 2023 年 6 月 30 日的批发价格

附件 4：各单品近期的运输损耗率

1.3 问题要求

根据题意及附件信息，本文需建立数学模型解决以下问题：

1. 蔬菜类产品不同品类或单品中可能有一定关联性，通过分析蔬菜各类及单品的销量，找出可能存在的分布规律和相互关系。

2. 由于商场以品类为单位进行补货，分析各品类蔬菜的总销量和成本加成定价的关系，为商场制定各蔬菜品类未来一周使收益最大化的每日进货总量和定价策略。

3. 为商场进一步完善单品的补货计划，要求售卖商品数在 27 至 33 个，且各单品订购量需大于等于 2.5 千克。根据 2023 年 6 月 24-30 日的可售品类，给出 7 月 1 日使商场收益最大的单品补货量和定价策略。

4. 为了优化蔬菜商品的补货与定价策略，找出商场还需采集的相关数据，并说明数据的用途，给出相应意见和理由。

二、问题分析

2.1 问题一分析

问题一的核心目标在于针对不同品类或单品和它们的销售量数据，进行其分布规律和相互关系的分析。对于不同品类或单品的分布规律，首先对附件一和附件二进行数据预处理，在处理过程中我们发现不同单品的销售量数据有大量缺失值，因此我们选择部分有代表性的单品进行分布规律和相关性的分析。本文通过直方图和核密度估计图反映不同品类或单品代表之间的分布规律。在相关性分析中，我们先对不同品类或单品代表使用 Shapiro - Wilk 进行正态性检验，再采用 Pearson 相关性分析得出不同品类或单品代表之间的相关性矩阵，最后用热力图可视化成相关性矩阵。

2.2 问题二分析

问题二需要考虑未来一周的日补货总量，并且还要制定一个定价策略以满足最大利益。首先需要分析各蔬菜品类的销售总量与成本加成定价之间的关系。其次我们需要一条各品类单日销售总量关于成本加成定价的函数，以描述各品类单日销售总量与对应成本加成定价之间的关系。接着我们先使用 XGBoost 模型预测未来一周的蔬菜品类的补货成本，即供货价，以帮助商超在未知进货价格的情况下制定补货计划和定价策略。在收益模型的建立中需要考虑销售价格、成本、销量以及利润之间的关系，最后通过数学建模和优化方法，我们可以找到商超的最佳决策点，以实现最大的利润。

2.3 问题三分析

针对问题三，题目可以拆解成四个部分。首先我们需要对 2023 年 7 月 1 日的可售单品进行补货量预测，因此我们需要构建出可售单品的时间序列模型，并使用 XGBoost 模型进行训练，以预测 2023 年 7 月 1 日的 49 种单品进货价格。这一步骤有助于了解未来成本的估计。其次是构建可售单品改进收益模型以更好地反映出单品销售总量与成本加成定价之间的关系，同时需要考虑价格分布的离散性，因此引入了概率密度函数，用于对销售总量和成本加成定价的关系进行加权修正。接着需要构建一共启发式的单品补货量与定价策略模型：使用遗传算法来寻找最佳的单品补货量和定价策略。并且需要考虑单品种类数量、销售最小陈列质量、市场的商品需求以及商超利益等不确定因素。最终，我们将使用遗传算法模型得出单品的补货量和定价策略，以实现最大化商超的收益。

2.4 问题四分析

问题四要求采集一些相关数据以做出更好的补货和定价决策，在建立了蔬菜类统计预测模型后，本文通过不同蔬菜品类或单品的销售量以及销售单价进行可能存在的噪音判断，因此本文选择了影响性较大的一种噪音数据，即天气数据。首先先进行了国内 2020 的 7 月 1 日到 2023 年 6 月 30 日天气数据的爬取，并且选择不同蔬菜类作为基本单位，再进行分析天气数据中的温度和降雨量与销售量之间的关系。由于数据的线性连续性和具有多组变量且数据量较大，从而本文选择了典型相关分析并且结合第二问和第三位得出的定价策略和补货策略通过典型变量和数据之间的强解释性进行数据驱动模型的完善，从而得到更好的补货和定价策略。

三、模型假设

1. 所有附件所给的数据真实且合理。
2. 附件二中的所有情况总和趋近于供需平衡。
3. 可以认为期望的商超销售利益最大值所对应的补货量与定价策略即为问题中所指的商超收益最大情况下的补货量与定价策略。

4. 问题四中所指的市场蔬菜类商品需求为历史三年的市场供需关系所决定的量。
5. 供应商的补货价格与商超补货数量无强关联。
6. 消费者的消费行为不受商超的成本价格波动直接影响。

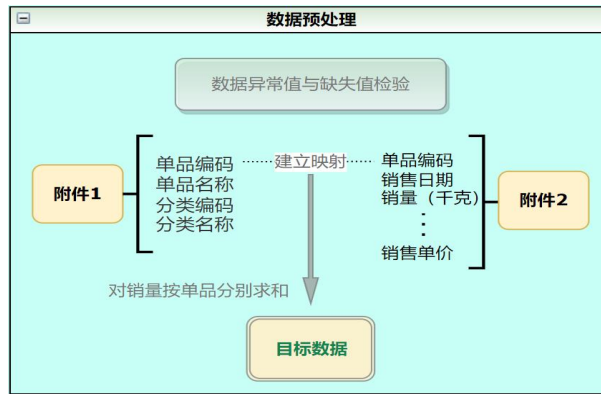
四、符号说明

符号	含义说明	单位
λ	随机事件的平均发生率	单位时间或单位面积
K	事件发生次数	次
n	样本大小	个
$x(i)$	数据集中的第 i 个值	千克
ρ	Spearman 相关系数	/
$\sigma_{R(x)}$	等级变量的协方差	/
$\sigma_{R(y)}$	等级变量的标准差	/
s	销售总量	千克
p	成本加成定价	元/千克
U_i, V_i	变量集合	/
ρ_i^*	典型相关系数	/
n_2	商超补货成本	元/千克

五、模型建立与求解

5.1 数据预处理

对于数据预处理，本文的大体思路如图 1：



(图 1 数据预处理)

5.1.1 数据异常值检验

通过 scipy 对附件数据进行异常值检测，并未发现存在空值等异常数据。对于后续分析各单品日销售情况数据，检测并剔除空值。最后通过 DBSCAN 聚类后剔除离群值。

5.1.2 合并数据

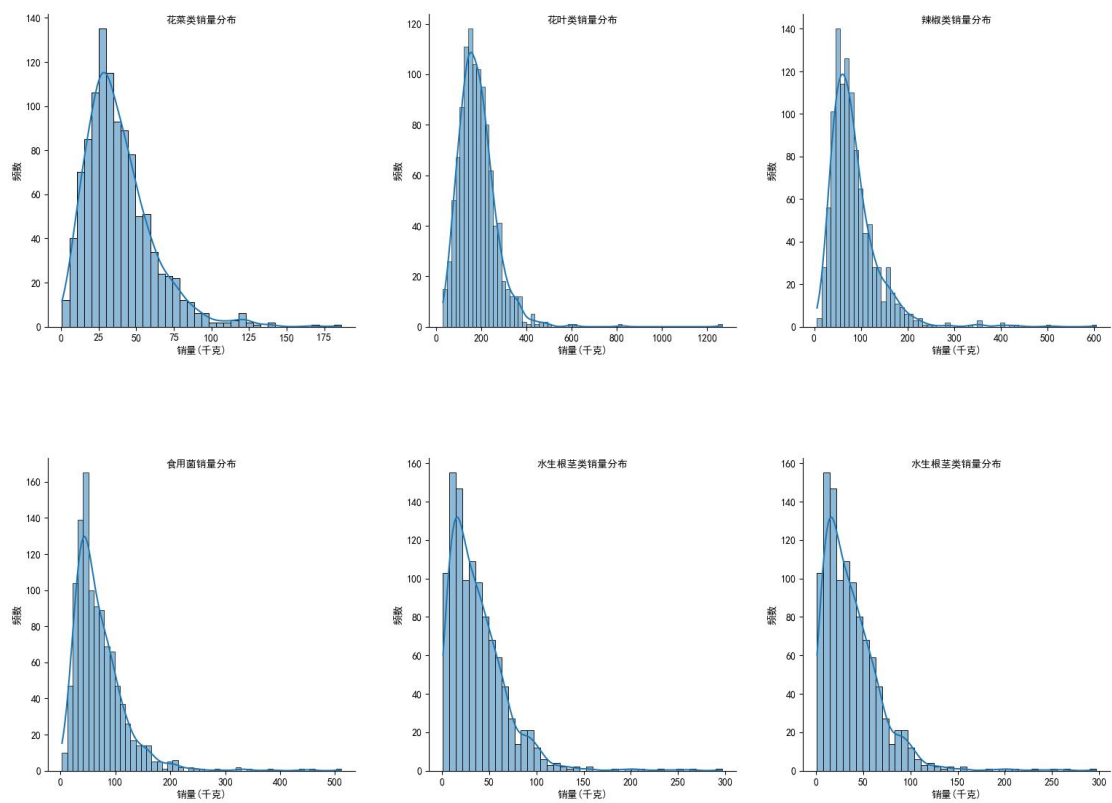
由附件一和附件二可知，蔬菜单品通过单品编码一一对应，因此将蔬菜品类的商品信息和销售流水明细数据通过单品编码映射的方式进行合并，并且将每一天的蔬菜品类和单品销售量进行汇总。

5.2 问题一的模型建立与求解

5.2.1 可视化蔬菜品类的分布规律

针对不同蔬菜品类，本文基于市场分布分析理论，结合了各品类在 2020-2023 三年的销售量，进行了可视化分析，以得到它们的分布规律。如图（2）所示，本文首先通过直方图的形式反映了六大蔬菜品类在时间维度上的分布情况，从这些直方图可知得出结论：六个蔬菜品类的销售量分布规律图都呈现左偏态并且具有单峰极大值，其中花叶类在所有品类中的销售量情况最好，日销量集中在 200 千克左右。食用菌和水生根茎的销售情况次之，销售量最不理想的是茄类，通过

观察可知其日销售量在 10-30 千克左右。这表明市场对不同品类的蔬菜的需求存在较为可观的差异。



(图 2 各蔬菜品类销售量分布图)

