基于 LSTM 和遗传算法的商超蔬菜补货和定价决策

摘要

在商超运营中，对于不易保存的生鲜产品，制定合理的补货策略和定价策略是控制成本、最大化收益的关键。本文基于 **ARIMA 模型、LSTM 模型、多元线性回归模型、利润模型、规划模型，**利用 **K-means 聚类、层次聚类、遗传算法**等算法，解决以下问题：

针对问题一，本文将附件数据首先进行预处理，针对所得数据先进行描述性统计，观察数据特征，画出六种蔬菜品类时间分布的**柱形图和百分比图**，总结不同品类蔬菜商品销量随时间季度变化的分布规律和相互关系规律。然后本文对六种品类蔬菜的销量进行**皮尔逊相关性**分析与检验，发现六种品类的蔬菜商品销量之间均为正相关，不存在竞品，接着利用 K-means 聚类和层次聚类模型，对不同单品的蔬菜销量进行时间和特征上的分类，对聚类结果进行归纳分析。

针对问题二，本文对所用数据进行了 **ADF 检验和 Ljung-box 白噪声检验**，检验通过后使用 ARIMA(1,0,3)模型,分别对六个蔬菜品类未来一周需要的补货总量进行预测。由于 ARIMA 模型表现欠佳，本文随后引入 LSTM 神经网络模型来进行优化预测。建立多元线性回归模型，以每日正常出售和打折出售的加权平均价格为解释变量，以每日销售总量为被解释变量，建立多元线性回归模型，对六个蔬菜品类分别进行回归。求出销售总量关于加权平均价格的回归方程式后，根据经济学基本原理，建立利润模型，求解出使利润最大化的平均价格，利用过去三年各品类的平均价格并根据“成本加权定价”方法确定此时的成本加成定价利润率。最后本文根据 LSTM 模型预测出的各品类补货总量和建立的利润模型中使收益最大化的成本加成定价利润率为商超制定未来一周的日补货总量和定价策略，日补货总量结果详见**表 6**，定价策略结果详见**表 16**。

针对问题三，本文根据 2023 年 6 月 24 日至 2023 年 6 月 30 日的七天历史数据挑选出 49 种应季可售蔬菜单品，根据前七天销售数据，基于 LSTM 模型预测出 2023 年 7 月 1 日的各单品销量。根据问题二的利润方程式和品类补货总量预测结果，选择合理的约束条件，以各单品补货量及各单品成本加成定价利润率为自变量，以 7 月 1 日蔬菜商品销售总利润为目标函数，建立规划模型。利用遗传算法，以 LSTM 模型预测结果为初始参数，迭代得到商超收益最大化的规划结果。具体结果见**表 17**。

针对问题四，本文从库存情况、市场形式、节庆假日、天气情况、历史销售记录、顾客消费习惯、季节变化、批发蔬菜的品质、营销策略、基本陈列面、永续订单、商超其他营运成本等十余个角度分析其对商超补货决策和定价决策的影响，并以此对商超运营提出了更优意见，扩大成果出口。

关键词：成本加成定价 补货定价决策 LSTM 聚类 遗传算法

一、问题重述

1.1 问题背景

蔬菜存在保鲜期短且品相会随着时间推移在较短的时间内变差，一般情况下生鲜商超中的蔬菜大多数不能隔日销售，因此商超需要每日对蔬菜商品进行补货。受囿于蔬菜的进货交易一般发生在凌晨，商家必须在不知道具体蔬菜单品和进货价格的情况下，依据各蔬菜单品的历史销售情况和顾客需求情况做出每天的补货决策。“成本加成定价”法是在蔬菜定价中被广泛运用和接受的方法，同时，商超一般还会对运损或者因为未及时卖出而品相变差的蔬菜商品进行折扣销售。基于此，对商超而言，进行合理的补货决策和定价决策，制定合理的销售组合尤为重要。

1.2 问题的数据条件

题目给出的四个附件中，附件 1 是某商超经销的蔬菜商品信息，共包含 6 个蔬菜品类、251 个蔬菜单品。附件 2 是该商超 2020.07.01-2023.06.30 期间各蔬菜单品的销售明细。附件 3 是该商超 2020.07.01-2023.06.30 期间各蔬菜单品的批发价格即进货价格明细。附件 4 是该商超经销的各蔬菜单品近期的损耗率。

1.3 问题的提出

（1）不同品类和不同单品的蔬菜类商品之间可能存在关联，本文须分析各个蔬菜品类和蔬菜单品在销售量上的分布规律和相互关系。

（2）商超在进行补货决策时以蔬菜品类为单位，本文须分析不同蔬菜品类的销售总量与成本加成定价的关系，并据此给出使得该商超未来一周（2023.07.01-2023.07.07）收益最大化的各单品补货量和定价策略。

（3）由于蔬菜商品销售空间限制，结合商超意愿，本文须进一步制定 2023.07.01 的单品补货计划和定价策略，在使单品总数在 27-33 个之间，且每个单品订购量均不低于 2.5 千克最小陈列量的前提下，使商超收益最大化。同时，补货计划需要参考 2023.06.24-2023.06.30 期间的可售品种。

（4）本文须对商超的数据采集给出意见，使商超的数据能更好的服务于蔬菜商品的补货和定价决策，同时说明新采集的数据对本文上述问题的解决有何帮助，并对本文提出的意见给出支撑理由。

二、问题分析

2.1 问题一的分析

本问主要基于附件 1 和附件 2 的数据，进行数据预处理之后先进行描述性统计，然后画出六种蔬菜品类时间分布的柱形图和百分比图，分析总结不同品类和单品蔬菜销量的分布规律和相互关系。随后对六种品类蔬菜的销量进行皮尔逊相关性分析与检验，并利用 K-means 聚类和层次聚类，对不同单品的蔬菜销量进行时间和特征上的分类。

2.2 问题二的分析

本问首先对所用数据进行了 ADF 检验和 Ljung-box 白噪声检验，然后使用模型 ARIMA(1,0,3)，分别对六个蔬菜品类未来一周需要的日补货总量进行预测，并引入 LSTM 神经网络模型来进行优化预测。接着本文建立多元线性回归模型，分别对六个蔬菜品类回归计算。根据回归方程式，结合经济学基本原理，建立利润模型，根据**“成本加权定价”**方法确定此时的利润率，最终为给出商超未来一周的补货和定价决策。

2.3 问题三的分析

本问首先根据七天历史数据挑选出 49 种应季可售蔬菜单品，利用 LSTM 模型预测出 2023 年 7 月 1 日的各单品销量。随后选择合理的约束条件，以各单品补货量及各单品成本加成定价利润率为自变量，以 7 月 1 日蔬菜商品销售总利润为目标函数，建立规划模型。利用遗传算法，以 LSTM 模型预测结果为初始参数，迭代得到商超收益最大化的规划结果。

2.4 问题四的分析

本问从十余个角度分析了未尽因素对商超补货决策和定价决策的影响，并以此对商超运营提出了更优意见。

三、模型假设

1. 假设题目所给的数据真实可靠；
2. 假设不同的蔬菜单品单位质量在销售时所占的空间相同；
3. 假设每一份（盒、袋）蔬菜单品与每一千克蔬菜单品所占空间相同；
4. 假设该商超在蔬菜商品的销售上不存在竞争商家；
5. 本文通过计算发现各单品计算折扣销售后实际平均出售价格相较于原价变动一般不超过 5%，总平均偏差仅为 1.14%，因此本文假设平均出售价格与原价近似相等。

四、定义与符号说明

表 1 符号说明

符号	含义
$P_{平均}$	商品的平均出售价格
AC	商品的平均批发价格
W	总利润
α_0, β_k	多元线性回归系数
K	成本加成定价利润率
S	补货量(进货量)

五、模型建立与求解

5.1 数据预处理

5.1.1 未经销单品

通过附件 1 与附件 2 的数据比对，存在五种商品存在从未出现过销售记录的情况。可以直接忽略这五种商品的存在，将五种单品进行剔除。

表 2 未出现商品列表

单品编码	单品名称	分类编码	分类名称
102900005116776	本地菠菜	1011010101	花叶类
102900005116042	藕	1011010402	水生根茎类
102900011023648	芜湖青椒(2)	1011010504	辣椒类
102900011032145	芜湖青椒(份)	1011010504	辣椒类
102900011011782	虫草花(盒)(1)	1011010801	食用菌

5.1.2 退货单

观察数据可得，附件 2 的商品销售流水中存在 461 条退货单。除了个别情况，这些退货单都能在当日或是日前发现蔬菜品类、重量和销售价格相符合的销售单。退货单以及对应的无效销售单应当视为异常数据 922 条，占总数据 0.105%，对此进行直接删除处理，对于整体数据影响甚微。在处理过程中发现 2021-02-10 当日的退货单（4.128kg 大白菜）在前一周内都无对应的销售数据，因此直接将该异常数据手动删除。此操作后，将数据重新导出保存为文件 output_name.xlsx。

5.1.3 缺失数据

在利用 EXCEL 对数据进行预览计数时发现附件 2 和附件 3 中存在不同数量的天数缺失，附件 2 中缺失 2021.2.11、2021.2.12、2022.1.31、2022.11.02、2022.11.04、2022.11.30、2022.12.01、2022.12.02、2022.12.03、2023.1.21 共计十天的流水数据，

附件 3 中缺失 2021.2.11、2021.2.12、2022.1.31、2023.1.21 共计四天的批发价格数据，结合实际分析为因疫情、春节等实际因素导致商超未正常经营。由于附件 2 总数据量为 1085 天，附件 3 总数据量为 1091 天，缺失天数均小于 1%，故忽略不计。

5.1.4 归类保存

使用 VLOOKUP 函数，参考附件 1 中对于特定编号的蔬菜品类划分，将 output_name.xlsx 中蔬菜编号所属的品类信息导入。使用数据透视表功能，以日期为行，蔬菜品类为列，按蔬菜品类求和每日销量，按日期升序排列，获得统计每日六个品类销量的数据。该数据中存在当天未售卖该品类的空缺值。在空缺值填入 0，其中花菜类 1 个空缺值，茄类 35 个空缺值。将数据导出，保存为 sum_by_day.xlsx。

将附件 3 的数据按照日期进行升序排列，参考附件 1 数据，使用 VLOOKUP 函数将附件 3、附件 4 中各单品与所属品类对应，分别导出保存为 new_3.xlsx 和 new_4.xlsx。

5.1.5 标准化

在下面模型的运行时，会有对上述所提到文件数据的进一步处理，比如标准化、归一化等，将在下文具体阐释。

5.2 问题一模型的建立与求解

首先，利用 MATLAB 软件对所得数据进行描述性统计，结果如下：

表 3 蔬菜品类描述性统计结果

	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类	总计
MIN	0	31.298	6.066	0	3.012	0.926	64.243
MAX	186.155	1265.473	604.231	118.931	511.136	296.792	2483.88
MEAN	38.49442488	182.9690627	84.41348295	20.67445346	70.12601382	37.40174931	434.0791871
MEDIAN	34.072	173.192	72.925	18.302	57.535	30.194	397.806
SKEWNESS	1.520410293	2.897314128	3.142225134	1.586211994	2.991960274	2.504537717	3.039995064
KURTOSIS	7.166275779	29.07531282	21.33636433	8.284536515	19.80912988	15.09938151	22.79285305
STD	22.69488043	86.2137504	53.43388988	13.48425691	48.49257232	31.35807215	208.5133041

六种品类销量中，最小值最低的是花菜类和茄类，销量最小值为 0，最大值最高的是花叶类，销量最大值为 1265.473 千克，除花叶类蔬菜商品销量均值明显高于其他五种品类蔬菜商品的均值外，整体较为平稳。六种品类的销量平均数与中位数相近，数据分布相对平稳。结合均值和两个极值可以看出，销量最大值相对均值有较大偏离。结合偏度和峰度可知，数据整体左偏且高峰，与最大值相对均值偏离的结论相符。

