最大化商超收益的单品补货优化模型

摘　要

本文的问题场景为一个生鲜商超在补货和定价决策上所面临的挑战。商超销售的蔬菜品类众多，供给和需求的变化对决策产生影响，因此可靠的市场需求分析对决策至关重要。通过分析销售数据、商品信息和损耗率等数据，建立数学模型来解决问题。

针对问题 1，采用时间序列分析，通过分析附件 2 中的销售流水明细数据，得到蔬菜各品类和单品的销售量分布情况，绘制出销售量的折线图等来展示分布规律，分析不同蔬菜品类和单品的销售趋势，检查是否存在季节性变化、趋势或周期性波动。生成滚动统计时间序列图来观察销售量的平均值和标准差的变化。计算不同品类之间的销售量相关系数，了解品类销售关联性。使用线性回归模型描述不同品类间的线性关系。

针对问题 2，通过分析附件 2 和附件 3 中的销售流水明细和批发价格数据，计算出各蔬菜品类的销售总量和成本加成定价之间的关系。

首先进行数据整合，获取每笔销售的成本以及损耗情况；其次计算总成本及每个蔬菜品类的销售总量，然后基于销售总量与定价的历史数据，建立数学模型。依据建立的数学模型，预测未来一周的销售总量，最终得到最优的补货总量，定价策略则通过计算品类的平均成本和市场需求强度制定。

针对问题 3，根据 2023 年 6 月 24-30 日的可售品种数据和附件 4 中的损耗率数据，筛选出 2023 年 6 月 24-30 日的可售品种，建立每个单品的销售需求模型和 ARIMA 模型；其次用回归分析的方法，将商品的定价与其他相关因素（如批发价格、销售量等）建立关联，构建线性回归模型。通过建立的模型，在满足可售单品总数的限制条件和最小陈列量的要求同时，优化商超的收益，求解出 7 月 1 日的单品补货量和定价策略，以最大化商超的收益。

针对问题 4，除了已提供的销售数据、商品信息和损耗率数据外，通过分析不同单品销售量的时间序列图，建议商超还可以考虑收集以下数据：市场行情数据、供应商数据、顾客调查数据。

以上是关于商超蔬菜补货和定价决策问题的分析和解决方案。

关键词： 时间序列分析；市场需求强度；销售需求模型；线性回归模型；ARIMA 模型

1．问题的重述

蔬菜类商品保鲜期较短隔日无法再售、销售量与时间具有一定的关系及蔬菜品种在4月到10月较丰富，商超对于品相变差和运损的蔬菜通常进行打折促销，且通常按照“成本加成定价”的方法给蔬菜定价。在此背景下需解决以下四个问题。

问题1：销售数据分析。在生鲜超市中，如何分析不同蔬菜品类和单品的销售量分布规律，以及它们之间可能存在的关联关系？

问题2：品类级别的补货与定价。在商超以品类为单位进行补货计划和定价策略制定时，如何根据销售总量和成本加成定价，优化未来一周各蔬菜品类的日补货总量和定价策略，以最大化商超收益？

问题3：单品级别的补货与定价。商超希望制定单品级别的补货计划和定价策略，同时确保可售单品总数在27-33个之间，如何在满足市场需求的前提下，最大化商超的收益，包括7月1日的单品补货量和定价策略？

问题4：数据需求建议。为更好的制定蔬菜商品的补货和定价决策，商超还需要采集哪些相关数据，以满足上述问题的解决需求？

2．问题的分析

本文主要运用统计学及数据分析相关知识，通过分析销售数据、商品信息和损耗率等数据，建立数学模型来解决蔬菜类商品的定价以及补货决策。

围绕蔬菜类商品的定价与补货决策分析，主要包括以下问题：

2.1 问题一

对于问题一，考虑到不同时间段内的销售量可能有所不同，使用附件2中的销售流水明细数据，分析不同蔬菜品类和单品的销售趋势，例如每月、每周或每日的销售量。

绘制销售量的时间序列图，以识别季节性趋势和销售峰值。使用相关性分析来查看不同品类或单品之间的销售关联性。

为更好的建立模型解决问题，首先假设蔬菜类商品不同单品或不同品类之间存在线性关系。通过分析附件2中的销售流水明细数据以及使用附件1的分类信息，将销售数据与品类信息关联起来。通过单品编码和分类编码之间的映射，计算出不同分类的总销售量。