5.2.2 核密度估计模型

本文分析每日时间内各单品的销售量，由于其数据离散且相关性弱（将于后续进一步论证），每个销量的发生频数小、概率低，与核密度估计的分布规律契合。在不同单品的分布性规律的分析中，由于部分单品数据较少且过于离散，出于保证数据可靠性、准确性的目的，本文筛选出销售量天数最多的十种单品蔬菜，如表(1)所示

表 1 蔬菜单品销售天数前十

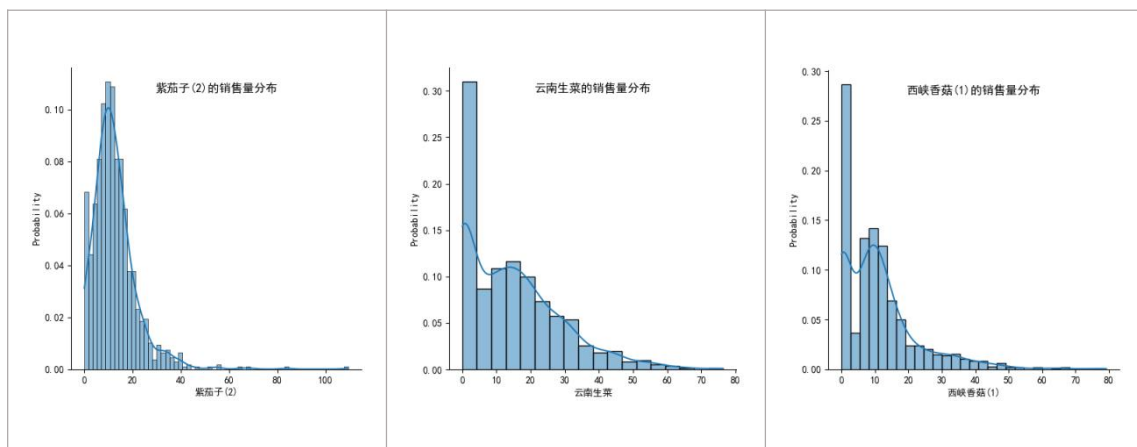
蔬菜单品（前十销量）	总销量天数
上海青	919
云南生菜	810
净藕(1)	1057

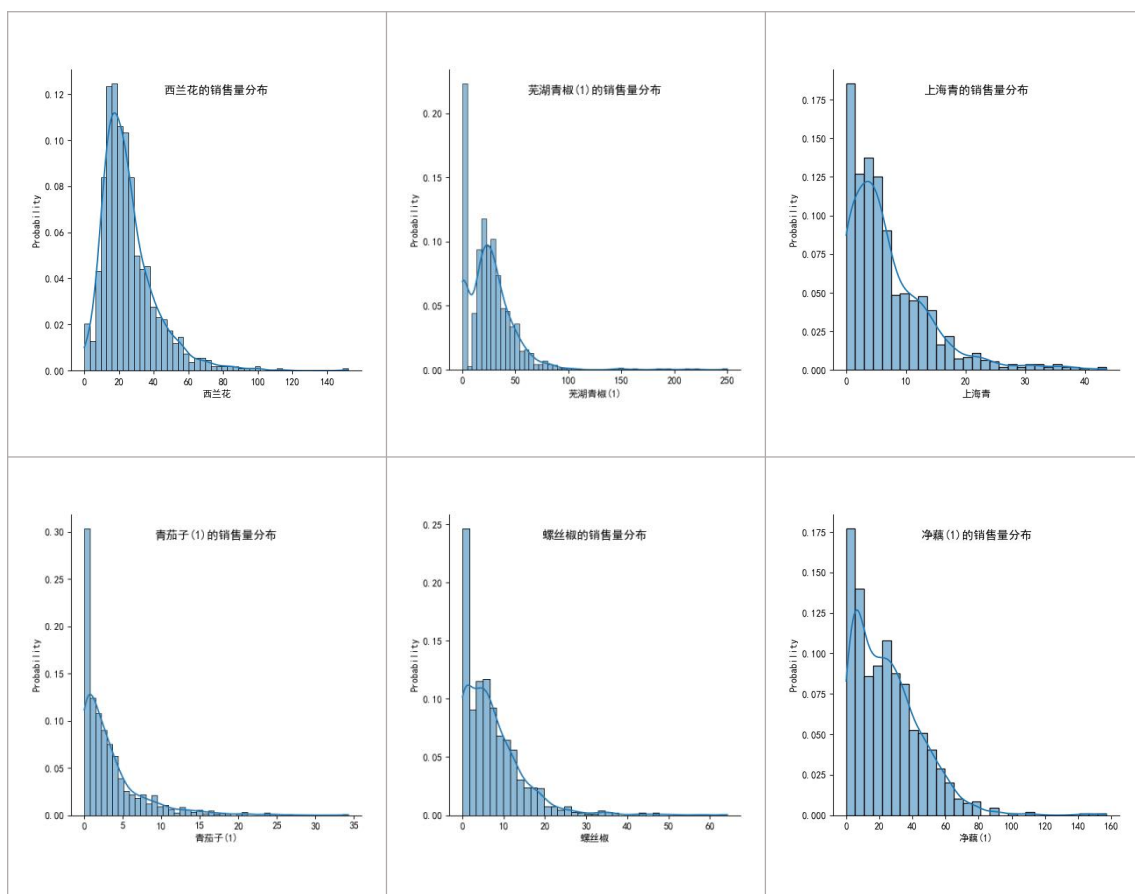
紫茄子(2)	1022
芜湖青椒(1)	858
螺丝椒	858
西兰花	1076
西峡香菇(1)	821
青茄子(1)	845
黄白菜(2)	904

核密度估计 (kernel density estimation) 是用一种非参数统计方法, 主要目标是根据观测到的数据点来估计数据的概率密度分布, 常用于估计或预测概率偏小事件的发生概率的情况, 符合本题要求的背景, 其具体公式如①所示:

$$\textcircled{1} \quad \begin{cases} p(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{s_i - x}{h}\right) \\ f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \end{cases}$$

由于本文利用核密度估计进行分布规律的分析, 所以先采用直方图反映出九个单品三年内每天各销售量的分布概率, 随即通过核密度估计拟合曲线进行拟合, 拟合状况如图 3 所示.



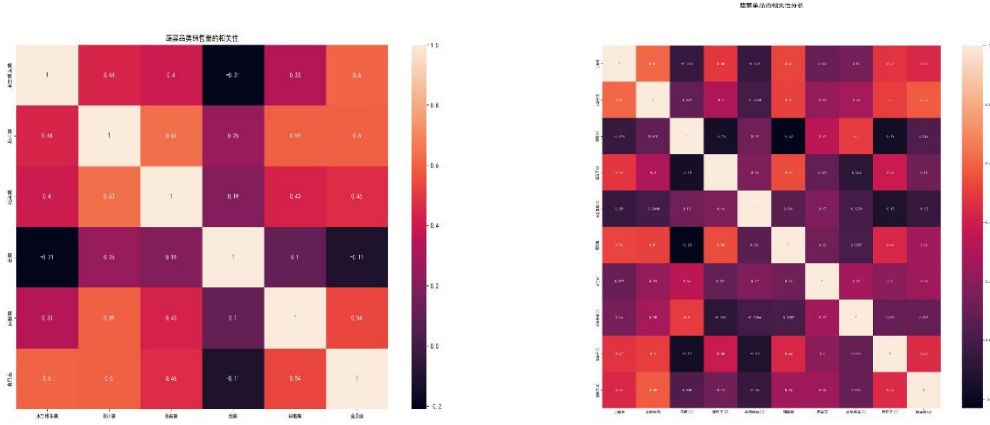


(图 3 核密度估计拟合曲线)

5.3.1.1 Pearson 相关性分析

在上述正态分布中可得知不同蔬菜品类和筛选出的单品销售量都不是正态分布的，因此本文选择使用 Spearman 相关系数进行相关性判断，Spearman 相关系数是一种非参数的方法，它基于数据的秩次（排名）而不是实际数值。因此，Spearman 相关系数对于非正态数据更为稳健。它的具体数学公式如下①所示，但使用等级变量来计算， $cov(R(X), R(Y))$ 为等级，通过：

$$① \quad r_s = \rho_{R(X), R(Y)} = \frac{cov(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}}$$



(图 4 拟合曲线图)

根据拟合情况易知，销售天数最多的十个蔬菜单品的销售量分布概率均随销售量的增加呈现递减的趋势，因此本文判断蔬菜类的大部分单品分布规律为销售量分布概率都随着销售量增加而递减。而基于核密度估计的拟合结果，我们也得知了其较好地反应了单品销售量的分布规律。

5.3 问题二的模型建立与求解

5.3.1 各蔬菜品类的销售总量与成本加成定价的关系

5.3.1.1 Shapiro - Wilk 正态性检验

Shapiro - Wilk 正态性检验是一种用于检验数据集是否符合正态分布的统计方法，其基本思想是通过计算观测值与正态分布的理论值之间的差异来评估数据集的正态性，公式如（4）所示其中 a 是和样本大小相关的系数。

$$W = \frac{(\sum_{i=1}^n a_i X_{(i)})^2}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

对不同蔬菜种类和单品进行 Shapiro - Wilk 正态性检验，得到检验统计量 W 和 p -value，如表 2、表 3 所示。由此表可知不同蔬菜品类和销售量前 10 的蔬