画出六种品类时间分布的柱形图和百分比示意图，如下：

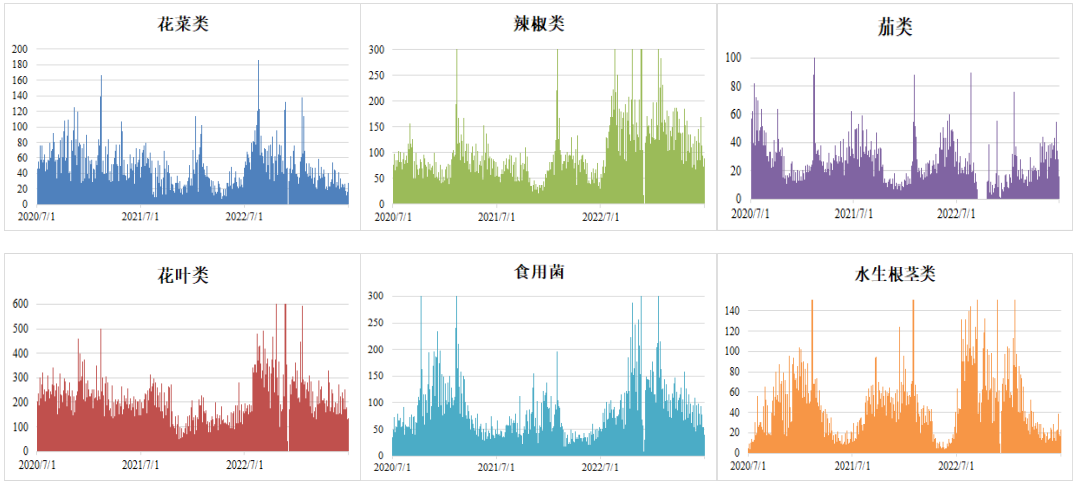


图 1 蔬菜各品类销售量随时间分布

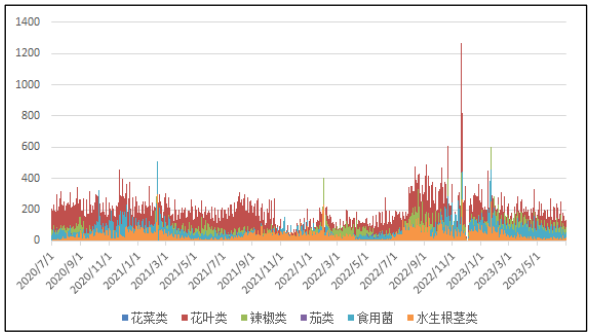


图 2 蔬菜各品类销售量随时间分布

其中，辣椒类、茄类、食用菌、水生根菌类的销量随时间季度变化呈现较为明显的周期性特征；从整体销量上来看，花叶类的平均销量最大，茄类的平均销量最小。而且，蔬菜的整体销量变化曲线，与蔬菜的成熟期也相互吻合。各种品类的蔬菜商品在每年春节及各种节假日前后均有销售峰值出现，符合人们的日常生活习惯。在一周中的某些特定日子，蔬菜销售量可能会有所不同，例如，周末（周六和周日）人们可能更多地购买蔬菜，因为他们有更多时间准备食物。中间销售量的突然增加推测可能与新产品推出或促销活动有关，人们可能会被特价或新品吸引而购买更多蔬菜。

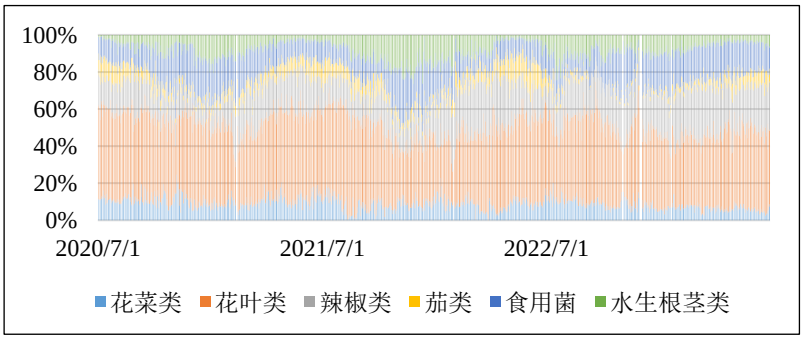
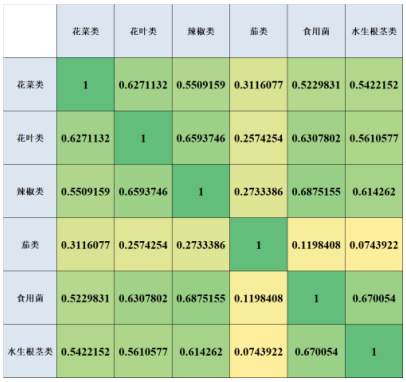


图 3 蔬菜各品类销售量占比随时间分布

由图可知，六种品类的蔬菜商品中，花叶类占总蔬菜商品销售量百分比最大，且整体较为平稳，茄类占比最小，且随季节周期变化，销量波动较大，与前文的结论相吻合。

对六种品类的蔬菜销量进行皮尔逊相关系数分析，获得了六种品类的相关关系，双尾检验均在$\alpha=0.05$的显著性水平下显著，下图为该商超经销的六种蔬菜品类每日销量的皮尔逊相关系数矩阵。



	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
花菜类	1	0.6271132	0.5509159	0.3116077	0.5229831	0.5422152
花叶类	0.6271132	1	0.6593746	0.2574254	0.6307802	0.5610577
辣椒类	0.5509159	0.6593746	1	0.2733386	0.6875155	0.614262
茄类	0.3116077	0.2574254	0.2733386	1	0.1198408	0.0743922
食用菌	0.5229831	0.6307802	0.6875155	0.1198408	1	0.670054
水生根茎类	0.5422152	0.5610577	0.614262	0.0743922	0.670054	1

图 4 蔬菜各品类销售量皮尔逊相关性矩阵热力图

其中花菜-花叶、花叶-辣椒，花叶-食用菌、辣椒-食用菌、辣椒-水生根茎、食用菌-水生根茎销量的皮尔逊相关系数均大于 0.6，存在较强的线性正相关性。较强的正相关性表明这些品类的销售量往往同向变化。而各品类之间不存在负相关的情况，表明以品类为单位时，各品类的蔬菜商品之间不存在明显的竞品关系。

使用数据进行聚类分析：

（1）从日期数据中提取该日与当年 1 月 1 日的距离，距离是[1,366]之间的整数。数据的列标签包括日期距离、246 种蔬菜单品当日对应总销量。所有数据标准化以后，在 SPSS 软件中进行 K-means 聚类，K=4，迭代次数取 100，得到的部分最终聚类中心如下（完整聚类中心见附录）：

表 4 部分蔬菜销量聚类结果（单位:天）

	夏秋 1	春季	冬季	夏秋 2
在一年中的第几天	0.6629972438	0.3995443652	0.1752588100	0.7093116005
艾蒿	0.0000000000	0.0046582076	0.0000000000	0.0000000000
白菜苔	0.0042388094	0.0224548389	0.1466367078	0.0000000000
白蒿	0.0000000000	0.0156464356	0.0000000000	0.0000000000
白玉菇(1)	0.0120120120	0.0000000000	0.0030534351	0.0000000000
白玉菇(2)	0.0942192192	0.0000000000	0.0000000000	0.0000000000
白玉菇(袋)	0.0688188188	0.0329823614	0.1078244275	0.0548523207
白玉菇(盒)	0.0000000000	0.0153587713	0.0169635284	0.0000000000
薄荷叶	0.0000000000	0.0117632412	0.0286106052	0.0060152034

保康高山大白菜	0.0000000000	0.0238923819	0.0000000000	0.2263550455
本地黄心油菜	0.0000000000	0.0026297214	0.0041019387	0.2350897635
本地上海青	0.0317248048	0.0022802478	0.0737595078	0.0007091603
冰草	0.0118631579	0.0000000000	0.0005986389	0.0000000000
菜心(份)	0.0000000000	0.0182685385	0.0003180662	0.1266701828

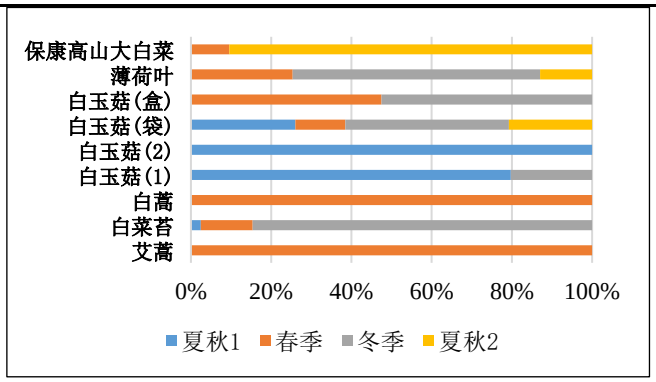


图 5 部分蔬菜销量聚类百分比（单位:天）

结果显示共有四种聚类特征，其中第二类聚类特征为春季，对应的特色蔬菜单品包括大白菜秧、艾蒿、白蒿等；第三类聚类特征为冬季，对应的特色蔬菜单品包括白菜苔、荸荠等；第一、四类聚类特征为 4-10 月份的夏秋季，对应的蔬菜单品众多，表明了单品出售在一年当中的分布规律。冬春季蔬菜品类有限，而夏秋季蔬菜品类丰富，符合现实情况。

（2）从日期数据中提取对应的小时时间，小时是[9,22]之间的整数。数据的列标签包括小时时间、246 种蔬菜单品该小时内对应总销量。所有数据标准化之后，在 SPSS 软件中进行 K-means 聚类，K=5，最大迭代次数取 100，得到的部分最终聚类中心如下（完整聚类中心见附录）：

表 5 部分蔬菜销量聚类结果（单位:小时）

	12:30	15:00	17:00	20:00	10:30
小时	0.2511554255	0.4555864847	0.6122565027	0.8275397808	0.0971476400
艾蒿	0.0000176628	0.0000905014	0.0000000000	0.0003485818	0.0009672055
白菜苔	0.0088126262	0.0000000000	0.0084960142	0.0013795523	0.0148258946
白蒿	0.0008063244	0.0005376174	0.0000000000	0.0006693016	0.0000000000
白玉菇(1)	0.0001891074	0.0000000000	0.0011941317	0.0002822467	0.0043002346
白玉菇(2)	0.0009928139	0.0000000000	0.0098942340	0.0009878634	0.0195465207
白玉菇(袋)	0.0115355522	0.0305704886	0.0285661115	0.0135991584	0.0301371810
白玉菇(盒)	0.0021747352	0.0000000000	0.0029000341	0.0013171512	0.0026062028
薄荷叶	0.0016380431	0.0020486556	0.0010437240	0.0013945371	0.0018258102

保康高山大白菜	0.0065919184	0.0694557579	0.0009254986	0.0070579752	0.0023241564
本地黄心油菜	0.0075570707	0.0650118069	0.0003612153	0.0027499455	0.0003009504
本地上海青	0.0030165602	0.0000609943	0.0063933239	0.0012129991	0.0210476520
本地小毛白菜	0.0009410025	0.0118065811	0.0000000000	0.0001042269	0.0000000000
荸荠	0.0117363492	0.0217238840	0.0190886456	0.0074534761	0.0126089458
荸荠(份)	0.0004727685	0.0000000000	0.0000000000	0.0007056167	0.0000000000

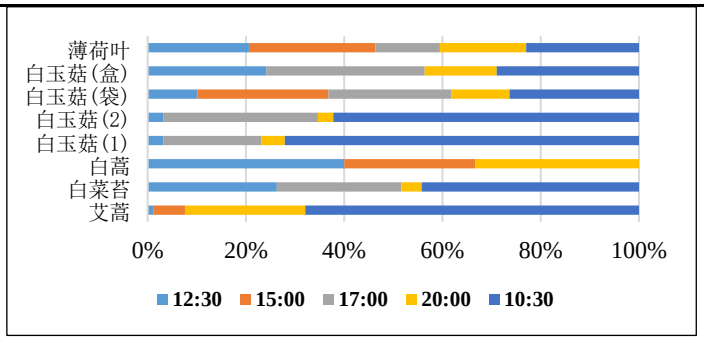


图 6 部分蔬菜销量聚类百分比（单位:小时）

其中聚类中心分别接近 10:30，12:30，15:00，17:00，20:00 点，对应上午、中午、下午、傍晚、夜晚。表明单品的出售在一天当中的分布规律，例如上午艾蒿、苋菜等蔬菜腐烂较快且上午刚刚到货较为新鲜，故而上午销量较多；一些盒装食用菌销量高峰集中于晚间，推测是因为盒装商品借助捆绑销售变相打折。