考虑到不同品类的蔬菜，季节不同，蔬菜的种类也不尽相同，且蔬菜销售量的变化主要受时间和季节性因素影响，我们采用时间序列分析法，将蔬菜销售量按销售日期进行排列，构成统计的时间序列，绘制蔬菜销售量随时间的变化图表，查看是否存在季节性或趋势性变化。

然后对品类进行线性摘要，帮助评估模型的质量和拟合程度。

其次绘制出蔬菜销售量随时间变化的变化图表和线性回归拟合线，确定销售量与其他因素的关系。

最后结合相关系数计算公式（$corr$）：

$$A = mean((y\_real - mean(y\_real)) * (y\_predict - mean(y\_predict))) \quad (1)$$

$$B = std(y\_real) * std(y\_predict) \quad (2)$$

$$corr = A / B \quad (3)$$

建立线性回归模型：使用线性回归模型，将其中一种蔬菜品类的销售量作为解释变量，另一种蔬菜品类销售量作为响应变量来建立两者之间的线性回归方程，描述两种蔬菜品类之间的线性关系。

最终得到蔬菜各品类及单品销售量的分布规律及相互关系。

2.2 问题二

对于问题二，参考问题一的解决方法，将附件 2，附件 3，附件 4 的数据处理后，分析销售流水明细数据，基于线性回归分析，构建销售总量和定价的数学模型，进而确定销售总量与成本加成定价的关系。

为实现商超收益最大化，我们按照以下步骤制定最优补货总量和定价策略：

1.计算各品类的平均成本：我们可以使用附件 2、附件 3、附件 4 中的数据计算每个品类的平均成本。

2.建立定价策略模型：我们可以使用简化的定价策略模型，基于成本加成和市场需求强度来制定定价策略。这里，我们假设一个加成率，然后根据市场需求强度调整价格。

3.计算未来一周的需求预测：使用时间序列分析模型（如 ARIMA）以及历史销售数据，可以预测未来一周（2023 年 7 月 1-7 日）的销售需求。

4.确定补货策略：基于需求预测、当前库存水平、平均成本和定价策略，计算出每天的补货总量。

5.计算收益：最后，我们可以计算每天的销售收益，即销售量乘以价格减去成本，以确定最优的补货和定价策略，使商超的收益最大化。

2.3 问题三

为了解决问题三，我们可以按照以下步骤制定单品的补货计划：

1.使用销售需求模型，预测每个单品在未来 7 天的销售需求。这些预测值将指导我们在接下来的 7 天内需要订购多少每个单品。

2.使用定价模型，预测每个单品在未来 7 天的定价。这将帮助我们确定每个单品的订购成本。

3.将预测的销售需求和定价结合在一起，计算每个单品的预计销售收入。

4.根据预计销售收入，制定一个补货计划，以确保总共 27-33 个可售单品的限制。

5.对于每个单品，计算需要订购的数量，以满足市场需求和最小陈列量的要求。

6．最后，将补货计划的结果保存到表格中，包括单品编码、单品名称、补货量、定价等信息。

3．模型的假设与符号说明

3.1 模型的假设

为建模需要，我们做出如下假设：

针对问题 1，模型假设如下：

线性关系假设： 我们假设销售量与销售单价之间存在线性关系，即销售量随销售单价的变化而变化。这个假设有助于我们使用线性回归模型来分析销售趋势和预测未来销售量。

加成系数假设： 我们假设成本加成是基于批发价格的，加成系数为 1.2。这个假设用于计算成本加成，并考虑了额外的成本。

销售趋势假设： 我们假设销售数据可能存在季节性或趋势性的变化。这意味着销售量可能在不同的时间段内有所不同，需要考虑这种趋势以更准确地预测未来销售。

销售数据独立性假设： 我们假设销售数据之间是独立的，即一天的销售不受前一天的销售影响。这个假设有助于建立时间序列模型。

时间序列稳定性假设： 在时间序列分析中，我们假设时间序列是稳定的，即均值和方差在不同时间段内保持不变。这个假设在建立时间序列模型时很重要。

针对问题 2，模型假设如下：

销售总量与价格弹性： 模型假设销售总量与价格之间存在一定的弹性关系，即价格的变化会影响销售数量。一般来说，价格上涨可能导致销售量下降，价格下降可能导致销售量增加。