菜单品的销售量数据都不是正态分布。

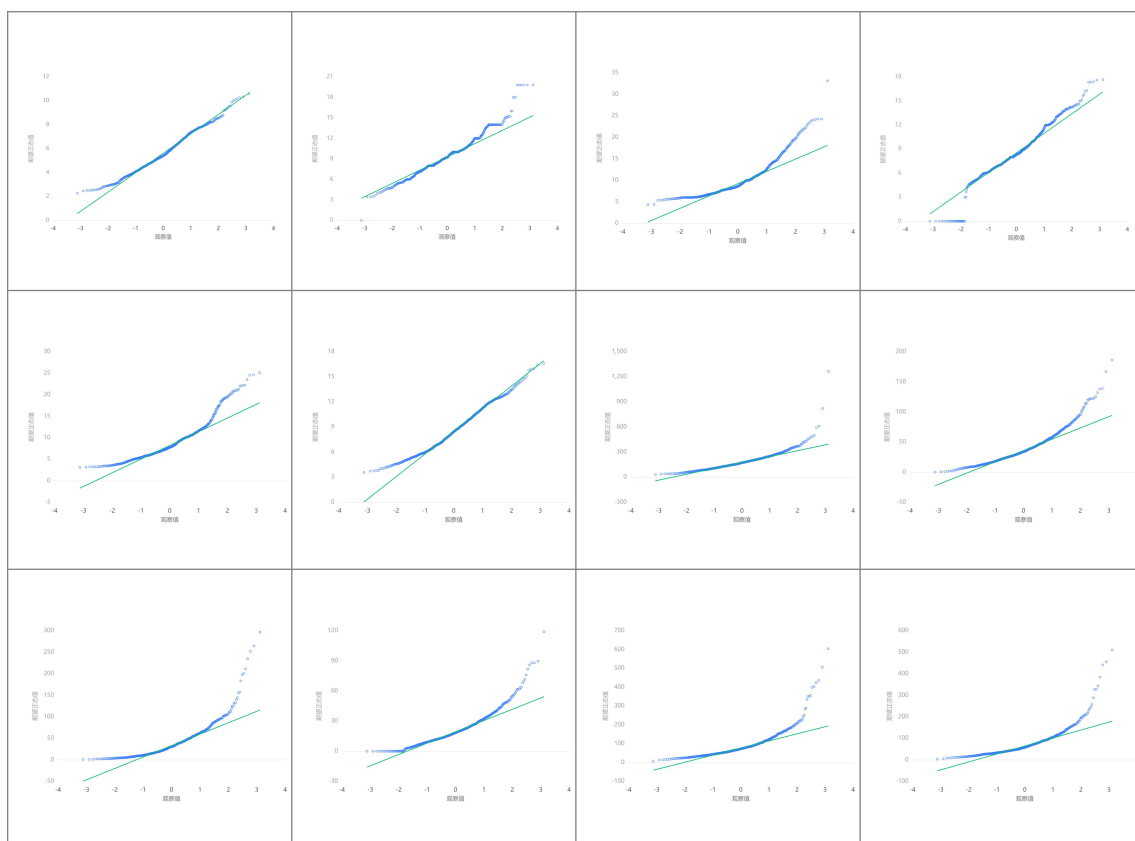
表 2 品类正态分布检验

蔬菜品类	W 值	P-value
水生根茎类	0.8124035596847534	1.0346200772483714e-33
花叶类	0.8525438904762268	1.2247579084720286e-30
花菜类	0.9028472900390625	1.148442194748215e-25
茄类	0.8869614005088806	5.2212742719895385e-27
辣椒类	0.7697542905807495	1.8980493980438353e-36
食用菌	0.7802593111991882	8.164946883120183e-36

表 3 单品正态分布检验

蔬菜单品（前十销量）	W 值	P-value
上海青	0.8317036032676697	2.6284814816978506e-32
云南生菜	0.8924208879470825	7.577795544437106e-27
净藕(1)	0.9010010957717896,	6.838687572655439e-26
紫茄子(2)	0.8355997204780579	5.232364568610394e-32
芜湖青椒(1)	0.7876903414726257	2.3715648750376612e-35
螺丝椒	0.8085261583328247	5.5784370115217866e-34
西兰花	0.8631017804145813	1.0121254624986789e-29
西峡香菇(1)	0.8233287334442139	6.239203691954103e-33
青茄子(1)	0.7432156801223755	5.962701529308602e-38
黄白菜(2)	0.6189067363739014	1.1770907100328463e-43

计算出它们的相关性系数后，本文使用 QQ 图去反应不同蔬菜品类或不同单品的销售量之间的相关性系数矩阵，如（图 5）所示：



(图 5 相关性系数矩阵)

问题二要求分析各蔬菜品类的销售总量与成本加成定价的关系，以下对蔬菜品类的销售总量与成本加成定价进行定义：

$$\begin{cases} \text{单品类销售总量} = \sum \text{单次销量} \\ \text{单品类成本加成定价} = \frac{\sum \text{单次销量} \times \text{单次销售单价}}{\sum \text{单次销量}} \end{cases}$$

附件二提供商超 2020 年 7 月 1 日至 2023 年 6 月 30 日销售流水明细数据，以单日单品类为统计单位，对花菜类、花叶类、辣椒类、茄类、食用菌和水生根茎类六类蔬菜品类的销售总量与成本加成定价的关系进行数据侧写。

5.3.1.2 各品类单日销售总量关于成本加成定价的函数

使用以自然常数 e 为底的指数函数描述 2020 年 7 月 1 日至 2023 年 6 月 30 日三年时间内每日各蔬菜类商品不同品类的销售总量与对应成本加成定价。

$$\textcircled{1} \quad s = A \times e^{-B \times p} + C$$

上述所构造的含三个参数的指数函数，其图像可过平面直角坐标系的第一象限，即包含蔬菜类商品不同品类的销售总量与成本加成定价关系所对应的散点坐标的定义域。而对于多项式函数而言，其几何解析曲线并不能较好地描述销售总量与成本加成定价之间的负相关关系。

对上述构造函数求导得：

$$\textcircled{2} \quad s^{(1)} = \frac{ds}{dp} = -A \times B \times e^{-B \times p}$$

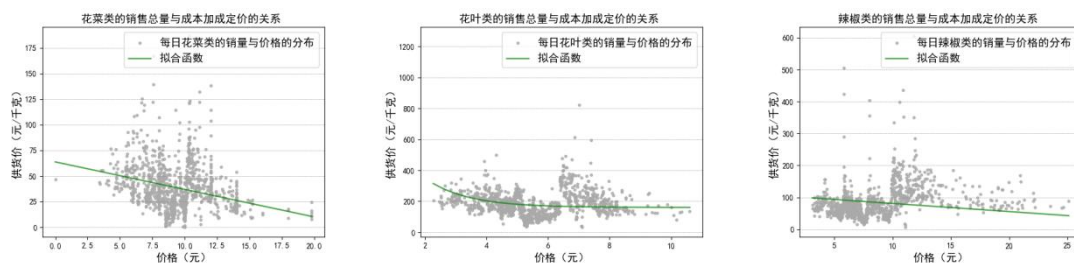
对于上式，当满足以下条件时：

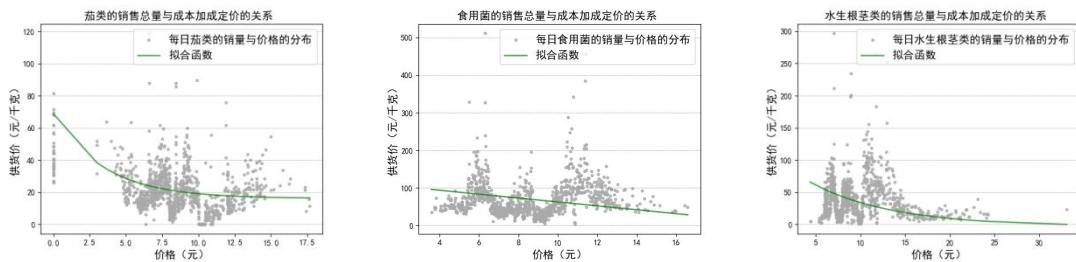
$$A \times B > 0$$

上述构造函数在第一象限单调递减，符合相关性分析中各类销售总量与成本加成定价均呈负相关关系。因此，使用上述函数对各品类单日销售总量的进行描述是相对的优解。

对各蔬菜类销售总量与成本加成定价的散点数据进行拟合，得到最优拟合函数，即可最优描述各品类单日销售总量与成本加成定价的以自然常数 e 为底的指数函数。下图描述了构造函数对销售总量与成本加成定价的关系的拟合效果。

此函数的数据为销售量与价格，影响二者之间大小的因素最主要为市场的供需关系，即可以认为此函数反映市场中宏观的单品类销量与价格的变换关系。





(图 6 各蔬菜品类的销售总量与成本加成定价的关系)

以下为六个品类蔬菜的构造函数中的 A, B, C 参数:

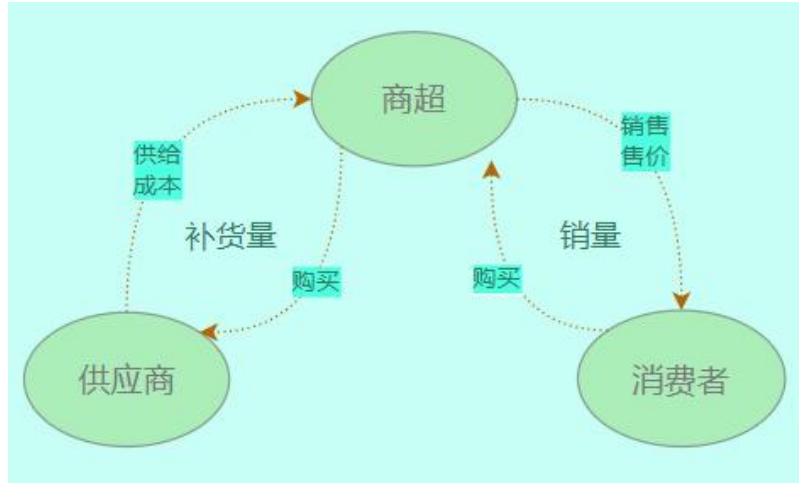
表 4 六个品类蔬菜的构造函数参数

A	B	C
829.7143823	0.740769825	160.4623945
49319.56406	5.44E-05	-49255.75816
111.2389213	0.110633835	-2.951092908
52.75622339	0.287256568	16.13595602
-317965.8811	-7.95E-06	318072.1666
13739.93153	0.000379206	-13625.16225

5.3.2 商超补货总量与定价策略

5.3.2.1 商超收益模型建立

根据题意,商超需向蔬菜类供应商购买商品,并指定合适的价格向消费者出售,示意图如下:



(图 7 商超收支示意图)

商超通过成本加成定价中的利润率获取利益，其利益关系为：

$$\text{利润} = (\text{价格} - \text{成本}) \times \text{销量}$$

由上图及题意可知，商家须在不确切知道具体单品和进货价格的情况下做出补货决策，消费者并不知悉商品的成本价格，即消费者所做出的决策与成本价格无直接关系。因此可以对各蔬菜品类未来一周(2023 年 7 月 1-7 日)的批发价格进行单一变量预测，以此为商超在未知进货价格的情况下做出补货决策与定价策略提供依据。