（3）借助 Python 语言进行层次聚类，列标签为 246 种蔬菜每日销量，部分结果展示如下（完整聚类层次分析谱系图见支撑材料）：

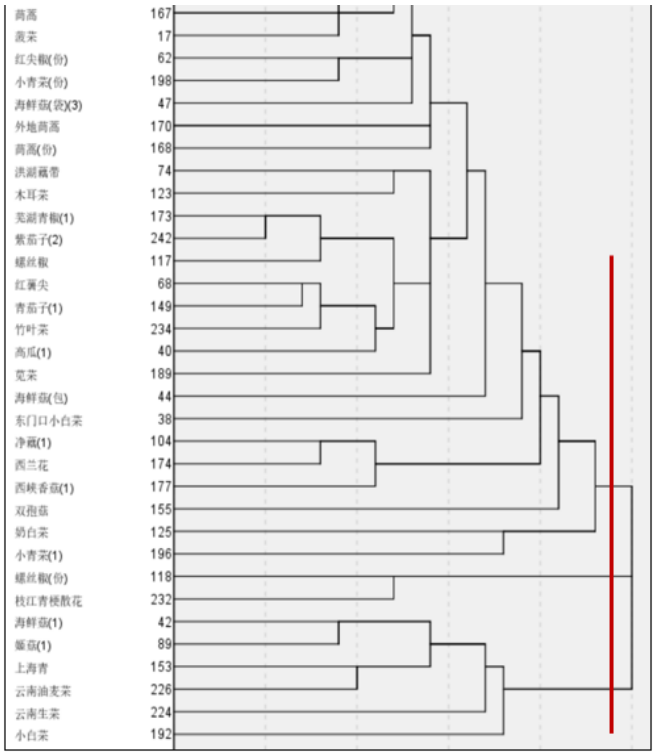


图 7 部分蔬菜销售量层次聚类谱系图

由此可以说明对于单品偏好的规律和相互关系的探究。如图所示，纵向画一条红线，可以看到红线与谱系图有三个交点，通过三个交点向左追溯，可以看到，该所有的蔬菜品类被依此分为三个大类：首先是海鲜菇(1)、上海青等蔬菜商品，是生活中极为常见的一类商品，在商超销售中往往量大价低，且通常在秋冬季集中上市销售；然后是螺丝辣椒(份)和枝江青梗散花，被分为一类，这两种单品一单的最低起售量较高，商品价格均在 7 元上下，在同品类商品的价格中接近三分之一位数，且销量较大，是本文日常生活中较为常见的配菜，但往往会搭配其他蔬菜一同购买，不单独作为主菜使用；而剩余的菜品被归为一类，观察分析可知，这一类菜品的购买频率一般较低，且购买时比较零散，菜品季节性较强。

5.3 问题二模型的建立与求解

5.3.1 模型的建立

由于商超需要在补货单品和成本未知的情况下确定补货策略，本文首先结合历史销量数据，利用 Python 编程建立 ARIMA 模型，预测未来一周每天各品类蔬菜商品的需求，并以此为补货标准，即补货量应等于预测市场需求。

首先进行数据平稳性检验和白噪声检验，由于数据平稳性良好，无需进行差分，观察各品类的自相关图和偏自相关图，最终使用的预测模型为 ARIMA(1,0,3)。

由于 ARIMA 模型预测表现欠佳，本文尝试引入 LSTM 神经网络模型来进行优化预测。使用函数创建训练数据集和标签，这个函数以一个固定的 look_back 为基础，从销售数据中创建了 X_train 和对应的 y_train。例如，如果 look_back 为 7，模型将使用过去 7 天的销售数据来预测下一天的销售数据。接着构建了一个 Sequential 模型，该模型由几个层组成：添加了一个 LSTM 层，具有 64 个神经元，激活函数为 ReLU。输入形状为(7, 6)，表示输入数据是 7 天的销售数据，每天有 6 个特征（六个品类的销售数据）；添加了一个全连接层，具有 32 个神经元，激活函数为 ReLU；添加了一个输出层，具有 6 个神经元，激活函数为线性激活函数。这是希望模型输出未来七天的销售数据，每天对应一个神经元。最后使用 model.compile 配置了模型的优化器(Adam)和损失函数。完整代码将附于附录部分。

在获得了未来一周对各品类商品的预测补货量后，以单日某蔬菜品类销售总量为被解释变量，以单日某品类正常出售的平均成本加成定价、季度、新冠肺炎疫情环境为解释变量,建立多元线性回归模型，其中，季度和新冠肺炎疫情环境为虚拟变量。对季度，用 SEASON1、SEASON2、SEASON3 表示，取值为 0/1，0 表示不在该季度，1 表示处于该季度。对新冠肺炎疫情环境，用 COVID 表示，

取值为 0/1，0 表示不处于新冠肺炎疫情封控管理中，1 表示处于新冠肺炎疫情封控管理中。回归模型如下：

$$N_i = \beta_{0i} + \beta_{1i}P_{平均i} + \beta_{2i}COVID + \beta_{3i}SEASON1 + \beta_{4i}SEASON2 + \beta_{5i}SEASON3 + u_i, \quad (i = 1,2,..,6) \quad \textbf{(1)}$$

则回归方程为：

$$N_i = \beta_{0i} + \beta_{1i}P_{平均i} + \beta_{2i}COVID + \beta_{3i}SEASON1 + \beta_{4i}SEASON2 + \beta_{5i}SEASON3, \quad (i = 1,2,..,6) \quad \textbf{(2)}$$

将 $\beta_{0i} + \beta_{2i}COVID + \beta_{3i}SEASON1 + \beta_{4i}SEASON2 + \beta_{5i}SEASON3$ 记作 α_{0i}

即上列回归方程式可化为：

$$N_i = \alpha_{0i} + \beta_{1i}P_{平均i}, \quad (i = 1,2,..,6) \quad \textbf{(3)}$$

下面，本文约定一些符号并给出计算方式：

对某品类蔬菜商品的平均单位成本，有：

$$AC_i = \frac{\sum_{j=1}^{j} P_{批发ij} m_{ij}}{m_{ij}} \quad \textbf{(4)}$$

对任一蔬菜单品的平均折扣价格和平均价格，有：

$$P_{折扣ij} = \frac{\sum_{k=1}^{k} P_{折扣ijk} m_{折扣ijk}}{\sum_{k=1}^{k} m_{折扣ijk}} \quad \textbf{(5)}$$

$$P_{平均ij} = \frac{\sum_{k=1}^{k} P_{折扣ijk} m_{折扣ijk} + P_{原价ij} \times \sum_{k=1}^{k} m_{原价ijk}}{\sum_{k=1}^{k} m_{折扣ijk} + \sum_{k=1}^{k} m_{原价ijk}} \quad \textbf{(6)}$$

$$m_{ij} = \sum_{k=1}^{k} m_{折扣ijk} + \sum_{k=1}^{k} m_{原价ijk} \quad \textbf{(7)}$$

其中，i 表示第 i 个蔬菜品类，j 表示品类中第 j 个单品，k 为某日某单品蔬菜商品同一种销售方式中的第 k 单。

同理得到某品类的平均销售价格为：

$$P_{平均i} = \frac{\sum_{j=1}^{j} P_{平均ij} m_{ij}}{\sum_{j=1}^{j} m_{ij}} \quad \textbf{(8)}$$

建立某日商超经销某个蔬菜品类的利润模型如下：

$$W_i = \begin{cases} N_i P_{平均i} - S_{补货i} AC_i, (i = 1,2,...,6 且 S_{补货i} \geq N_i, 滞销) \\ S_{补货i} P_{平均i} - S_{补货i} AC_i, (i = 1,2,...,6 且 S_{补货i} < N_i, 脱销) \end{cases} \quad (9)$$

其中，W_i 为某日商超经销某品类蔬菜商品的总利润，AC_i 为某日某种品类蔬菜商品的平均单位价格。

(3)代入**(9)**，得到

$$W_i = \begin{cases} \beta_{1i} P_{平均i}{}^2 + \alpha_{0i} P_{平均i} - S_{补货i} AC_i, (i = 1,2,...,6 且 P_{平均i} \geq \dfrac{S_{补货i} - \alpha_{0i}}{\beta_{1i}}) \\ S_{补货i} P_{平均i} - S_{补货i} AC_i, (i = 1,2,...,6 且 P_{平均i} < \dfrac{S_{补货i} - \alpha_{0i}}{\beta_{1i}}) \end{cases} \quad (10)$$

在 $\beta_{1i}<0$时 （即一般情况下需求函数的正常形态），上式的极值为

$$W_{i\max} = \begin{cases} \dfrac{-4\beta_{1i} S_{补货i} AC_i - \alpha_{0i}{}^2}{4\beta_{1i}}, S_{补货i} > \dfrac{\alpha_{0i}}{2} \\ \dfrac{S_{补货i}(S_{补货i} - \alpha_{0i} - \beta_{1i} AC_i)}{\beta_{1i}}, S_{补货i} \leq \dfrac{\alpha_{0i}}{2} \end{cases} \quad (11)$$

分别当 $P_{平均i} = -\dfrac{\alpha_{0i}}{2\beta_{1i}}$ 和 $P_{平均i} = \dfrac{S_{补货i} - \alpha_{0i}}{\beta_{1i}}$ 时取到

由成本加成定价理论，根据参考文献，可得

$$P_{原价i} = \frac{AC_i}{1 - K_i} \quad (12)$$

其中，K_i 为某品类蔬菜商品定价时确定的利润率。因为商超的流水数据显示，打折出售的商品单仅占总出售商品单不到 5%，且商超各品类蔬菜的平均出售价格与平均原价每日对比如下，偏差率极低，六个品类分别的平均偏差为-1.46%、-2.30%、-2.05%、-0.55%、0.76%、-1.22%，总平均偏差仅为 1.14%，因此可以将商超出售平均价格视作商超出售平均价格。

因此本文计算得到的平均利润率即为加成定价利润率。

$$K_i = 1 - \frac{AC_i}{P_{平均i}} \quad (13)$$

5.3.2 模型的求解

5.3.2.1 补货量决策

在使用 ARIMA 进行该题的预测时没有必要进行差分，所以模型实际上表示为 ARIMA(1,0,3)，或者说 ARMA(1,3)。在使用该模型对于六个品类的历史数据进行训练与测试，其中训练集占 80%左右，测试集占 20%，以花菜类蔬菜商品为例，获得的结果表明 R 方为-0.16。下图画出了测试集的预测值与真实值。图 9 是测试集的残差示意图，图 10 是正态性检验示意图。

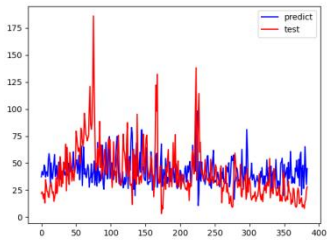


图 8 测试集的预测值与真实值

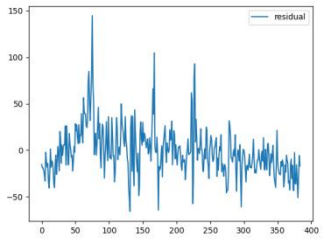


图 9 残差示意图

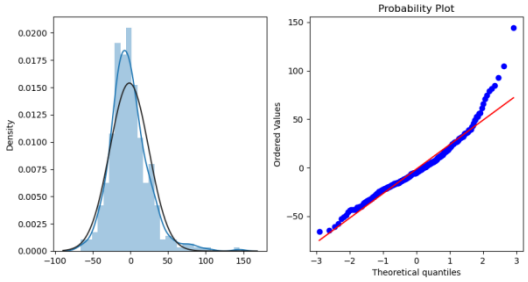


图 10 正态性检验

以花菜类为例，进行 ADF 单位根检验，得到的 P 值为 0.02446，表明花菜类序列在显著性水平α=0.05 下平稳。使用 Ljung-box 白噪声检验表明该时间序列数据非白噪声。接着画出改时间序列的自相关 ACF 图和偏自相关 PACF 图如下图所示：

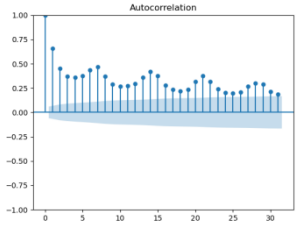


图 11 花菜类 ACF 拖尾

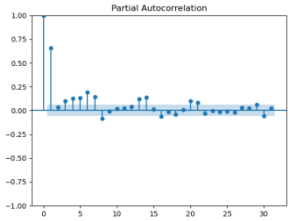


图 12 花菜类 PACF 3 阶截尾

其他品类销量的时间序列也使用与花菜类相同的处理方法。结果显示，除了花叶类的 ADF 检验 p 值略大于 0.05 外，其他五类的 ADF 检验 p 值均小于 0.05，即在α=0.05 的显著性水平下平稳；其中平稳性相对较差的花叶类对应的 ACF 与 PACF 图如下:

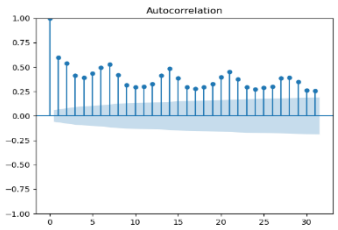


图 13 花叶类 ACF 拖尾

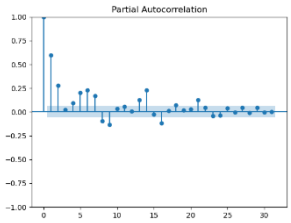


图 14 花叶类 PACF 截尾

除花菜与花叶类以外的四类品类对应的 ACF 与 PACF 图如下:

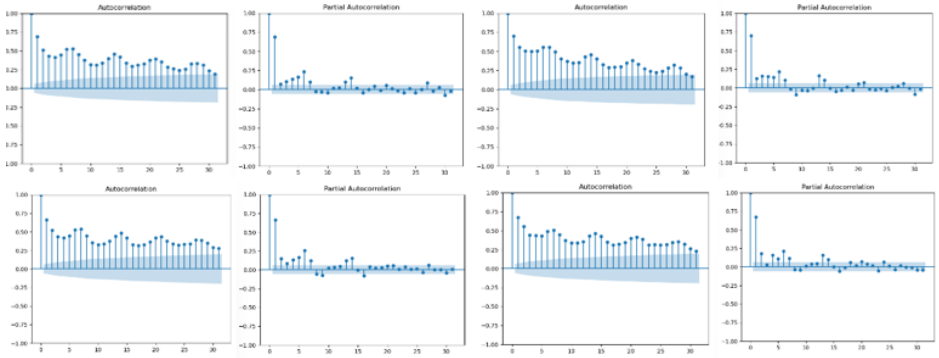


图 15 其他类 ACF 拖尾与 PACF 尾

观察到具有相似的特性，即 ACF 拖尾，PACF 3 阶截尾，为了在保证模型准确的情况下实现模型的简洁，将六个品类统一使用 ARMA (1,3)模型进行处理。

进行时间序列预测获得的 2023 年 7 月 1 日-2023 年 7 月 7 日的各品类需要的补货量。最终给出的未来一周各品类日补货总量决策如下：

表 6 蔬菜各品类补货表

日期	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023/7/1	38.27339	182.96570	84.40620	21.33766	70.10963	37.32686
2023/7/2	43.80171	196.19674	79.13343	31.25061	47.31433	15.25599
2023/7/3	42.24643	195.18954	72.23237	29.92207	52.74325	13.22388
2023/7/4	41.48861	190.73998	70.93626	32.65644	49.48240	15.42263
2023/7/5	50.05861	208.82642	80.65309	44.61479	50.17828	12.54873
2023/7/6	49.40383	215.48052	89.55030	47.66885	63.66992	10.29672
2023/7/7	42.16591	199.16509	66.07037	32.94625	48.47624	9.76040

5.3.2.2 定价决策

配置好运行环境，运行 LSTM 神经网络代码，可以得到六个蔬菜品类未来七天的预期销售情况。将已知的过去半年的销售数据与预测得到的未来七天的销售数据汇总制作得到如下的销售量随时间变化的折线图。

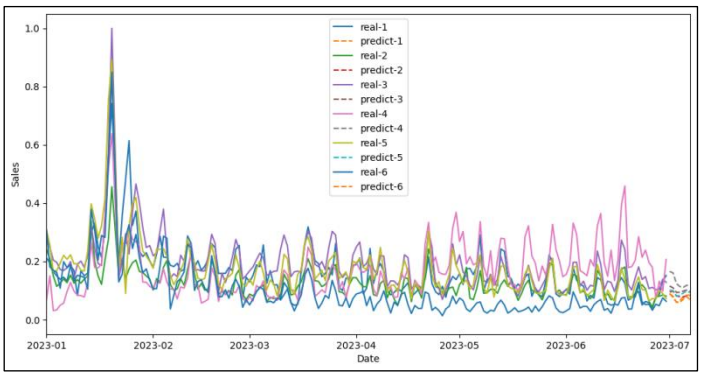


图 16 蔬菜各品类销售量时序图

使用 LSTM 模型检验得到的 R 方为 0.78，RMSE=0.63，MAE=0.14，与 ARIMA 模型比较具有较好的拟合。

根据这两个模型的比较，本文对于预测值的确定更偏向于后者。

结合附件 2、3 数据，利用 Eviews 软件，以单日某蔬菜品类销售总量为被解释变量，以单日某品类正常出售和打折出售的加权平均价格、季度、新冠肺炎疫情环境为解释变量，进行多元线性回归，使用 OLS 最小二乘法进行参数估计的结果如下：

表 7 多元线性回归结果

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/08/23 Time: 22:28
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	66.37183	3.308830	20.05900	0.0000
PRICE	-2.789647	0.256470	-10.87709	0.0000
COVID	-0.348809	1.752015	-0.199090	0.8422
SEASON1	-2.544053	1.828411	-1.391401	0.1644
SEASON2	-11.51547	1.815136	-6.344137	0.0000
SEASON3	8.621447	1.782591	4.836469	0.0000

R-squared	0.180880	Mean dependent var	38.49442
Adjusted R-squared	0.177085	S.D. dependent var	22.69488
S.E. of regression	20.58759	Akaike info criterion	8.892768
Sum squared resid	457332.8	Schwarz criterion	8.920359
Log likelihood	-4818.327	Hannan-Quinn criter.	8.903213
F-statistic	47.65362	Durbin-Watson stat	0.845272
Prob(F-statistic)	0.000000		

花菜类

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/08/23 Time: 22:33
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	310.8758	10.62836	29.24965	0.0000
PRICE	-19.10960	1.665162	-11.47612	0.0000
COVID	-26.96431	6.531307	-4.128470	0.0000
SEASON1	-14.32890	6.854996	-2.090286	0.0368
SEASON2	-42.09697	6.782478	-6.206724	0.0000
SEASON3	59.44311	6.914220	8.597226	0.0000

R-squared	0.211067	Mean dependent var	182.9691
Adjusted R-squared	0.207411	S.D. dependent var	86.21375
S.E. of regression	76.75391	Akaike info criterion	11.52460
Sum squared resid	6356564.	Schwarz criterion	11.55219
Log likelihood	-6246.096	Hannan-Quinn criter.	11.53505
F-statistic	57.73405	Durbin-Watson stat	1.035928
Prob(F-statistic)	0.000000		

花叶类

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/08/23 Time: 22:38
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	130.9266	5.304301	24.68311	0.0000
PRICE	-2.904563	0.463628	-6.264851	0.0000
COVID	-34.04311	4.404591	-7.729006	0.0000
SEASON1	31.25463	4.471434	6.989846	0.0000
SEASON2	-14.73979	4.262716	-3.457841	0.0006
SEASON3	8.867227	4.173199	2.124803	0.0338

R-squared	0.192638	Mean dependent var	84.41348
Adjusted R-squared	0.188897	S.D. dependent var	53.43389
S.E. of regression	48.12323	Akaike info criterion	10.59092
Sum squared resid	2498797.	Schwarz criterion	10.61851
Log likelihood	-5739.575	Hannan-Quinn criter.	10.60137
F-statistic	51.49038	Durbin-Watson stat	0.773134
Prob(F-statistic)	0.000000		

辣椒类

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/08/23 Time: 22:42
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	4.456221	1.758568	2.534006	0.0114
PRICE	0.465353	0.152817	3.045164	0.0024
COVID	3.969914	1.019279	3.894826	0.0001
SEASON1	7.221291	1.174231	6.149807	0.0000
SEASON2	14.50813	1.074314	13.50454	0.0000
SEASON3	14.18199	1.054340	13.45105	0.0000

R-squared	0.192418	Mean dependent var	20.67445
Adjusted R-squared	0.188676	S.D. dependent var	13.48426
S.E. of regression	12.14574	Akaike info criterion	7.837349
Sum squared resid	159173.1	Schwarz criterion	7.864940
Log likelihood	-4245.762	Hannan-Quinn criter.	7.847794
F-statistic	51.41756	Durbin-Watson stat	0.756458
Prob(F-statistic)	0.000000		

茄类

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/08/23 Time: 22:44
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	136.6781	5.375085	25.42808	0.0000
PRICE	-2.201562	0.639757	-3.441248	0.0006
COVID	-28.15988	3.863792	-7.288145	0.0000
SEASON1	-12.82836	3.799765	-3.376093	0.0008
SEASON2	-53.89559	3.753988	-14.35689	0.0000
SEASON3	-31.94510	3.921893	-8.145327	0.0000

R-squared	0.251689	Mean dependent var	70.12601
Adjusted R-squared	0.248222	S.D. dependent var	48.49257
S.E. of regression	42.04556	Akaike info criterion	10.32090
Sum squared resid	1907487.	Schwarz criterion	10.34849
Log likelihood	-5593.088	Hannan-Quinn criter.	10.33134
F-statistic	72.58292	Durbin-Watson stat	0.878760
Prob(F-statistic)	0.000000		

食用菌

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/08/23 Time: 22:46
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	60.79721	3.300756	18.41918	0.0000
PRICE	-1.302308	0.301176	-4.324081	0.0000
COVID	-1.812688	2.316795	-0.782412	0.4341
SEASON1	-0.694528	2.486485	-0.279321	0.7801
SEASON2	-29.84209	3.026422	-9.860518	0.0000
SEASON3	-6.204623	2.482389	-2.499457	0.0126

R-squared	0.236222	Mean dependent var	37.40175
Adjusted R-squared	0.232683	S.D. dependent var	31.35807
S.E. of regression	27.46862	Akaike info criterion	9.469480
Sum squared resid	814132.6	Schwarz criterion	9.497071
Log likelihood	-5131.193	Hannan-Quinn criter.	9.479925
F-statistic	66.74283	Durbin-Watson stat	0.842603
Prob(F-statistic)	0.000000		

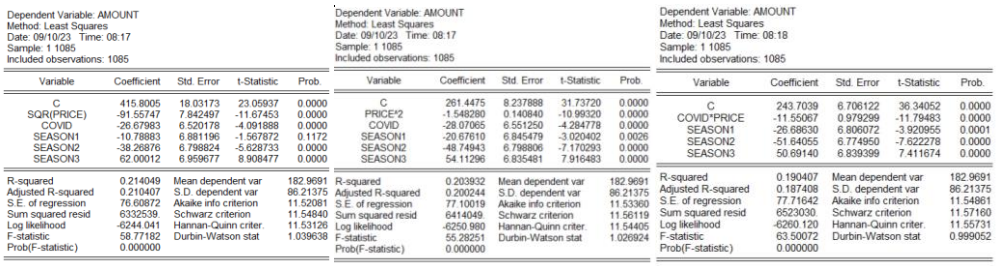
水生根茎类

由于需求函数除常规的线性以外，还存在价格平方、价格平方根等常用需求函数。因此更改模型设定，分别使用价格的平方与价格的平方根建立回归模型；或者改变虚拟变量的位置，以乘价格的设定出现，最后得到的回归结果，其 R 方与原模型设定相近，对改进原模型帮助不大，因此选择价格的一次项模型作为最

终的多元线性回归模型选择。

以花叶类为例，不同模型假定下的多元线性回归结果如下，对于模型改良帮助不大。

表 8 花叶类多模型多元线性回归结果



Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/10/23 Time: 08:17
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	415.8005	18.03173	23.05937	0.0000
SQR(PRICE)	-91.55747	7.842497	-11.67453	0.0000
COVID	-26.67983	6.520178	-4.091888	0.0000
SEASON1	-10.78883	6.881196	-1.567872	0.1172
SEASON2	-38.26876	6.798824	-5.628733	0.0000
SEASON3	62.00012	6.959677	8.908477	0.0000

R-squared	0.214049	Mean dependent var	182.9691
Adjusted R-squared	0.210407	S.D. dependent var	86.21375
S.E. of regression	76.60872	Akaike info criterion	11.52081
Sum squared resid	6332539.	Schwarz criterion	11.54840
Log likelihood	-6244.041	Hannan-Quinn criter.	11.53126
F-statistic	58.77182	Durbin-Watson stat	1.039638
Prob(F-statistic)	0.000000		

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/10/23 Time: 08:17
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	261.4475	8.237888	31.73720	0.0000
PRICE^2	-1.548280	0.140840	-10.99320	0.0000
COVID	-28.07065	6.551250	-4.284778	0.0000
SEASON1	-20.67610	6.845479	-3.020402	0.0026
SEASON2	-48.74943	6.798806	-7.170293	0.0000
SEASON3	54.11296	6.835481	7.916483	0.0000

R-squared	0.203932	Mean dependent var	182.9691
Adjusted R-squared	0.200244	S.D. dependent var	86.21375
S.E. of regression	77.10019	Akaike info criterion	11.53360
Sum squared resid	6414049.	Schwarz criterion	11.56119
Log likelihood	-6250.980	Hannan-Quinn criter.	11.54405
F-statistic	55.28251	Durbin-Watson stat	1.026924
Prob(F-statistic)	0.000000		

Dependent Variable: AMOUNT
Method: Least Squares
Date: 09/10/23 Time: 08:18
Sample: 1 1085
Included observations: 1085

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	243.7039	6.706122	36.34052	0.0000
COVID*PRICE	-11.55067	0.979299	-11.79483	0.0000
SEASON1	-26.68630	6.806072	-3.920955	0.0001
SEASON2	-51.64055	6.774950	-7.622278	0.0000
SEASON3	50.69140	6.839399	7.411674	0.0000

R-squared	0.190407	Mean dependent var	182.9691
Adjusted R-squared	0.187408	S.D. dependent var	86.21375
S.E. of regression	77.71642	Akaike info criterion	11.54861
Sum squared resid	6523030.	Schwarz criterion	11.57160
Log likelihood	-6260.120	Hannan-Quinn criter.	11.55731
F-statistic	63.50072	Durbin-Watson stat	0.999052
Prob(F-statistic)	0.000000		

最终得到：

表 9 蔬菜品类模型多元线性回归结果

模型名称	多元线性回归结果
花菜类模型	N=66.37183+8.621447-2.789647P
花叶类模型	N=310.8758+59.44311-19.10960P
辣椒类模型	N=130.9266+8.867227-2.904563P
茄类模型	N=4.456221+14.18199+0.465353P
食用菌模型	N=136.6781-31.9451-2.201562P
水生根茎类模型	N=60.79721-6.204623-1.302308P

对于模型价格这一项系数的 t-检验均能表明，销售量受到商品价格关系显著。除了茄类外的五类蔬菜商品，商品价格对应的参数估计都为负，这符合实际中需求随价格变动的情况。

而茄类商品价格前的参数为正，表明模型认定茄类近似吉芬商品，即销量随着价格的提升而提升的商品，或者销量基本不受到价格影响的商品。

另外，季节对于不同品类的蔬菜销量均有显著影响，这是供给与需求两方面共同被季节因素影响造成的。同时可以发现花叶类、辣椒类、茄类和食用菌的销售量与疫情封控的关系，即疫情封控会降低这四种品类的需求，例如，在其他条件都相同的情况下，存在疫情封控时的花叶类销量平均比无疫情封控低 26.96 千克。

根据历史三年数据确定六种品类的进货平均成本 AC 如下：花菜类 6.217734345 元/kg，花叶类 3.38537272 元/kg，辣椒类 5.549386229 元/kg，茄类 5.439537992 元/kg，食用菌 5.414779469 元/kg，水生根茎类 6.62328737 元/kg。

使用 ARMA 的预测结果利用前文推导的经济学模型得到各品类未来一周每日的最大利润如下：

表 10 各蔬菜品类最大利润 ARMA 预测

日期	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	266.0319	1174.4089	1213.6331	-302.6904	865.9649	324.9021
2023-07-02	231.6582	1129.8684	1242.8937	-356.6122	977.8026	359.7669
2023-07-03	241.3286	1133.2782	1281.1903	-349.3856	960.0000	332.4801
2023-07-04	246.0405	1148.3417	1288.3830	-364.2593	973.8785	361.7231
2023-07-05	192.7546	1087.1118	1234.4605	-429.3072	971.7138	322.0107
2023-07-06	196.8258	1064.5850	1185.0865	-445.9199	900.8345	282.0278
2023-07-07	241.8292	1119.8193	1310.3445	-365.8357	976.2306	271.3577

上表中茄类最大利润出现负值，是因为茄类属于**吉芬商品**，吉芬商品的出现会导致模型求解的利润函数单调，故不能用普通商品计算利润极值的方式进行计算，应以其销量最大值为约束上限，确定利润最大值。这里将茄类商品的历史平均销量作为日销量，并带入计算的得到定价策略。

表 11 各蔬菜品类最大利润 ARMA 预测修正

日期	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	266.0319	1174.4089	1213.6331	777.4969	865.9649	324.9021
2023-07-02	231.6582	1129.8684	1242.8937	723.575	977.8026	359.7669
2023-07-03	241.3286	1133.2782	1281.1903	730.8017	960.0000	332.4801
2023-07-04	246.0405	1148.3417	1288.3830	715.928	973.8785	361.7231
2023-07-05	192.7546	1087.1118	1234.4605	650.8801	971.7138	322.0107
2023-07-06	196.8258	1064.5850	1185.0865	634.2674	900.8345	282.0278
2023-07-07	241.8292	1119.8193	1310.3445	714.3515	976.2306	271.3577

此时根据前文公式**(10)**，

$$W_i = \begin{cases} \beta_{1i} P_{平均i}{}^2 + \alpha_{0i} P_{平均i} - S_{补货i} AC_i, (i = 1,2,...,6 且 P_{平均i} \geq \dfrac{S_{补货i} - \alpha_{0i}}{\beta_{1i}}) \\ S_{补货i} P_{平均i} - S_{补货i} AC_i, (i = 1,2,...,6 且 P_{平均i} < \dfrac{S_{补货i} - \alpha_{0i}}{\beta_{1i}}) \end{cases} \quad \textbf{(10)}$$

此处茄子品类的 $\beta_{1茄类}$ 和 $\alpha_{0茄类}$ 均为正数，其利润方程单调递增，因此考虑以历史平均销量为上限确定 7 月 1 日预计利润率。

补货量或销售量即需求量在正常条件下应该与价格负相关。除茄类以外的五种品类的回归结果均符合条件。只有茄类模型出现吉芬商品的特征，对应的平均定价策略为：

表 12 各蔬菜品类的平均定价 ARMA 策略

	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	13.4414	9.8041	24.0645	28.1532	23.7861	20.9599
2023-07-02	13.4414	9.6893	24.0645	28.1532	26.0809	30.2053
2023-07-03	13.4414	9.6893	24.0645	28.1532	23.7861	31.7657

2023-07-04	13.4414	9.6893	24.0645	28.1532	25.0961	30.0773
2023-07-05	13.4414	9.6893	24.0645	28.1532	24.7800	32.2841
2023-07-06	13.4414	9.6893	24.0645	28.1532	23.7861	34.0134
2023-07-07	13.4414	9.6893	25.3819	28.1532	25.5531	34.4252
average	6.2177	3.3854	5.5494	28.1532	5.4148	6.6233

解得最终每日各品类的成本加成定价利润率如下：

表 13 各蔬菜品类的 ARMA 成本加成定价利润率

日期	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	0.5374	0.6547	0.7694	0.8068	0.7724	0.6840
2023-07-02	0.5374	0.6506	0.7694	0.8068	0.7924	0.7807
2023-07-03	0.5374	0.6506	0.7694	0.8068	0.7724	0.7915
2023-07-04	0.5374	0.6506	0.7694	0.8068	0.7842	0.7798
2023-07-05	0.5374	0.6506	0.7694	0.8068	0.7815	0.7948
2023-07-06	0.5374	0.6506	0.7694	0.8068	0.7724	0.8053
2023-07-07	0.5374	0.6506	0.7814	0.8068	0.7881	0.8076

使用 LSTM 的预测结果解得各品类未来一周每日的最大利润如下：

表 14 各蔬菜品类最大利润 LSTM 预测

	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	256.8090	1201.7468	1178.5730	796.2434	923.1984	403.8821
2023-07-02	248.6517	1169.6647	1176.4962	802.0809	946.0034	401.8559
2023-07-03	198.6823	1176.3111	1219.2843	818.3924	975.7527	388.8264
2023-07-04	157.5701	1174.4729	1230.7051	824.0556	972.8547	391.1303
2023-07-05	131.3839	1168.3571	1232.7798	824.6914	965.2745	401.9082
2023-07-06	149.4550	1114.2569	1197.0495	816.4946	929.2859	405.4281
2023-07-07	139.5395	1054.2595	1154.0080	813.4538	902.6769	402.6246

对应的平均定价策略为：

表 15 各蔬菜品类的平均定价 LSTM 策略

	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	20.3846	10.3529	24.0645	28.1532	23.7861	23.1164
2023-07-02	20.7486	9.7273	24.0645	28.1532	23.7861	25.9716
2023-07-03	22.5117	9.8360	24.0645	28.1532	25.4440	27.8626
2023-07-04	23.6408	9.8052	24.0645	28.1532	24.9386	27.6071
2023-07-05	24.2744	9.7068	24.0645	28.1532	24.0726	25.9598
2023-07-06	23.8431	9.6893	24.0645	28.1532	23.7861	24.6553
2023-07-07	24.0828	9.6893	24.0645	28.1532	23.7861	22.7550
average	6.2177	3.3854	5.5494	28.1532	5.4148	6.6233

解得最终每日各品类的成本加成定价利润率如下：

表 16 各蔬菜品类的 LSTM 成本加成定价利润率

	花菜类	花叶类	辣椒类	茄类	食用菌	水生根茎类
2023-07-01	0.6950	0.6730	0.7694	0.8068	0.7724	0.7135
2023-07-02	0.7003	0.6520	0.7694	0.8068	0.7724	0.7450

2023-07-03	0.7238	0.6558	0.7694	0.8068	0.7872	0.7623
2023-07-04	0.7370	0.6547	0.7694	0.8068	0.7829	0.7601
2023-07-05	0.7439	0.6512	0.7694	0.8068	0.7751	0.7449
2023-07-06	0.7392	0.6506	0.7694	0.8068	0.7724	0.7314
2023-07-07	0.7418	0.6506	0.7694	0.8068	0.7724	0.7089

以上表结果为成本加成定价利润率，结合当日实际商品单位成本，根据成本加成定价公式即可确认当日商超定价。

5.4 问题三模型的建立与求解

5.4.1 模型的建立

此处在问题二模型求解时已经处理好的数据的基础上进行二次处理，通过筛选 2023-06-24 至 2023-06-30 七天的营业流水，构建出所有商品的七天销售情况 EXCEL 表，将其作为神经网络的训练数据集，并挑选出应季可售商品六种品类共计 49 种单品。对销售数据进行归一化，将数据缩放到 0 到 1 之间。然后构建训练数据集和标签，用于定义用于预测的历史时间步数。最后，使用已创建的训练数据集和标签，将数据分为训练集（80%）和测试集（20%）。

选择使用预测 LSTM 神经网络模型。首先创建一个顺序模型(Sequential)，它是 Keras 模型的一种类型，按照顺序堆叠神经网络层。添加一个 LSTM 层，具有 64 个神经元(units)，激活函数为 ReLU，并指定输入形状为(look_back, 50)。添加一个全连接(Dense)层，具有 32 个神经元，激活函数为 ReLU。添加一个输出层，具有 50 个神经元，激活函数为线性。使用 adam 优化器和均方误差(mean squared error)损失函数来编译模型。使用 model.fit 方法训练模型。指定训练数据集 X_train 和标签 y_train，并设置训练参数，如迭代次数(epochs)和批量大小(batch_size)。训练模型时，模型将学习历史销售数据的模式，以便进行未来销售的预测。

得到各个单品 2023 年 7 月 1 日这一天的预期销量后，结合问题二中已求解的品类需求函数，考虑每种蔬菜品类的需求量、超市蔬菜类商品的销售空间等条件，制定 7 月 1 日的单日补货量和定价策略。

设第 i 种品类中第 m 种单品的成本加成定价利润为K_{im}，设第 i 种品类中第 m 种单品的补货量为S_{im}，根据 2023.06.24-2023.06.30 的七天历史数据，计算出第 i 种品类中第 m 种单品的近期平均批发价格AC_{im}并以此作为 2023.07.01 的预测批发价格。

由成本加成定价方法，结合参考文献可知，

$$P_{im} = \frac{AC_{im}}{1 - K_{im}} \tag{14}$$

根据问题二中的多元线性回归模型得到单品的平均销量为

$$N_{im}=\alpha_{0i}+\beta_{1i}P_{im} \tag{15}$$

由此得到单品的利润方程式如下：

$$W_{im}=\begin{cases} N_{im}P_{im}-S_{im}AC_{im},(S_{im}\geq N_{im},\text{滞销}) \\ S_{im}P_{im}-S_{im}AC_{im},(S_{im}<N_{im},\text{脱销}) \end{cases} \tag{16}$$

$$W_{im}=\begin{cases} AC_{im}[\frac{\beta_{1i}}{(1-K_{im})^2}+\frac{\alpha_{0i}}{1-K_{im}}-S_{im}],(S_{im}\geq \alpha_{0i}+\frac{\beta_{1i}AC_{im}}{1-K_{im}}) \\ S_{im}AC_{im}(\frac{K_{im}}{1-K_{im}}),(S_{im}<\alpha_{0i}+\frac{\beta_{1i}AC_{im}}{1-K_{im}}) \end{cases} \tag{17}$$

对各单品利润求和得，

定义 COUNT 函数：

若集合*{x}*中等于 0 的数有 *t* 个，则 *COUNT{x}=t* **(18)**

则约束条件为

$$S_{im}\geq 2.5\text{或}S_{im}=0 \tag{19}$$

$$\sum_{m=1}^{m}S_{im}\geq S_{\text{补货}i},(S_{\text{补货}i}\text{为问题二中预测的品类补货总量}) \tag{20}$$

$$16\leq COUNT\{x\}\leq 22 \tag{21}$$

$$0\leq K_{im}<1 \tag{22}$$

5.4.2 模型的求解

此处在问题二模型求解时已经处理好的数据的基础上进行二次处理，通过筛选 2023-06-24 至 2023-06-30 七天的营业流水，构建出所有商品的七天销售情况 EXCEL 表，将其作为神经网络的训练数据集。之后本文选择包含了销售数据的第二列到第五十列的数据，使用 MinMaxScaler 对销售数据进行归一化，将数据缩放到 0 到 1 之间。然后使用 create_dataset 函数构建训练数据集和标签。该函数接受数据和 look_back 参数，用于定义用于预测的历史时间步数。对于每个时间步，从历史数据中提取 look_back 个时间步的数据作为输入，并提取下一个时间步的数据。使用已创建的训练数据集和标签，将数据分为训练集（80%）和测试集（20%）。