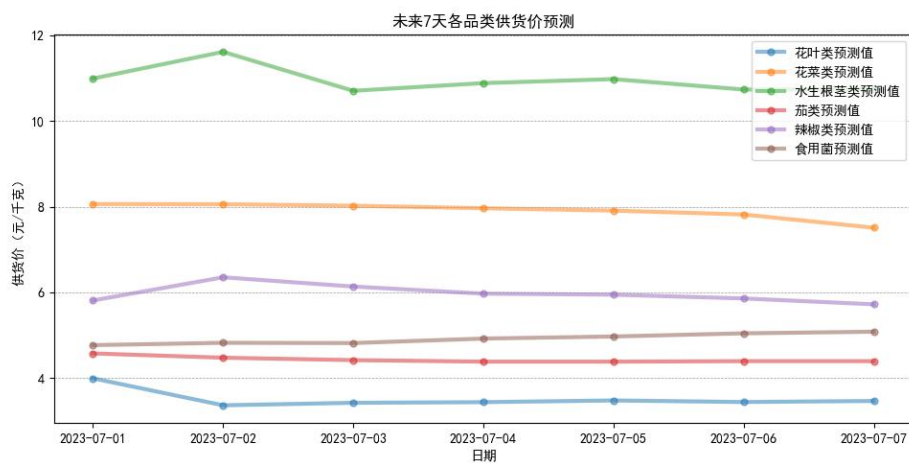
5.3.2.2 对于商超补货成本的 XGBoost 预测模型

为优化 GBDT 易过拟合问题，我们采用 XGBoost 算法给 GBDT 算法增加正则化项，使用 gain 计算每次叶子节点是否分裂。它基于决策树的集成机器学习算法，它以梯度提升 (Gradient Boost) 为框架，利用加法模型与前向分步算法实现学习的优化过程。其基本模型如下：

$$\textcircled{1} |D| = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

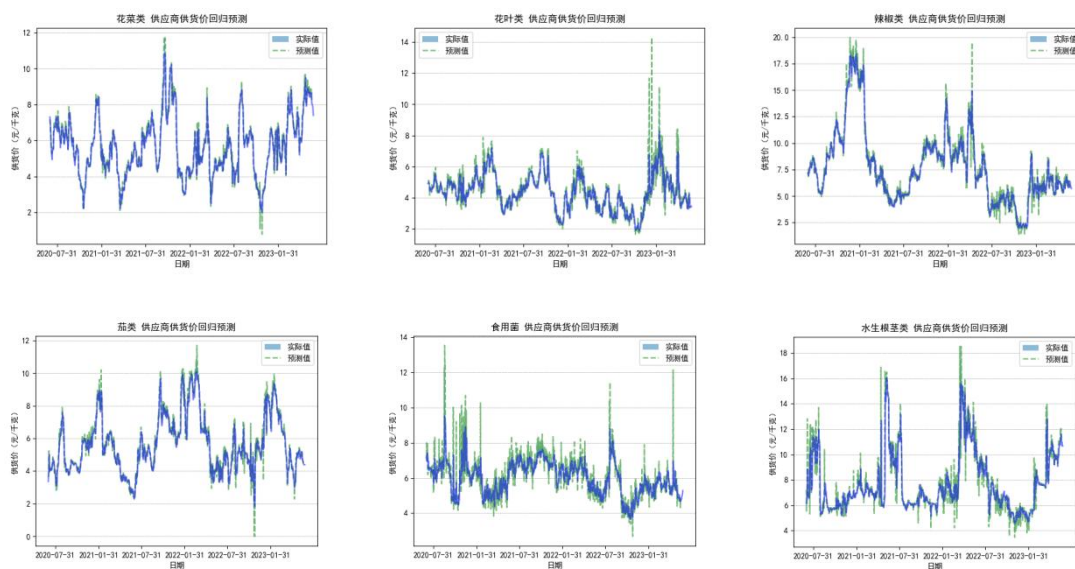
$$\textcircled{2} \hat{y}_i = \phi(x_i) = \sum_{k=1}^K f_k(x_i)$$

根据模型，易得各品类未来 7 天的各品类蔬菜的补货成本，即供货价，如图 8。



(图 8 未来 7 天各品类供货价预测图)

将所有预测数据与真实数据对比，易知模型的拟合效果较好, 准确度较高。



(图 9 各品类的模型预测效果对比)

基于模型预测结果可知，商家补货成本 n_2 为：

$$\textcircled{3} \quad n_2 = \sum_{k=1}^K f_k(x_i)$$

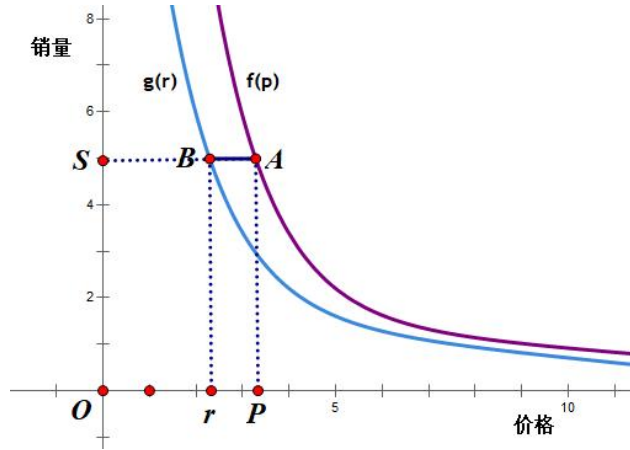
构造的描述品类销售总量与成本加成定价关系的指数函数反映了市场的宏观规律，在此基础上，使用成本加成定价表达式等量代换指数函数中的成本加成定价得：

$$\begin{cases} p = n_2 + r \\ s = A \times e^{-B(n_2+r)} + C \end{cases}$$

可知商超收益为：

$$m = s \times r$$

上述等量代换在平面直角坐标系中的变换以图 4 为例，体现为价格-销量函数 $s = f(p)$ (*色曲线)向左平移 n 个单位得到利润-销量函数 $s = g(r)$ (*色曲线)。



(图 10 成本-销量、利润-销量关系图)

5.3.2.3 商超收益模型最优化

根据题意，需求解商超收益最大化时的各蔬菜品类未来一周(2023 年 7 月 1-7 日)的日补货总量和定价策略。以下运算求解商超收益最大值时模型状态。

商超收益模型表达式展开得：

$$\begin{aligned} \textcircled{1} \quad m &= (A \times e^{-B(n_2+r)} + C) \times r \\ &= A \times r \times e^{-B(n_2+r)} + r \times C \end{aligned}$$

对上式求一阶微分得：

$$\textcircled{2} \quad m' = \frac{dm}{dr} = A(1 - B \times r)e^{-B(n_2+r)} + C$$

当一阶微分等于零值时可以求得：

$$\textcircled{3} \quad (1 - B \times r)e^{-B(n_2+r)} = -\frac{C}{A}$$

对原式求二阶微分：

$$\textcircled{4} \quad m'' = \frac{d^2m}{dx^2} = A(1 - B \times r)(1 - B) \times e^{-B \times (n_2+r)}$$

将一阶微分等于零值的等式带入二阶微分得在原式极值点的二阶微分为：

$$\textcircled{5} \quad m'' = C(B - 1)$$

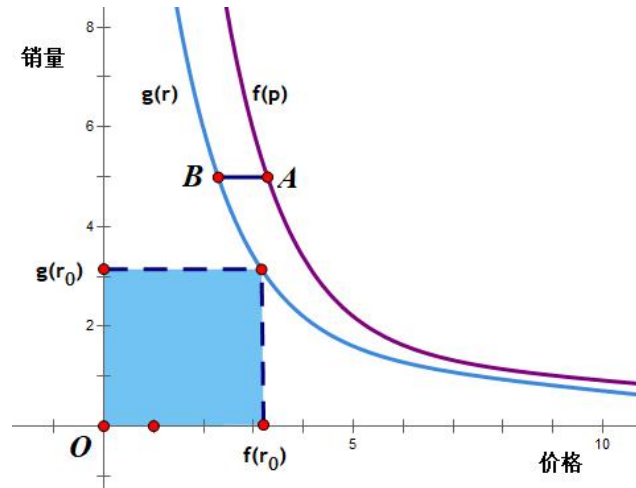
根据以上拟合得到的六组参数 A, B, C 计算可知，此六大类均满足：

$$\textcircled{6} \quad m'' = C(B - 1) < 0$$

故得到原函数存在极大值点，其解 r_0 满足下式：

$$\textcircled{7} \quad (1 - B \times r_0)e^{-B(n_2+r_0)} = -\frac{C}{A}$$

根据以上结论，可以判断下图中曲线上存在一个点 $P(r_0, g(r_0))$ 使商超收益达到最大值，且品类销售总量与品类价格的关系满足市场的宏观规律。



(图 11 收益最大值示意图)

5.3.2.4 商超收益模型求解结果

经过上述分析，结合商超收益最优化模型解析式，可以求得 2023 年 7 月 1 日至 2023 年 7 月 7 日的每日补货总量和每日定价策略。

表 5 每日补货总量

蔬菜品类 预测日期	花叶类	花菜类	水生根茎类	茄类	辣椒类	食用菌
2023-7-1	160.62	8.08	4.70	16.31	39.48	30.32
2023-7-2	160.53	6.58	5.39	16.32	42.05	24.65
2023-7-3	160.57	9.36	5.47	16.32	39.10	31.43
2023-7-4	160.60	9.21	5.94	16.32	38.11	27.99
2023-7-5	160.57	8.29	6.68	16.34	36.18	28.49
2023-7-6	160.54	12.62	5.15	16.30	39.55	25.10
2023-7-7	160.53	8.51	6.39	16.34	43.62	22.77