使用 Python 编程进行代码的运行求解，得到 2023 年 6 月 24-30 日可售蔬菜品种的 7 月 1 日预期销量，如下图所示：

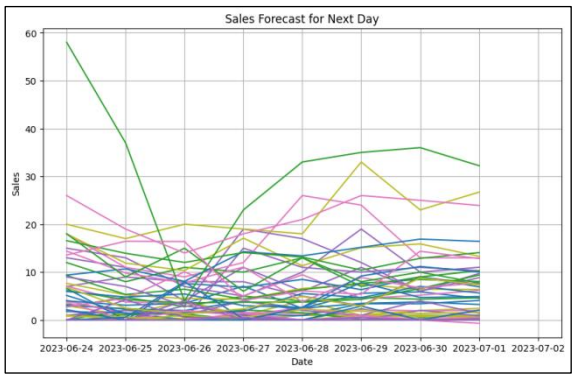


图 17 蔬菜品种销量预测图

使用 Python 语言，用遗传算法求目标函数的最大值以及对应的变量条件。为保证最终结果能成功收敛，通过设定基因个数为 96，变异概率 0.2，交叉概率 0.7，选择最大迭代次数 10000000 次，使用上面 LSTM 预测给出的预计值作为参数 S 的初始值，取第二问中得到的当天各品类利润的平均值作为参数 K。最终计算选出 30 种单品，给出的最大总利润约为 **976.532119 元**，其定价与对应的补货量序列与单品成本价加成定价利润率如下表所示，仅以此计算各单品的成本加成定价利润率并作为商超 2023 年 7 月 1 日的定价策略。

表 17　7 月 1 日单品补货量和定价策略

单品	补货量(kg)	定价（元）	平均批发成本（元/kg）	总利润（元）	成本加成定价利润率
苋菜	5.63	3.16	2.324285714	4.705071429	26.45%
云南生菜	9.02	5.58	5.738	-1.42516	-2.83%
竹叶菜	8.33	2.32	2.318571429	0.0119	0.06%
菜心	3.74	4.61	4.6075	0.00935	0.05%
木耳菜	5.52	3.21	3.211428571	-0.007885714	-0.04%
娃娃菜	53.3	4.73	4.731428571	-0.076142857	-0.03%
红薯尖	12.58	7.44	3.204285714	53.28528571	56.93%
外地茼蒿	2.99	13.2	8.593333333	13.77393333	34.90%
奶白菜	8.27	5.2	2.528333333	22.09468333	51.38%
小青菜(1)	9.21	6.09	2.83	30.0246	53.53%
云南生菜(份)	42.91	6.17	3.604285714	110.0948	41.58%
云南油麦菜(份)	23.6	2.46	2.864285714	-9.541142857	-16.43%
菠菜(份)	19.02	2.97	4.098	-21.45456	-37.98%
西兰花	17.69	13.28	7.824285714	96.51158571	41.08%
枝江青梗散花	14.74	9.39	9.462	-1.06128	-0.77%
净藕(1)	28.42	10.48	10.74571429	-7.5516	-2.54%
高瓜(1)	2.88	11.54	11.54	-5.12E-15	0.00%
紫茄子(2)	12.64	4.59	3.747142857	10.65371429	18.36%
青茄子(1)	4.25	6	4.038571429	8.336071429	32.69%
长线茄	5.57	6.98	6.982857143	-0.015914286	-0.04%
小米椒(份)	35.44	9.04	2.135714286	244.6878857	76.37%

小皱皮(份)	14.69	5.67	1.542857143	60.62772857	72.79%
青线椒(份)	5.32	4.83	2.733333333	11.15426667	43.41%
螺丝椒(份)	27.81	5.99	3.277142857	75.44455714	45.29%
姜蒜小米椒组合装(小份)	7.5	4.8	2.437142857	17.72142857	49.23%
红椒(2)	2.9	21.65	12.89714286	25.38328571	40.43%
虫草花(份)	4.78	4.4	2.657142857	8.330857143	39.61%
双孢菇(盒)	16.74	5.68	3.402857143	38.11937143	40.09%
金针菇(盒)	71	3.92	1.452857143	175.1671429	62.94%
海鲜菇(包)	6.28	3.79	1.954285714	11.52828571	48.44%
合计				976.532119	

5.5 问题四的求解

为了更好地制定蔬菜商品的补货和定价决策，本文给出以下建议：

5.5.1 考虑库存情况

（1）库存品可以高效调节商超的蔬菜供应，在需求旺盛时可以调用库存，充分实现利润，而在需求冷淡时，也可以实现蔬菜商品的保鲜储存，减少因变质带来的损失；

（2）在存有库存的条件下，还需要考虑库存管理，如库存容积、当日库存量的大小以及可售卖天数多少，通过建立模型规划库存模式，实现库存的稳定与平衡；

（3）另外如果还需要冷藏库保鲜，冷藏的成本。

5.5.2 参考市场形式

（1）在补货时应当关注蔬菜是否有新品种上市，并积极调查新品种的优缺点；

（2）参考市场中新的流行趋势，并关注市场的价格波动等。

5.5.3 考虑依据节庆假日、天气情况等各种“旺日”调整下单

（1）前文的图中可以观察到日销售总量的峰值，集中在春节、国庆等节假日附近，在假日能够实现的利润可能远超平日数倍。因此要针对性考虑和规划假日期间的补货和定价策略，考虑是否到假日人流高峰日，是否某一节庆高峰日，并有特殊品项需求（如端午节售卖粽叶）等等；

（2）考虑第二天的天气状况对人流量的影响，在天气状况较为糟糕时，线下商超往往趋于无人问津，人们青睐使用生鲜快运等方式，足不出户购买蔬菜。

5.5.4 依据历史销售记录及顾客消费习惯订货

（1）特别是在春节等这样的重大节日及销售旺季，需要保留以往的销售记录，作为参考；

（2）参考日均销售量及商品周转率，通过使用以往销量的相关数据如 DMS、

月销售量、商品周转率等指标来作为订货参考值；

（3）积极把握当地顾客的生活规律与消费习惯。

5.5.5 考虑季节变化

蔬菜的季节性体现最强，从夏至秋都有当年应季品项上市，蔬菜在应季时订购成本即批发价格会下降，且蔬菜品质会提高，销量会提升，考虑可以在旺季加大订货量，增大陈列面积与陈列量。

5.5.6 考虑批发蔬菜的品质

首先订购蔬菜最基本的要求是必须满足健康、安全、卫生的前提，打造良好口碑。

5.5.7 考虑营销策略

（1）在营销时可以考虑不同程度的价格歧视。例如同种蔬菜可以分为精品和普通品质，对于不同档次的蔬菜实现不同利润率的成本加成定价；

（2）在营销时可以通过发行优惠券等方法来调节需求，考虑促销期及折扣期；

（3）商超通过开展其他营销活动，分配各时间段之间的需求，实现利润的增盈。

5.5.8 考虑基本陈列面

参考商品的基本陈列面乘以补货次数，即是大致订货量。

5.5.9 考虑永续订单

依据蔬菜当季的商品组合与供货商建立永续订单，以永续订单为订货下单的依据。

5.5.10 考虑商超其他营运成本

补货成本仅是商超运营可变成本的一部分，除此之外，商超运营过程中还存在着诸多成本，如人力成本、水电成本等，可以考虑通过引进自动收银系统等新技术，以调控成本。

六、模型评价与推广

6.1 模型的优点

（1）使用 ARIMA 与 LSTM 时间序列预测相互印证；在推导最大利润时经济学模型与数学模型相互结合；

（2）聚类时使用 K-means 聚类与层次聚类，从不同角度研究问题；

（3）在第三问中先使用 LSTM 预测 7 月 1 日销量，对应的在解决规划问题时将该销量作为最初传入的参数，可以优化遗传算法，加快收敛速度，因为使用

神经网络预测的销量属于根据以往规律所得的较优解。

6.2 模型的缺点

（1）六种品类的相关性分析仅用皮尔逊相关系数，仅能衡量线性关系；

（2）数据量过大，每次运行需要较长时间。

6.3 模型的展望

对于问题二，由于茄类商品的一元线性回归模型给出的系数在实际情况中较为罕见，因此针对茄类商品销量价格关系可以考虑建立更为复杂的模型，并在第三问茄类利润预测时导入新模型，再用遗传算法求解总利润最大。

七、参考文献

[1]姜爽,沈烈志,金玉然.基于成本加成定价法的玉件产品定价模型研究[J].商场现代化,2007(13):66-67.

[2]张敦穆,李楚霖.再论成本加成定价的动力学[J].数学杂志,1996(04):2-8.

[3]吕晓永.基于报童模型的中小型超市生鲜产品订货策略改进研究[J].价值工程,2015,34(30):151-152.DOI:10.14018/j.cnki.cn13-1085/n.2015.30.061.

[4]陈咏鑫,李霞,宋词.超市商品的最优库存量研究[J].商业经济,2018(08):127-128+144.

[5]杨皓旭.蔬菜价格与销量的相关性分析研究——以油麦菜为例[J].食品安全导刊,2018(21):179-181.DOI:10.16043/j.cnki.cfs.2018.21.146.

八、附录

8.1 数据处理补充说明

首先使用问题一中去除了退货单和对应无效销售单的总流水数据，通过将销售日期与商品品类名称组合获得新列 concat1；通过每单的销量乘销售时的单价计算销售价值，vlookup1 对应当天该品类总质量，将前面的销量除以这一项得到该单质量占当天总品类质量占比。将销售日期与商品单品名称组合获得新列 concat2，通过 vlookup2 对应出当天改品类的批发价。单位利润是将该单流水的出售价减去批发价，计算质量占比乘批发价，对该列进行数据透视表求和可以得到每种品类的平均批发价格。使用 vlookup3 对应该单对应的单品的附件四中给定的质量损耗率。计算平均出售价格，使用数据透视表对该列数据进行分类求和可以得到每一种品类每一天的平均售价。计算折扣出售的总出售价值，即非折扣出售等于 0，折扣出售则等于前面的价值。计算质量折损率的贡献，批发价值的贡献与价值损耗的贡献。

通过相似方法，并使用数据透视表，最后得到了一些刻画数据特征的表格。例如，损耗总值、折扣比率贡献、折扣出售总量、平均出售价格、平均出售原价、平均折扣、平均批发价格等。

8.2 支撑材料列表

1. Q1.txt　//问题一 Python 代码
2. Q2.txt　//问题二 Python 代码
3. Q3.txt　//问题三 Python 代码
4. ARIMA.txt　//问题二 ARIMA 代码
5. Matlab.txt　//问题一 Matlab 代码
6. 0624-0630LSTM 预测.xlsx
7. ARIMA.xlsx
8. new_3.xlsx
9. new_4.xlsx
10. Result_Q3.xlsx
11. sum_by_day.xlsx
12. 按日期 K 聚类_K=4_最终聚类中心.xlsx
13. 按日期 K 聚类与系统聚类数据源.xlsx
14. 按小时 K 聚类_K=5_最终聚类中心.xlsx
15. 层次聚类.png

16. 多元线性回归数据源.xlsx
17. 季节聚类.png
18. 可选单品.xlsx
19. 皮尔逊相关系数与检验.xlsx
20. 平均批发价格.xlsx
21. 时间聚类.png
22. 损耗总值.xlsx
23. 问题二：使用 ARIMA 预测得到的未来一周品类补货量.xlsx
24. 题一各品类销量、描述性统计和皮尔逊相关系数分析.xlsx
25. 折扣比率贡献.xlsx
26. 折扣出售总量.xlsx
27. 按日期 K 聚类_K=4_聚类输出.spv
28. 按日期 K 聚类_K=4_最终聚类中心.sav
29. 系统聚类输出.spv

8.3 模型源代码

8.3.1 Q1.txt

```
# 进退货量相抵，删除相关数据
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
import pandas as pd

file_path = './dates/name.xlsx'
sheet_name = 'Sheet1'
column_name1 = '销量(千克)'
column_name2 = '销售单价(元/千克)'

df = pd.read_excel(file_path, sheet_name=sheet_name)

# 标记负值所在的行
negative_indices = []
for i in range(len(df) - 1, -1, -1):
    if df.at[i, column_name1] < 0:
        negative_indices.append(i)
# print(negative_indices)

# 删除最近的相反数所在的行
while len(negative_indices) > 0:
    i = negative_indices.pop()
```

```
        print(i)
        j = i - 1    # 前一行索引

        while j >= 0:
            if j in df.index and df.at[j, column_name1] == -df.at[i, column_name1] and
df.at[j, column_name2] == df.at[j, column_name2]:
                df.drop([i, j], inplace=True)    # 删除这两行
                break
            j -= 1



output_file = './dates/output_name.xlsx'
df.to_excel(output_file, index=False)

# 将数据标准化处理
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
import pandas as pd

# 读取 Excel 文件
file_path1 = './dates/clustering.xlsx'
sheet_name = 'Sheet3'

df = pd.read_excel(file_path1, sheet_name=sheet_name)

# 循环遍历每一列并进行 Min-Max 标准化
for column in df.columns:
    # 检查列是否包含数值数据
    if df[column].dtype in ['int64', 'float64']:
        # 计算列的最小值和最大值
        min_value = df[column].min()
        max_value = df[column].max()

        # 对列中的每个值进行 Min-Max 标准化
        df[column] = (df[column] - min_value) / (max_value - min_value)


output_file = 'new_data2_day.xlsx'
df.to_excel(output_file, index=False)

print("数据已标准化并保存到新的 Excel 文件。")
```

```
# 空值补 0
import pandas as pd
import string

file_path = './dates/clustering_hour.xlsx'
sheet_name = 'Sheet1'

df = pd.read_excel(file_path, sheet_name=sheet_name)

start_row = 2
end_row = 14177
start_col = 'B'
end_col = 'IM'

# 转换列名为列索引
def col_to_index(col_name):
    if len(col_name) == 1:
        return ord(col_name) - ord('A') + 1
    else:
        first_letter, second_letter = col_name[0], col_name[1]
        return (ord(first_letter) - ord('A') + 1) * 26 + (ord(second_letter) - ord('A') +
1)

start_col_index = col_to_index(start_col)
end_col_index = col_to_index(end_col)

# 将指定范围内的空单元格替换为零
for row in range(start_row, end_row + 1):
    for col in range(start_col_index, end_col_index + 1):
        cell_value = df.iloc[row - 1, col - 1]
        if pd.isna(cell_value):
            df.iloc[row - 1, col - 1] = 0


output_file = './dates/clustering_hour_zero.xlsx'
df.to_excel(output_file, index=False)

# 数据标准化处理
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
import pandas as pd
```

```
file_path1 = './dates/clustering_hour_zero.xlsx'
sheet_name = 'Sheet1'

df = pd.read_excel(file_path1, sheet_name=sheet_name)

# 循环遍历每一列并进行 Min-Max 标准化
for column in df.columns:
    # 检查列是否包含数值数据
    if df[column].dtype in ['int64', 'float64']:
        # 计算列的最小值和最大值
        min_value = df[column].min()
        max_value = df[column].max()

        # 对列中的每个值进行 Min-Max 标准化
        df[column] = (df[column] - min_value) / (max_value - min_value)

output_file = './dates/output_clustering_hour_zero.xlsx'
df.to_excel(output_file, index=False)

print("数据已标准化并保存到新的 Excel 文件。")

# 散布矩阵
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


file_path = './dates/皮尔逊相关系数.xlsx'
sheet_name = 'Sheet1'
df = pd.read_excel(file_path, sheet_name=sheet_name)

# 假设日期列为第一列，品类名称在第一行，从第二列第二行开始是销量数据
# 将日期列设置为索引，然后去掉品类名称行
df.set_index('行标签', inplace=True)
df.columns = df.iloc[0]
df = df[1:]

# 创建散布矩阵
sns.set(style="ticks")
sns.pairplot(df)
```

```
# 显示图形
plt.show()

# FG-Growth 算法
import pandas as pd
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import association_rules

# 读取包含商品销量的 DataFrame
file_path = './dates/new_data2_day.xlsx'
sheet_name = 'Sheet4'

df = pd.read_excel(file_path, sheet_name=sheet_name)

df = df.set_index('Date')  # 将日期列设置为索引
df = df.applymap(lambda x: 1 if x > 0 else 0)  # 将销量列转换为二进制

# 使用 FP-Growth 算法来找出频繁项集
frequent_itemsets = fpgrowth(df, min_support=0.1, use_colnames=True)

# 使用关联规则来找出关联性强的商品对
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1.0)

print(rules)

# SAM 算法
import pandas as pd
from collections import defaultdict

# 读取包含商品销量的 DataFrame
file_path = './dates/new_data2_day.xlsx'
sheet_name = 'Sheet2'

df = pd.read_excel(file_path, sheet_name=sheet_name)

df = df.set_index('Date')  # 将日期列设置为索引
df = df.applymap(lambda x: 1 if x > 0 else 0)  # 将销量列转换为二进制

# 定义 SAM 算法的最小支持度阈值
min_support = 0.1

# 扫描数据并计算单项的支持度
```

```
item_support = df.sum() / len(df)

# 初始化频繁一项集
frequent_itemsets = []

# 遍历每个商品，寻找频繁一项集
for item, support in item_support.items():
    if support >= min_support:
        frequent_itemsets.append([item])

# 从频繁一项集构建频繁二项集
k = 2
while True:
    new_frequent_itemsets = []
    for i in range(len(frequent_itemsets)):
        for j in range(i + 1, len(frequent_itemsets)):
            itemset1 = set(frequent_itemsets[i])
            itemset2 = set(frequent_itemsets[j])
            union_itemset = itemset1.union(itemset2)
            if len(union_itemset) == k:
                # 计算新项集的支持度
                support = (df[list(union_itemset)].sum(axis=1) == k).mean()
                if support >= min_support:
                    new_frequent_itemsets.append(sorted(list(union_itemset)))
    if not new_frequent_itemsets:
        break
    frequent_itemsets = new_frequent_itemsets
    k += 1

# 打印频繁项集
for itemset in frequent_itemsets:
    print(itemset)

# 谱系图
import pandas as pd
import matplotlib.pyplot as plt

# 读取 Excel 表格
file_path = './dates/new_data2_day.xlsx'
sheet_name = 'sum'
df = pd.read_excel(file_path, sheet_name=sheet_name)

# 数据预处理
```

```
df = df.transpose()
df.columns = df.iloc[0]
df = df.iloc[1:]
df.index = df.index.astype(str)


plt.figure(figsize=(100, 6))

for column in df.columns:
        plt.plot(df.index, df[column], label=column)

plt.xlabel('Category')
plt.ylabel('Sales')
plt.title('Hierarchical Clustering Dendrogram')
plt.legend(df.columns)
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# 聚类层次分析谱系图
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import linkage, dendrogram

file_path = './dates/new_data2_day.xlsx'
sheet_name = 'Sheet2'
data = pd.read_excel(file_path, sheet_name=sheet_name)

data.set_index('Category', inplace=True)

# 使用层次聚类方法进行聚类
linkage_matrix = linkage(data, method='ward')

plt.figure(figsize=(3, 20))
dendrogram(linkage_matrix, labels=data.index, orientation='right')
plt.title('Hierarchical Clustering Dendrogram')
plt.ylabel('Categories')
plt.xlabel('Sales')
plt.tight_layout()
plt.show()
```

```
# 皮尔逊相关系数热力图
import seaborn as sns
import matplotlib.pyplot as plt

matrix_data = [
    [1, 0.627113, 0.550916, 0.311608, 0.522983, 0.542215],
    [0.627113, 1, 0.659375, 0.257425, 0.63078, 0.561058],
    [0.550916, 0.659375, 1, 0.273339, 0.687516, 0.614262],
    [0.311608, 0.257425, 0.273339, 1, 0.119841, 0.074392],
    [0.522983, 0.63078, 0.687516, 0.119841, 1, 0.670054],
    [0.542215, 0.561058, 0.614262, 0.074392, 0.670054,1],
]

sns.set(style="whitegrid")
cmap = "coolwarm_r"
plt.figure(figsize=(8, 6))
sns.heatmap(matrix_data, annot=True, fmt=".6f")
plt.show()
```

8.3.2 Q2.txt

```
# 品类分析
## 神经网络预测未来七天
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt


file_path = './dates/predict_standard.xlsx'
sheet_name = 'Sheet1'
df = pd.read_excel(file_path, sheet_name=sheet_name)


sales_data = df.iloc[:, 1:7]
scaler = MinMaxScaler()
scaled_sales_data = scaler.fit_transform(sales_data)

train_size = int(len(scaled_sales_data) * 0.8)
train_data = scaled_sales_data[:train_size]
test_data = scaled_sales_data[train_size:]
```

```
def create_dataset(data, look_back=7):
    X, y = [], []
    for i in range(look_back, len(data)):
        X.append(data[i - look_back:i])
        y.append(data[i])
    return np.array(X), np.array(y)

X_train, y_train = create_dataset(train_data)
X_test, y_test = create_dataset(test_data)

model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(7, 6)))
model.add(Dense(32, activation='relu'))
model.add(Dense(6, activation='linear'))
model.compile(optimizer='adam', loss='mean_squared_error')


model.fit(X_train, y_train, epochs=1000, batch_size=164, verbose=2)
future_sales = []
last_week_sales = X_test[-1]

for _ in range(7):
    next_day_sales = model.predict(last_week_sales.reshape(1, 7, 6))
    future_sales.append(next_day_sales[0])
    last_week_sales = np.roll(last_week_sales, shift=-1, axis=0)
    last_week_sales[-1] = next_day_sales[0]
future_sales = scaler.inverse_transform(np.array(future_sales))
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
print(future_sales)
future_df = pd.DataFrame(data=future_sales, columns=sales_data.columns)
output_file_path = './dates/output_sales_prediction.xlsx'
with pd.ExcelWriter(output_file_path, engine='xlsxwriter') as writer:
    future_df.to_excel(writer, sheet_name='Predictions', index=False)
# 绘制销售预测图表
plt.figure(figsize=(10, 6))
dates = df['Date'].iloc[train_size + 7:train_size + 7 + len(future_sales)].values
for i in range(6):
    plt.plot(dates, test_data[:, i][:len(future_sales)], label=f'real-{i+1}')
    plt.plot(dates, future_sales[:, i], label=f'predict-{i+1}', linestyle='--')
plt.xlabel('Date')
plt.ylabel('Sales')
```

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

## LSTM
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
import matplotlib.pyplot as plt

file_path = './dates/predict_standard.xlsx'
sheet_name = 'Sheet1'
df = pd.read_excel(file_path, sheet_name=sheet_name)

sales_data = df.iloc[:, 1:7]
scaler = MinMaxScaler()
scaled_sales_data = scaler.fit_transform(sales_data)

def create_dataset(data, look_back=7):
    X, y = [], []
    for i in range(look_back, len(data)):
        X.append(data[i - look_back:i])
        y.append(data[i])
    return np.array(X), np.array(y)

look_back = 7
X, y = create_dataset(scaled_sales_data, look_back)

train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]


model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(look_back, 6)))
model.add(Dense(32, activation='relu'))
model.add(Dense(6, activation='linear'))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=1000, batch_size=164, verbose=2)
future_sales = []
last_week_sales = X_test[-1]

for _ in range(7):
```

```
        next_day_sales = model.predict(last_week_sales.reshape(1, look_back, 6))
        future_sales.append(next_day_sales[0])
        last_week_sales = np.roll(last_week_sales, shift=-1, axis=0)
        last_week_sales[-1] = next_day_sales[0]
future_sales = scaler.inverse_transform(np.array(future_sales))

dates = df['Date'].iloc[look_back:].values
future_df = pd.DataFrame(data=future_sales, columns=sales_data.columns)
future_df['Date'] = dates[-7:]
output_file_path = './dates/output_sales_prediction_category_lstm.xlsx'
with pd.ExcelWriter(output_file_path, engine='xlsxwriter') as writer:
        future_df.to_excel(writer, sheet_name='Predictions', index=False)
future_df = pd.DataFrame(data=future_sales, columns=sales_data.columns)
future_df['Date'] = pd.date_range(start='2023-07-01', periods=7)
plt.figure(figsize=(12, 6))
plt.xlim(pd.Timestamp('2023-01-01'), pd.Timestamp('2023-07-07'))
for i in range(6):
        plt.plot(df['Date'], scaled_sales_data[:, i], label=f'real-{i+1}')
        plt.plot(future_df['Date'], future_sales[:, i], label=f'predict-{i+1}', linestyle='--')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()

# 单品预测
## LSTM
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
import matplotlib.pyplot as plt

file_path = './dates/new_data2_day.xlsx'
sheet_name = 'Sheet1'
df = pd.read_excel(file_path, sheet_name=sheet_name)

sales_data = df.iloc[:, 1:247]
scaler = MinMaxScaler()
scaled_sales_data = scaler.fit_transform(sales_data)
def create_dataset(data, look_back=7):
        X, y = [], []
        for i in range(look_back, len(data)):
```