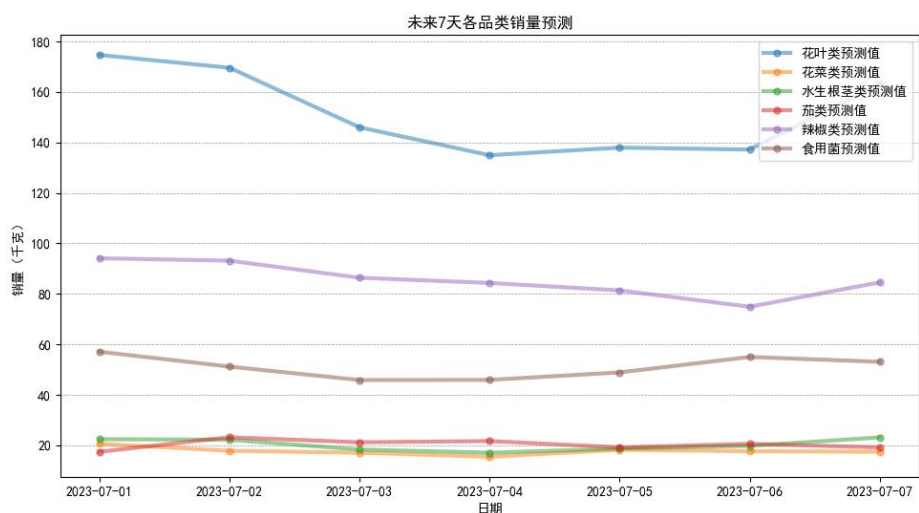
表 6 每日定价策略

蔬菜品类 预测日期	花叶类	花菜类	水生根茎类	茄类	辣椒类	食用菌
2023-7-1	11.59	20.78	24.18	19.83	26.41	16.25
2023-7-2	12.75	21.34	23.41	19.62	25.40	17.35
2023-7-3	12.09	20.30	23.33	19.76	26.56	16.04
2023-7-4	11.72	20.36	22.83	19.69	26.95	16.70
2023-7-5	12.02	20.71	22.11	19.26	27.72	16.61
2023-7-6	12.58	19.09	23.67	20.09	26.39	17.26
2023-7-7	12.62	20.62	22.39	19.41	24.78	17.71

5.3.3 商超收益模型验证

5.3.3.1 历史销量模型预测

各品类自 2020 年 7 月 1 日至 2023 年 6 月 30 日的销售总量数据较大，在不使用非参数回归分析所得到的函数解析解计算最优策略时，使用基于 XGBoost 改进的时间序列预测模型可得出局部较优的销量估计

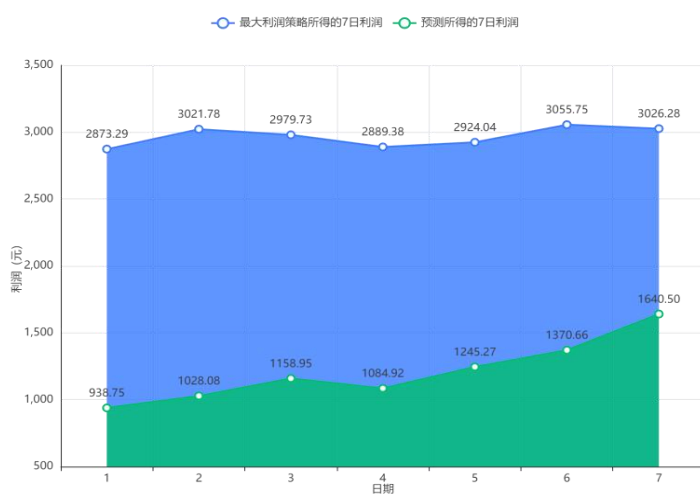


(图 12 未来 7 天各品类销量预测)

5.3.3.2 商超收益模型与历史销量模型验证

通过对比使用非参数回归分析的商超利益模型与使用 XGBoost 预测销量的预测模型最终产生的进货量与定价策略，下图可见商超利益模型于 2023 年 7 月 1 日至 7 日所产生的期望利益显著大于时间序列预测的销量模型。由此可见市场的供需关系决定的销量与价格的关系主要受市场自身因素影响，时间对此市场宏观特征的影响相对较小。

综上所述，本文所建立的商超利益模型具有较大可信度及对市场的适应性。



(图 13 两种模型利润结果对比)

5.4 问题三模型建立与求解

5.4.1 可售单品 7 月 1 日补货成本预测

5.4.1.1 可售单品时间序列构造

对附件二中与附件三 2023 年 6 月 24 - 30 日的可售品种数据进行预处理，得到以下 49 类可售单品类别

表 7 可售单品表

西峡花菇 (1)	苋菜	云南生菜	竹叶菜	上海青	菜心	木耳菜
云南油麦菜	紫茄子(2)	青茄子(1)	西兰花	净藕(1)	菠菜	高瓜(1)
娃娃菜	红薯尖	螺丝椒	菱角	外地茼蒿	红莲藕带	奶白菜
白玉菇 (袋)	芜湖青椒 (1)	野生粉藕	长线茄	小青菜 (1)	云南生菜 (份)	云南油麦菜(份)
菠菜(份)	鲜木耳 (份)	小米椒(份)	虫草花 (份)	小皱皮 (份)	青线椒 (份)	螺丝椒 (份)
七彩椒 (2)	高瓜(2)	姜蒜小米椒组合装(小份)	紫茄子 (1)	枝江青梗散花	双抱菇 (盒)	青红杭椒组合装 (份)
红椒(2)	蟹味菇与白玉菇双拼(盒)	木耳菜(份)	圆茄子 (2)	洪湖藕带	金针菇 (盒)	海鲜菇 (包)

同时将此部分可售单品的销售总量与成本加成定价筛出零值与异常值后，分别使用问题二的商超补货成本的 XGBoost 预测模型构造关于单品单日补货成本的时间序列。

5.4.1.2 XGBoost 补货成本模型

上述时间序列切分训练集与测试集后使用 XGBoost 模型训练，得到可用于通过历史可售单品销售总量对 2023 年 7 月 1 日的 49 种单品进货价格的预测。其中模型优化后平均准确率为

$$\overline{Accuracy} = \frac{\sum Accuracy_i}{49} = 77.52\%$$

5.4.2 可售单品改进收益模型

5.4.2.1 可售单品改进收益模型建立

本问中 49 种可售单品销售总量和成本加成定价二者的性质与问题二中 6 种品类的销售总量和成本加成定价的性质相似，同样对市场供需关系影响所销量与价格关系进行描述。因此同样使用问题二中以下期望构造函数对本问 49 种单品销售总量与成本加成定价的关系进行拟合描述。

$$s = A \times e^{-B \times p} + C$$

本问中所不同的是，经过 spearman 相关性分析，并非 49 种可售单品均为具有强相关性。由以下散点分布图对此部分数据的侧写，可以推断对于单品而言价格分布相较于品类整体更为离散稀疏。

价格分布的离散影响了拟合函数描述销售总量与成本加成定价的关系的置信度，因此需引入以下销售总量与成本加成定价所组成的有序数对 $K(p_r, s_r)$ 序列的概率密度函数 $pdf(K_t)$ ，对稀疏数据拟合误差进行加权修正。

$$pdf(K_r) = \frac{1}{N} \sum_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1(K_r - K_i)^2}{2\sigma^2}}$$

此时，改进后的模型为

$$Situation(m) = s \times r \times pdf(K_r)$$

其中, $Situation(m)$ 描述商超在指定条件下的期望利益系数, 系数越大说明商超利益达到对应最大值的期望越大。

5.4.2.2 模型销售总量与成本成定价拟合与分析

对构造的单品销售总量对于成本加成定价的以自然常数 e 为幂的指数函数函数 $s = A \times e^{-B \times p} + C$ 分别进行拟合。通过非参数拟合方法在其函数雅可比矩阵估计误差的残差最小值处取得最优参数。

5.4.2.3 模型概率密度函数求解

单品销售总量与成本加成定价的分布形成的有限个有序数对 $K(p_r, s_r)$ 可通过核密度估计 (kernel density estimation) 对其所在区域的概率密度函数进行估计。核密度估计是一种非参数检验方法, 在选择合适的核函数后通过残差回归出最优的概率密度函数

$$kde(K(p_0, s_0), \dots, K(p_i, s_i))$$

此函数可以计算临近区域的分布概率。

5.4.3 单品补货量与定价策略模型

5.4.3.1 启发式模型建立

遗传算法使用群体搜索技术, 将种群代表一组问题解, 通过对当前种群事假选择、交叉和变异等一系列遗传操作来产生新一代的种群, 并逐步使种群进化到包含近似最优解的状态。

本问中的情景涵盖了单品种类数量、销售最小陈列质量、市场的商品需求以及商超利益等诸多不完全确定因素, 在遗传算法中此部分因素可以视为自然环境。而单品补货量与定价策略可以视为种群间差异的基因型, 此模型通过问题情景中的因素对单品补货量与定价策略进行“自然选择”, 使策略逐步近似到最优解。

5.4.3.2 遗传算法模型求解

在遗传算法模型中需要设定最大值、最小值、种群大小、交叉率、变异率和

繁殖代数等参数类变量，同时也需要对适应度函数、个体选择函数、基因重组函数、基因突变函数以及自然选择流程等方法进行定义。

在本模型中，决定模型优劣的核心在于适应度函数的定义是否合适。本文对其定义如下：

$$\begin{cases} count > 33 \text{ or } count < 27, reward = 0 \\ E(s) < 2.5, reward = reward - 100 \times E(m) \times pdf(K_t) \\ others, reward = reward + E(m) \times pdf(K_t) \end{cases}$$

其含义分布为：

- ① 当单品数量不在[27,33]区间中时，重置该个体的适应度
- ② 当单品质量小于 2.5kg 时，适当减小个体适应度
- ③ 当正常情况时，个体适应度涨幅与商超期望利益成正相关关系

5.4.4 单品补货与定价最优估计

表 8 补货与定价最优估计结果

商品名称	销量	价格	商品名称	销量	价格
苋菜	2.5	18.01	小米椒(份)	17.98	4.71
云南生菜	19.64	8.01	小皱皮(份)	5.95	5.83
木耳菜	3.68	6.01	青线椒(份)	4.93	4.79
云南油麦菜	7.10	8.92	螺丝椒(份)	10.87	5.73
西兰花	16.53	189.27	高瓜(2)	2.82	25.91
净藕(1)	2.5	19.68	姜蒜小米椒组合装 (小份)	5.10	4.53
高瓜(1)	2.5	19.64	枝江青梗散花	13.24	10.22
娃娃菜	6.12	40.79	双孢菇(盒)	14.43	5.46
红薯尖	6.92	7.98	青红杭椒组合装 (份)	5.51	5.8
螺丝椒	5.59	25.8	木耳菜(份)	2.5	4.89
菱角	2.5	16.01	洪湖藕带	2.5	31.76
红莲藕带	2.5	10.01	金针菇(盒)	36.35	5.78
奶白菜	7.56	6.01	海鲜菇(包)	6.09	14.39