```
            X.append(data[i - look_back:i])
            y.append(data[i])
    return np.array(X), np.array(y)

look_back = 7
X, y = create_dataset(scaled_sales_data, look_back)
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(look_back, 246)))
model.add(Dense(32, activation='relu'))
model.add(Dense(246, activation='linear'))  # 输出层也要有 246 个神经元
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=1000, batch_size=164, verbose=2)

future_sales = []
last_week_sales = X_test[-1]

for _ in range(7):
    next_day_sales = model.predict(last_week_sales.reshape(1, look_back, 246))
    future_sales.append(next_day_sales[0])
    last_week_sales = np.roll(last_week_sales, shift=-1, axis=0)
    last_week_sales[-1] = next_day_sales[0]
future_sales = scaler.inverse_transform(np.array(future_sales))
dates = df['Date'].iloc[look_back:].values
future_df = pd.DataFrame(data=future_sales, columns=sales_data.columns)
future_df['Date'] = dates[-7:]

output_file_path = './dates/output_sales_prediction_product_lstm.xlsx'
with pd.ExcelWriter(output_file_path, engine='xlsxwriter') as writer:
    future_df.to_excel(writer, sheet_name='Predictions', index=False)


## 模型检验
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
actual_sales = sales_data.iloc[-7:].values
rmse = np.sqrt(mean_squared_error(actual_sales, future_sales))
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
mae = mean_absolute_error(actual_sales, future_sales)
print(f"Mean Absolute Error (MAE): {mae:.2f}")
r2 = r2_score(actual_sales, future_sales)
```

```
print(f"R-squared (R2): {r2:.2f}")
```

8.3.3 Q3.txt

```
# LSTM 预测 7.1 销量
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense


file_path = './dates/0624-0630.xlsx'
sheet_name = 'Sheet7'
df = pd.read_excel(file_path, sheet_name=sheet_name)
sales_data = df.iloc[:, 1:50]
# print(sales_data)
scaler = MinMaxScaler()
scaled_sales_data = scaler.fit_transform(sales_data)
def create_dataset(data, look_back=49):
    X, y = [], []
    for i in range(look_back, len(data)):
        X.append(data[i - look_back:i])
        y.append(data[i])
    return np.array(X), np.array(y)
look_back = 49
X, y = create_dataset(scaled_sales_data, look_back)
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(look_back, 50)))
model.add(Dense(32, activation='relu'))
model.add(Dense(50, activation='linear'))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=100, batch_size=16, verbose=2)
last_week_sales = X_test[-1]
next_day_sales = model.predict(last_week_sales.reshape(1, look_back, 50))
next_day_sales = scaler.inverse_transform(next_day_sales)
dates = df['Date'].iloc[look_back:].values
future_df = pd.DataFrame(data=next_day_sales, columns=sales_data.columns)
future_df['Date'] = dates[-1:]
output_file_path = './dates/output_0624-0630.xlsx'
with pd.ExcelWriter(output_file_path, engine='xlsxwriter') as writer:
    future_df.to_excel(writer, sheet_name='Predictions', index=False)
```

```
# 遗传算法
import random
import pandas as pd
import numpy as np
from tqdm import tqdm
from deap import base, creator, tools, algorithms


file_path = './dates/Q3_predict.xlsx'
sheet_name = 'Sheet2'
df = pd.read_excel(file_path, sheet_name=sheet_name)
i_values = df['i'].tolist()
a_values = df['a[i]'].tolist()
b_values = df['b[i]'].tolist()
AC_values = df['AC[i]'].tolist()
initial_K_values = df['K'].tolist()
initial_S_values = df['S'].tolist()
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
toolbox = base.Toolbox()
def generate_Ki():
    return random.uniform(0, 1)
def generate_Si():
    return random.uniform(0,1)

toolbox.register("individual", tools.initCycle, creator.Individual, (generate_Ki,
generate_Si), n=49)
toolbox.register("initial_K_values", lambda x: x, initial_K_values)
toolbox.register("initial_S_values", lambda x: x, initial_S_values)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evaluate(individual):
    total_w = 0.0
    zero_count=individual.count(0)
    for i in range(49):
        Ki = individual[i * 2]
        Si = individual[i * 2 + 1]
        if not (16<= zero_count<=22):
            return -float('inf'),
        # 添加约束条件
        if not (0 <= Ki < 1) or not ( Si == 0 or Si >= 2.5):
            return -float('inf'),
```

```
            if AC_values[i] / (1 - Ki) >= (Si - a_values[i]) / b_values[i]:
                w_i = b_values[i] * (AC_values[i] / (1 - Ki)) ** 2 + a_values[i] *
AC_values[i] / (1 - Ki) - Si * AC_values[i]
            else:
                w_i = Si * Ki * AC_values[i] / (1 - Ki)
            total_w += w_i
        return total_w,

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=0.1, indpb=0.2)
toolbox.register("select", tools.selTournament, tournsize=3)
population = toolbox.population(n=100)
fitnesses = list(map(toolbox.evaluate, population))
for ind, fit in zip(population, fitnesses):
    ind.fitness.values = fit
crossover_prob = 0.7
mutation_prob = 0.2
generations = 10000000

for gen in tqdm(range(generations), desc="Evolving"):
    offspring = algorithms.varAnd(population, toolbox, cxpb=crossover_prob,
mutpb=mutation_prob)
    fits = toolbox.map(toolbox.evaluate, offspring)
    for ind, fit in zip(offspring, fits):
        ind.fitness.values = fit
    population = toolbox.select(offspring, k=len(population))
best_individual = tools.selBest(population, k=1)[0]
best_Ki_values = best_individual[::2]
best_Si_values = best_individual[1::2]
best_fitness = best_individual.fitness.values[0]
print("Best Ki values:", best_Ki_values)
print("Best Si values:", best_Si_values)
print("Best Fitness:", best_fitness)
df = pd.DataFrame(predictions, columns=['Value'])
output_file_path = './dates/output_Q3_predict.xlsx'
df.to_excel(output_file_path, index=False)
```

8.3.4 ARIMA.txt

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

```
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.graphics.tsaplots import plot_pacf,plot_acf
from statsmodels.tsa.stattools import adfuller

df=pd.read_excel('C:/Users/Administrator/Desktop/ARIMA.xlsx')
df.info()
data=df.copy()
#不存在缺失值，不需要用移动平均法填充数据
plt.plot(data['日期'],data['花菜类'].values)
plt.show()

huacai=data['花菜类']
#单位根检验（ADF 检验）
print(adfuller(huacai))
#结果表明花菜类序列 0.05 显著性水平下平稳
#白噪声检验
acorr_ljungbox(huacai, lags = [6, 12],boxpierce=True)
#结果表明数据 0.01 显著性水平不是白噪声
train=huacai.loc[:700]
test=huacai.loc[701:].reset_index(drop=True)
model = sm.tsa.arima.ARIMA(train,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
predict=arima_res.predict(start=test.index[0],end=test.index[len(test)-1])
plt.plot(test.index,predict,color='blue',label='predict')
plt.plot(test.index,test,color='red',label='test')
plt.legend()
plt.show()
from sklearn.metrics import r2_score,mean_absolute_error
mean_absolute_error(test,predict)
#残差图
residual=test-predict
plt.plot(residual,label='residual')
plt.legend()
np.mean(residual)
#残差正态性检验
import seaborn as sns
from scipy import stats
plt.figure(figsize=(10,5))
ax=plt.subplot(1,2,1)
sns.distplot(residual,fit=stats.norm)
ax=plt.subplot(1,2,2)
```

```
res=stats.probplot(residual,plot=plt)
plt.show()
#计算 R 方
print(r2_score(test,predict))
#计算 ACF 拖尾
acf=plot_acf(huacai)
plt.show()
#PACF 3 阶截尾
pacf=plot_pacf(huacai)
plt.show()
'''也可以使用信息准则帮助判断最佳阶数
trend_evaluate = sm.tsa.arma_order_select_ic(huacai, ic=['aic', 'bic'], trend='n',
max_ar=20,max_ma=10)
print('AIC', trend_evaluate.aic_min_order)
print('BIC', trend_evaluate.bic_min_order)
'''
model = sm.tsa.arima.ARIMA(huacai,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
test=pd.read_excel('C:/Users/Administrator/Desktop/test.xlsx')
predict=arima_res.predict(start=test.index[0],end=test.index[6])
plt.plot(test.index,predict)
plt.show()
test['花菜类']=predict
##
plt.plot(data['日期'],data['辣椒类'].values)
plt.show()

lajiao=data['辣椒类']
#单位根检验（ADF 检验）
print(adfuller(lajiao))
#结果表明花菜类序列 0.05 显著性水平下平稳
#白噪声检验
acorr_ljungbox(lajiao, lags = [6, 12],boxpierce=True)
#结果表明数据 0.01 显著性水平下不是白噪声
#计算 ACF 拖尾
acf=plot_acf(lajiao)
plt.show()
#PACF 3 阶截尾
pacf=plot_pacf(lajiao)
plt.show()
```

```
model = sm.tsa.arima.ARIMA(lajiao,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
predict=arima_res.predict(start=test.index[0],end=test.index[6])
plt.plot(test.index,predict)
plt.show()
test['辣椒类']=predict
##
plt.plot(data['日期'],data['茄类'].values)
plt.show()

qie=data['茄类']
#单位根检验（ADF 检验）
print(adfuller(qie))
#结果表明花菜类序列 0.05 显著性水平下平稳
#白噪声检验
acorr_ljungbox(qie, lags = [6, 12],boxpierce=True)
#结果表明数据 0.01 显著性水平下不是白噪声
#计算 ACF  拖尾
acf=plot_acf(qie)
plt.show()
#PACF 3 阶截尾
pacf=plot_pacf(qie)
plt.show()


model = sm.tsa.arima.ARIMA(qie,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
predict=arima_res.predict(start=test.index[0],end=test.index[6])
plt.plot(test.index,predict)
plt.show()
test['茄类']=predict
##
plt.plot(data['日期'],data['食用菌'].values)
plt.show()

shiyongjun=data['食用菌']
#单位根检验（ADF 检验）
print(adfuller(shiyongjun))
#结果表明花菜类序列 0.05 显著性水平下平稳
#白噪声检验
```

```
acorr_ljungbox(shiyongjun, lags = [6, 12],boxpierce=True)
#结果表明数据 0.01 显著性水平下不是白噪声
#计算 ACF  拖尾
acf=plot_acf(shiyongjun)
plt.show()
#PACF 3 阶截尾
pacf=plot_pacf(shiyongjun)
plt.show()

model = sm.tsa.arima.ARIMA(shiyongjun,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
predict=arima_res.predict(start=test.index[0],end=test.index[6])
plt.plot(test.index,predict)
plt.show()
test['食用菌']=predict
##
plt.plot(data['日期'],data['水生根茎类'].values)
plt.show()

shuisheng=data['水生根茎类']
#单位根检验（ADF 检验）
print(adfuller(shuisheng))
#结果表明花菜类序列 0.05 显著性水平下平稳
#白噪声检验
acorr_ljungbox(shuisheng, lags = [6, 12],boxpierce=True)
#结果表明数据 0.01 显著性水平下不是白噪声
#计算 ACF  拖尾
acf=plot_acf(shuisheng)
plt.show()
#PACF 3 阶截尾
pacf=plot_pacf(shuisheng)
plt.show()

model = sm.tsa.arima.ARIMA(shuisheng,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
predict=arima_res.predict(start=test.index[0],end=test.index[6])
plt.plot(test.index,predict)
plt.show()
test['水生根茎类']=predict
##
```

```
plt.plot(data['日期'],data['花叶类'].values)
plt.show()

huaye=data['花叶类']
#单位根检验（ADF 检验）
print(adfuller(huaye))
#结果表明花叶类序列 0.1 显著性水平下平稳，不用使用差分进行平稳
#白噪声检验
acorr_ljungbox(huaye, lags = [6, 12],boxpierce=True)
#结果表明数据 0.01 显著性水平不是白噪声
#计算 ACF 3 阶截尾
acf=plot_acf(huaye)
plt.show()
#PACF 3 阶截尾
pacf=plot_pacf(huaye)
plt.show()

model = sm.tsa.arima.ARIMA(huaye,order=(1,0,3))
arima_res=model.fit()
arima_res.summary()
predict=arima_res.predict(start=test.index[0],end=test.index[6])
plt.plot(test.index,predict)
plt.show()
test['花叶类']=predict
test.to_excel('C:/Users/Administrator/Desktop/testpredict.xlsx')
```

8.3.5 Matlab.txt

```
num = xlsread('问题一各品类销量描述性统计与皮尔逊相关系数分析.xlsx')
%% 统计描述
MIN = min(num); % 每一列的最小值
MAX = max(num); % 每一列的最大值
MEAN = mean(num); % 每一列的均值
MEDIAN = median(num); %每一列的中位数
SKEWNESS = skewness(num); %每一列的偏度
KURTOSIS = kurtosis(num); %每一列的峰度
STD = std(num); % 每一列的标准差
format short
RESULT = [MIN;MAX;MEAN;MEDIAN;SKEWNESS;KURTOSIS;STD] %将这些
统计量放到一个矩阵中表示
```