白玉菇(袋)	4.19	7.92	小青菜(1)	5.97	8.01
芜湖青椒(1)	27.96	13.99	云南生菜(份)	35.82	206.4
长线茄	3.57	14	菠菜(份)	16.43	4.99
鲜木耳(份)	2.5	2.67			

5.5 问题四的模型建立与求解

5.5.1. 线性回归分析（最小二乘法）

首先利用最小二乘法对自变量温度和降雨量与因变量蔬菜品类分别进行线性回归分析，本文通过 spss pro 软件得出表 9，由表可知所有蔬菜种类与温度和降雨量具有线性关系，这为后续使用典型相关分析具有重要意义。

表 9 蔬菜品类和天气中国气温和降水量的拟合情况

蔬菜品类	与天气的线性拟合函数
水生根茎类	$y=49.151 - 0.882 \times \text{平均气温} + 0.047 \times \text{降水量}$
花菜类	$y=37.627 + 0.054 \times \text{平均气温} + 0.022 \times \text{降水}$
食用菌	$y=93.696 - 1.722 \times \text{平均气温} - 0.017 \times \text{降水量}$
辣椒类	$y=93.022 - 0.62 \times \text{平均气温} - 0.026 \times \text{降水量}$
花叶类	$y=176.703 + 0.388 \times \text{平均气温} + 0.168 \times \text{降水量}$
茄类	$y=14.969 + 0.418 \times \text{平均气温} + 0.0 \times \text{降水量}$

5.5.2 典型相关分析（Canonical Correlation Analysis, CCA）

5.5.2.1 模型合理性判断

典型相关分析用于研究两组变量之间的线性相关关系，而本文以及得知温度和降水量对于不同蔬菜品类都具有线性关系，且将不同蔬菜品类和温度与降水量分别分到两个集合 Y,X 中（变量：集合 Y:{水生根茎类，花叶类，花菜类，茄类，

辣椒类，食用菌}；集合 X:{平均气温，降水量}），因此研究两个集合直接按的线性相关关系，满足模型使用要求且符合题意。

5.5.2.2 典型相关系数

典型相关系数（Canonical Correlation Coefficient）是一种用于测量两组多元变量之间关联程度的统计量，通常用于探究两组变量之间是否存在关联或相关性，以及了解它们之间的模式或结构。它核心思想是通过线性组合来找到一对典型变量，这对典型变量能最大程度地捕捉了两组变量之间的相关性。利用 Spsspro 计算得到的具体系数如表 10 所示，公式如①所示，其中 $cov(U_i, V_i)$ 为 U_i 和 V_i 的协方差。

$$\textcircled{1} \quad cov(U_i, V_i) = \sum_{k=1}^p \sum_{l=1}^q a_{jk} b_{jl} cov(X_k, Y_l)$$

表 10 典型相关系数

典型变 量对	典型相关 系数	特征 值	Wilks	模型自 由度	误差自 由度	F	P
第 1 对	0.65	0.423	0.571	12	2154	57.958	0.000***
第 2 对	0.099	0.01	0.99	5	1078	2.138	0.059*

注：***、**、*分别代表 1%、5%、10%的显著性水平

由上表可知前 1 对典型变量通过显著性检验，认为前 1 对典型变量之间的相关性显著。第一对典型变量的相关系数为 0.65

5.3.2.3 典型变量系数

典型变量系数（Canonical Variable Coefficients）作为典型变量的系数，可得到典型变量的组成公式。典型变量在典型相关分析里面是集合里面原始变量

的线性组合，目的是能最大限度捕捉出两组变量的相关性，类似于特征提取和降维，具体数学公式为公式②

$$\textcircled{2} \quad \rho_i^* = \frac{\text{cov}(U_i, V_i)}{\sqrt{\text{var}(U_i)\text{var}(V_i)}}$$

具体的典型变量参数如表 11 所示

表 11 典型变量系数

	典型变量 Y1	典型变量 Y2
水生根茎类	0.01	0.03
花叶类	-0.007	0.011
花菜类	-0.008	-0.004
茄类	-0.032	-0.012
辣椒类	0	-0.016
食用菌	0.019	-0.009

	典型变量 X1	典型变量 X2
平均气温	-0.091	-0.02
降水量	-0.002	0.037

利用 spss pro 得出四个典型变量的计算公式：

(1) 集合 Y 的第 1 典型变量的计算公式：Y1=0.01×水生根茎类-0.007×花叶类-0.008×花菜类-0.032×茄类-0.0×辣椒类+0.019×食用菌

(2) 集合 Y 的第 2 典型变量的计算公式：Y2=0.03×水生根茎类+0.011×花叶类-0.004×花菜类-0.012×茄类-0.016×辣椒类-0.009×食用菌

(3) 集合 X 的第 1 典型变量的计算公式：X1=-0.091×平均气温-0.002×降水量

(4) 集合 X 的第 2 典型变量的计算公式：X2=-0.02×平均气温+0.037×降水量

5.3.2.4 典型负荷系数和交叉符合系数

典型载荷系数是指一个典型变量与本组所有变量的简单相关系数。交叉载荷系数是指一个典型变量与另一组变量各个变量的简单相关系数。其的绝对值越大说明该项与典型变量之间的相关关系越强。通过下表 12 可得出结论：典型变量 Y1 与温度（平均气温）之间存在较强的负相关关系，典型变量 Y2 与温度之间的相关性较弱。典型变量 Y1 与降水量之间存在一定的负相关关系，而典型变量 Y2 与降水量之间存在较强的正相关关系。

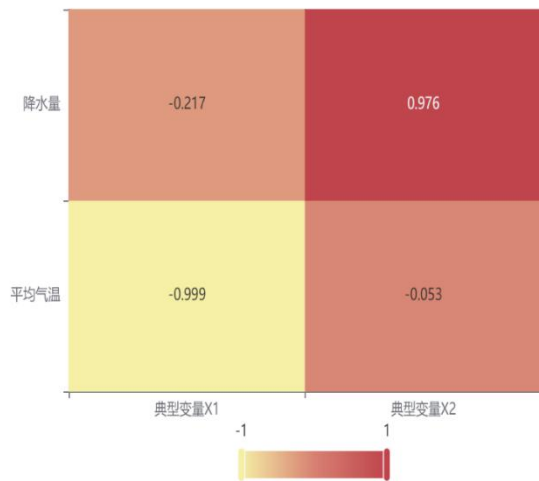
表 12 典型负荷系数和交叉负荷系数

变量	典型载荷系数		交叉载荷系数	
	典型变量 Y1	典型变量 Y2	典型变量 X1	典型变量 X2
水生根茎类	0.458	0.571	0.298	0.057
花叶类	-0.093	0.5	-0.061	0.05
花菜类	-0.049	0.243	-0.032	0.024
茄类	-0.521	-0.172	-0.339	-0.017
辣椒类	0.199	-0.065	0.13	-0.006

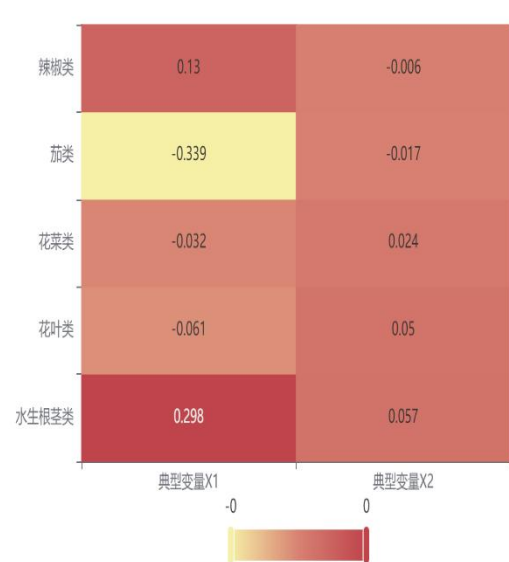
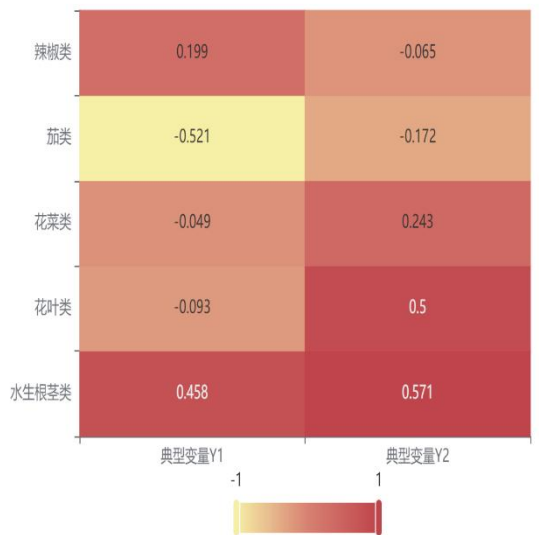
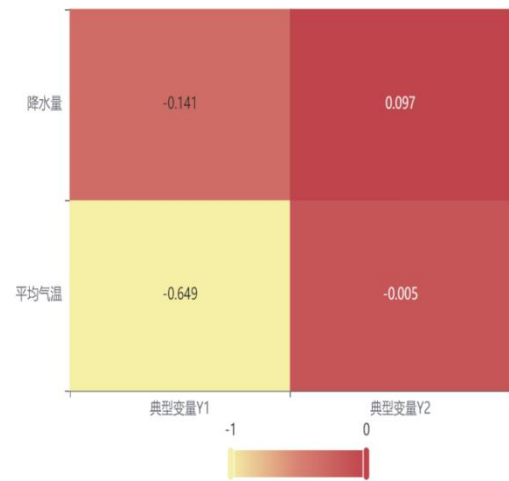
变量	典型载荷系数		交叉载荷系数	
	典型变量 X1	典型变量 X2	典型变量 Y1	典型变量 Y2
平均气温	-0.999	-0.053	-0.649	-0.005
降水量	-0.217	0.976	-0.141	0.097

5.3.2.5 载荷矩阵热力图

集合 X 交叉载荷矩阵热力图



集合 X 典型载荷矩阵热力图



集合 Y 典型载荷矩阵热力图

集合 Y 交叉载荷矩阵热力图

以上四张载荷矩阵热力图直观的反映出典型变量与本组所有变量的相关性或典型变量与另一组变量的相关性。为了更加直观且量化四个典型变量对变量之间的相互影响，本文使用了方差解释比例去反映四个典型变量分别反映的信息量，如表 13 所示，展示了典型变量的解释比例，包括组内解释比例和交叉解释比例，定量地判断典型变量所包含的原始信息量的大小。由此本文得出 X1 在此数据集中的解释性最强。尽管 X2 的解释能力低于 X1，但其仍具有一定的解释能力。通过观察 X1,X2 的计算公式本文得出结论：温度和降雨量都对不同蔬菜类的品类销售量具有不同程度的相互关系，而温度对不同蔬菜平类的销售量相关性关联

性强于降雨量。

表 13 方差解释比例

	集合 X(%)	集合 Y(%)
典型变量 X1	52.204	14.849
典型变量 X2	47.796	11.361
典型变量 Y1	6.28	22.078
典型变量 Y2	0.112	0.469

5.3.3 对于商品补货和定价的决策优化

本文在问题二已经提出了销售总量关于成本加成定价和补货关系的模型，因此关于定价和补货量都与销售量有一定关系，从而可以通过天气和不同蔬菜的品类销售量之间的关系推出其定价和补货策略，模型表达式如下③所示：

$$\textcircled{3} \quad \begin{cases} p = n + r \\ s = A \times e^{-B(n+r)} + C \end{cases}$$

在本题中使用典型相关性分析已经得出在天气的影响因素下，温度与不同蔬菜品类呈现的相关性最明显，通过典型变量系数得到的典型变量其中的 X1 解释的各个变量中指标的信息量涵盖面较广，即 X1 与目标变量具有强相关性，X2 与品类销售量的相关性弱于 X1。对于典型变量 X1，X2，商家可以利用其和蔬菜品类销售量数据关系方面的强解释能力，进行特征选择。

基于典型变量 X1, X2 在蔬菜品类的销售量中强大的解释能力，其可以用在模型的特征选择。典型相关分析可以确定哪些典型变量与目标变量之间具有强相关性。因此，商家可以优先考虑将这些重要的典型变量包含在模型中，同时忽略那些对目标变量影响较小或无关的变量，例如在销售量预测方面，将温度列为重要特征值之一。特征选择另外一个好处是可以简化模型，不是所有的特征都对目标变量具有重要影响，有时候使用太多的特征反而会引入噪音并降低模型的性能。因此商家可以根据典型变量在解释数据关系方面的能力，有效的选取适当的特征，

剔除无效特征值，防止模型被污染。

5.3.4 模型总结

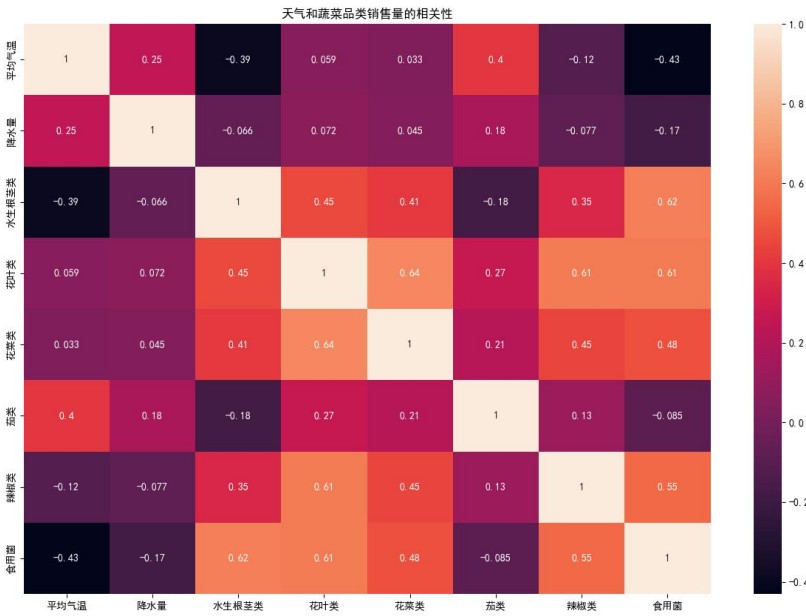
本文使用典型相关分析方法对天气的温度和降雨量与不同蔬菜品类的销售量之间的关系进行分析，从而得到了典型变量 X1，X2。在商家制定蔬菜商品的定价和补货策略时，本文提出可以利用一些典型变量与数据的强解释能力进行数据的特征选取，从而减少噪声干扰，简化模型，能够更加准确进行预测或者决策。

七、模型检验

7.2 针对问题四的典型相互分析的检验

在分析温度和降雨对不同蔬菜品类的相关性影响时，本文采用典型相互分享相关性分析，为了检验此模型的准确性，本文采用 Pearson 相关性系数进行相关系数统计指标分析相关性。

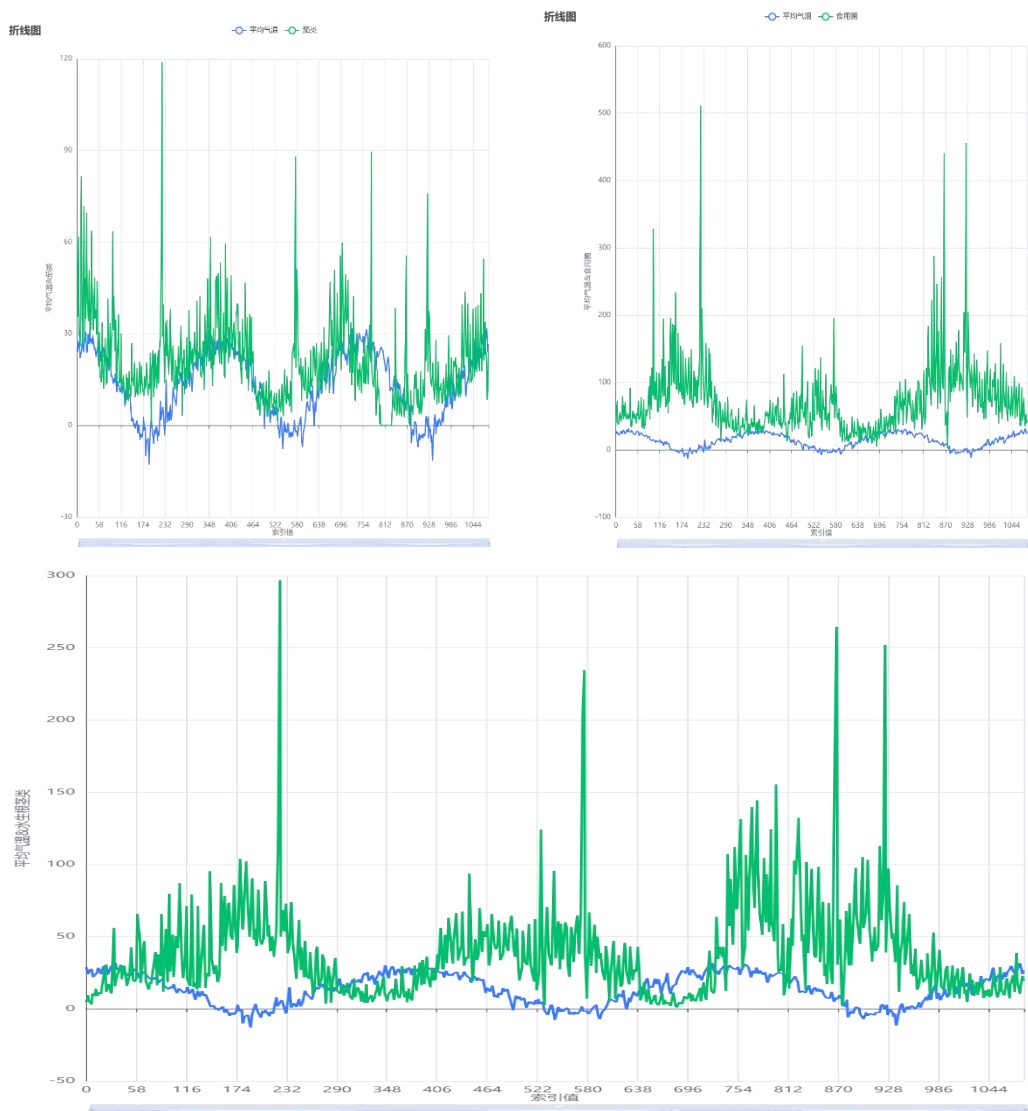
根据，随即我们利用热力图反应其相关性系数，如图 14 所示。



(图 14 相关性热力图)

在绘制完相关性热力图后，本文选出其中相关性最高的三组，分别是温度

和水生根茎类，茄类和食用菌类之间的关系，其中水生根茎和食用菌类和温度呈现的是负相关性，而茄类呈现的是正相关性，因此我们使用折线图展现温度和它们的变化规律。具体情况如（图 15）所示



(图 15 温度和销量的相关性)

由此可知 Pearson 相关性分析结果良好，大致反应了不同蔬菜品类和温度、降水的相关性。得到的结论是温度和不同蔬菜销售量的相关性总体较强，而降水量和蔬菜品类之间的相关性较弱。因此典型相关模型检验成功。

十、数据来源

2020-07-01 至 2023-06-30 气温、降雨数据: <https://rp5.ru/>中国天气_

十一、参考文献

- [1] 葛继科, 邱玉辉, 吴春明, 等. 遗传算法研究综述[J]. 计算机应用研究, 2008, 25(10): 2911-2916.
- [2] Chen T, He T, Benesty M, et al. Xgboost: extreme gradient boosting[J]. R package version 0.4-2, 2015, 1(4): 1-4.
- [3] 王学仁. 多个变量集合的典型相关分析[J]. 云南大学学报(自然科学版), 1983, (Z1): 5-19.

十二、附录

附录一 附录列表

- 1. 问题一-数据预处理.xlsx
- 2. 问题二-拟合函数参数.csv
- 3. 问题二-最终各品类定价.csv
- 4. 问题二-最终各品类补货量.csv
- 5. 问题三-最终补货量与定价.csv
- 6. 问题三-7月1日成本价格.csv
- 7. 问题三-优化商超利益模型最大期望值.csv
- 8. 问题四-网络数据集天气情况.xlsx
- 9. question-1.py
- 10. question-2.py
- 11. question-3.py
- 12. question-4.py

附录二 基于概率密度函数改进的遗传算法

```
1. from operator import itemgetter
```

```

2.     import random
3.
4.
5.     class Gene:
6.         """
7.         This is a class to represent individual(Gene) in GA algorithm
8.         m
9.         each object of this class have two attribute: data, size
10.        """
11.        def __init__(self, **data):
12.            self.__dict__.update(data)
13.            self.size = len(data['data']) # length of gene
14.
15.
16.        class GA:
17.            """
18.            This is a class of GA algorithm.
19.            """
20.
21.            def __init__(self, parameter):
22.                """
23.                Initialize the pop of GA algorithm and evaluate the pop
24.                by computing its' fitness value.
25.                The data structure of pop is composed of several individuals
26.                which has the form like that:
27.                {'Gene':a object of class Gene, 'fitness': 1.02(for example)}
28.                Representation of Gene is a list: [b s0 u0 sita0 s1 u1 sita1 s2 u2 sita2]
29.                """
30.                # parameter = [CXPB, MUTPB, NGEN, popsize, low, up]
31.                self.parameter = parameter
32.
33.                low = self.parameter[4]
34.                up = self.parameter[5]
35.
36.                self.bound = []
37.                self.bound.append(low)
38.                self.bound.append(up)
39.
40.                pop = []

```

```

41.         for i in range(self.parameter[3]):
42.             geneinfo = []
43.             for pos in range(len(low)):
44.                 # initialise popluation
45.                 geneinfo.append(random.randint(
46.                     self.bound[0][pos], self.bound[1][pos]))
47.
48.             fitness = self.evaluate(geneinfo) # evaluate each ch
romosome
49.             # store the chromosome and its fitness
50.             pop.append({'Gene': Gene(data=geneinfo), 'fitness': f
itness})
51.
52.         self.pop = pop
53.         # store the best chromosome in the population
54.         self.bestindividual = self.selectBest(self.pop)
55.
56.     def evaluate(self, geneinfo):
57.         """
58.         fitness function
59.         未超过 2.5 奖励-未超过部分*100
60.         不在 27-33 之间 奖励-1000
61.         常规奖励 x*y
62.         """
63.         y=0
64.
65.         for i in range(len(geneinfo)):
66.             if ex_m_origin_piont[i][1]<2.5:
67.                 sco=np.exp(cmf[i][0].score_samples([[ex_m_origin_
piont[i][0],2.5]]))
68.                 div_c=(cmf[i][1][0]-sco)/cmf[i][1][0]
69.                 y += -100*(2.5-ex_m_origin_piont[i][1])*div_c
70.
71.
72.         for i in range(len(geneinfo)):
73.             sco = np.exp(cmf[i][0].score_samples(
74.                 [[ex_m_origin_piont[i][0], 2.5]]))
75.             div_c = (cmf[i][1][0]-sco)/cmf[i][1][0]
76.             y += geneinfo[i]*ex_m_piont[i][0]*ex_m_piont[i][1]*di
v_c
77.         num=sum(geneinfo)
78.
79.         if num not in range(27,34):
80.             y=0

```

```

81.         y=y if y>0 else 0
82.         return y
83.
84.     def selectBest(self, pop):
85.         """
86.         select the best individual from pop
87.         """
88.         s_inds = sorted(pop, key=itemgetter(
89.             "fitness"), reverse=True)      # from large to sm
        all, return a pop
90.         return s_inds[0]
91.
92.     def selection(self, individuals, k):
93.         """
94.         select some good individuals from pop, note that good ind
        ividuals have greater probability to be choosen
95.         for example: a fitness list like that:[5, 4, 3, 2, 1], su
        m is 15,
96.         [-----|----|---|--|-]
97.         012345|6789|101112|1314|15
98.         we randomly choose a value in [0, 15],
99.         it belongs to first scale with greatest probability
100.        """
101.        s_inds = sorted(individuals, key=itemgetter("fitness"),
102.            reverse=True) # sort the pop by the refe
        rence of fitness
103.        # sum up the fitness of the whole pop
104.        sum_fits = sum(ind['fitness'] for ind in individuals)
105.
106.        chosen = []
107.        for i in range(k):
108.            # randomly produce a num in the range of [0, sum_fits]
        , as threshold
109.            u = random.random() * sum_fits
110.            sum_ = 0
111.            for ind in s_inds:
112.                sum_ += ind['fitness'] # sum up the fitness
113.                if sum_ >= u:
114.                    # when the sum of fitness is bigger than u, c
        hoose the one, which means u is in the range of
115.                    # [sum(1,2,...,n-1),sum(1,2,...,n)] and is ti
        me to choose the one ,namely n-th individual in the pop
116.                    chosen.append(ind)
117.                    break

```

```

118.         # from small to large, due to list.pop() method get the l
            ast element
119.         chosen = sorted(chosen, key=itemgetter("fitness"), revers
e=False)
120.         return chosen
121.
122.     def crossover(self, offspring):
123.         """
124.         cross operation
125.         here we use two points crossover
126.         for example: gene1: [5, 2, 4, 7], gene2: [3, 6, 9, 2], if
pos1=1, pos2=2
127.             5 | 2 | 4  7
128.             3 | 6 | 9  2
129.             =
130.             3 | 2 | 9  2
131.             5 | 6 | 4  7
132.         """
133.         dim = len(offspring[0]['Gene'].data)
134.
135.         # Gene's data of first offspring chosen from the selected
            pop
136.         geninfo1 = offspring[0]['Gene'].data
137.         # Gene's data of second offspring chosen from the selecte
            d pop
138.         geninfo2 = offspring[1]['Gene'].data
139.
140.         if dim == 1:
141.             pos1 = 1
142.             pos2 = 1
143.         else:
144.             # select a position in the range from 0 to dim-1,
145.             pos1 = random.randrange(1, dim)
146.             pos2 = random.randrange(1, dim)
147.
148.         newoff1 = Gene(data=[]) # offspring1 produced by cross o
            peration
149.         newoff2 = Gene(data=[]) # offspring2 produced by cross o
            peration
150.         temp1 = []
151.         temp2 = []
152.         for i in range(dim):
153.             if min(pos1, pos2) <= i < max(pos1, pos2):
154.                 temp2.append(geninfo2[i])

```

```

155.             temp1.append(geninfo1[i])
156.         else:
157.             temp2.append(geninfo1[i])
158.             temp1.append(geninfo2[i])
159.         newoff1.data = temp1
160.         newoff2.data = temp2
161.
162.         return newoff1, newoff2
163.
164.     def mutation(self, crossoff, bound):
165.         """
166.         mutation operation
167.         """
168.         dim = len(crossoff.data)
169.
170.         if dim == 1:
171.             pos = 0
172.         else:
173.             # chose a position in crossoff to perform mutation.
174.             pos = random.randint(0,1)
175.
176.             crossoff.data[pos] = random.randint(bound[0][pos], bound[
177. 1][pos])
178.             return crossoff
179.
180.     def GA_main(self):
181.         """
182.         main frame work of GA
183.         """
184.         popsize = self.parameter[3]
185.
186.         print("Start of evolution")
187.
188.         # Begin the evolution
189.         for g in range(self.parameter[2]):
190.             print("##### Generation {} #####".
191. format(g))
192.
193.             # Apply selection based on their converted fitness
194.             selectpop = self.selection(self.pop, popsize)
195.
196.             nextoff = []
197.             while len(nextoff) != popsize:

```

```

197.                                     # Apply crossover and mutation on the offspring
198.
199.                                     # Select two individuals
200.                                     offspring = [selectpop.pop() for _ in range(2)]
201.
202.                                     if random.random() < self.parameter[0]: # cross
                                         two individuals with probability CXPB
203.                                     crossoff1, crossoff2 = self.crossoverate(offspring)
204.                                     if random.random() < self.parameter[1]: # mutate
                                         two individuals with probability MUTPB
205.                                     muteoff1 = self.mutation(crossoff1, self.
                                         bound)
206.                                     muteoff2 = self.mutation(crossoff2, self.
                                         bound)
207.                                     # Evaluate the individuals
208.                                     fit_muteoff1 = self.evaluate(muteoff1.data)
209.                                     # Evaluate the individuals
210.                                     fit_muteoff2 = self.evaluate(muteoff2.data)
211.                                     nextoff.append(
212.                                         {'Gene': muteoff1, 'fitness': fit_muteoff1})
213.                                     nextoff.append(
214.                                         {'Gene': muteoff2, 'fitness': fit_muteoff2})
215.                                     else:
216.                                         fit_crossoff1 = self.evaluate(
217.                                             crossoff1.data) # Evaluate the individuals
218.                                         fit_crossoff2 = self.evaluate(crossoff2.data)
219.                                         nextoff.append(
220.                                             {'Gene': crossoff1, 'fitness': fit_crossoff1})
221.                                         nextoff.append(
222.                                             {'Gene': crossoff2, 'fitness': fit_crossoff2})
223.                                     else:
224.                                         nextoff.extend(offspring)
225.
226.                                     # The population is entirely replaced by the offspring

```

```

227.         self.pop = nextoff
228.
229.         # Gather all the fitnesses in one list and print the
        stats
230.         fits = [ind['fitness'] for ind in self.pop]
231.
232.         best_ind = self.selectBest(self.pop)
233.
234.         if best_ind['fitness'] > self.bestindividual['fitness
        ']:
235.             self.bestindividual = best_ind
236.
237.             print("Best individual found is {}, {}".format(self.b
        estindividual['Gene'].data,
238.                                                             self.b
        estindividual['fitness']))
239.             print("  Max fitness of current pop: {}".format(max(f
        its)))
240.
241.             print("----- End of (successful) evolution -----")
242.
243.
244.     def ga_t():
245.         CXPB, MUTPB, NGEN, popsize = 0.8, 0.1, 1000, 100 # popsize m
        ust be even number
246.
247.         up = [1 for _ in range(49)] # upper range for variables
248.         low = [0 for _ in range(49)] # lower range for variables
249.         parameter = [CXPB, MUTPB, NGEN, popsize, low, up]
250.         run = GA(parameter)
251.         run.GA_main()

```