市场需求强度： 模型假设市场需求强度是一个影响价格的因素。不同的品类可能具有不同的市场需求强度，高市场需求强度可能支持较高的价格，低市场需求强度可能需要更具竞争力的价格。

成本加成定价： 模型采用成本加成定价策略，即在产品成本的基础上加上一定的加成率来确定价格。这个加成率可以是静态的，也可以根据市场需求强度进行调整。

需求预测： 模型假设有可靠的需求预测数据，以便在未来一周内（2023 年 7 月 1-7 日）制定补货计划和定价策略。需求预测可以基于历史销售数据和时间序列分析方法，或者根据其他市场因素进行估算。

库存管理： 模型假设商超可以及时补货，以满足市场需求，并且可以根据当前库存水平制定补货计划。

单品销售独立性： 模型可能会忽略不同单品之间的销售关联关系，即每个单品的销售独立于其他单品。这可以简化模型，但在实际情况下，销售可能会相互影响。

损耗率稳定性： 模型假设损耗率在短期内相对稳定，即在未来一周内损耗率不会发生显著变化。

补货成本：模型可能不考虑补货的具体成本，如运输成本和存储成本。这些成本因实际情况而异，可以根据需要进行考虑。

竞争环境：模型可能不考虑竞争对手的定价策略和市场份额。这取决于商超所处的市场竞争情况。

针对问题 3，模型假设如下：

销售需求模型假设：

假设销售需求与历史销售数据和其他相关因素有关，可以使用时间序列模型（如 ARIMA）或其他相关模型来预测销售需求。

假设销售需求在未来 7 天内基本稳定，不受突发事件或季节性波动的影响。

定价模型假设：

假设定价与批发价格和销售量有关，可以使用线性回归或其他合适的定价模型进行建模。

假设定价模型在未来 7 天内仍然有效，不受市场变化或竞争因素的显著变化的影响。

补货计划制定假设：

假设补货计划的目标是满足市场需求，同时确保不超过总共 27-33 个可售单品的限制。

假设每个单品的订购量不能低于最小陈列量 2.5 千克的要求。

假设供应商有足够的库存和供应能力来满足补货需求。

单品之间独立性假设：

假设每个单品的销售需求和定价是独立的，不受其他单品的影响。这意味着每个单品的补货计划可以单独制定，不需要考虑交叉影响。

3.2 符号说明

符号	含义
correlation_matrix	相关系数
rolling_mean	7 天的滚动均值
sales_2023	2023 年的销售数据
category_avg_price	平均销售价格
current_inventory	当前库存水平
category_avg_loss_rate	每个品类的平均损耗率
category_net_demand	净需求
cost_per_category	每个品类的成本
market_demand_intensity_per_category	市场需求强度
pricing_strategy	每个品类的定价策略，通常以元/千克为单位
markup rate	加在成本上的百分比，用来制定产品的销售
demand forecast	未来一段时间内销售需求的预测数据
daily replenishment quantity	每天需要补货的数量，以千克为单位
A	实际值与预测值之间的相互系数
B	实际值与预测值标准差的乘积
Y	响应变量（品类的销售量）
X	解释变量（另一个品类的销售量）
β_0	模型的系数，表示截距
β_1	模型的系数，表示斜率
ε	误差项
markup rate	表示加在成本上的百分比
average cost	特定蔬菜品类的平均成本
sales revenue	每天的销售收入

4．模型的准备

在数学建模前进行模型的准备，主要包括以下几个步骤：

1.理解时间序列分析：时间序列分析是一种研究统计指标动态特征和周期特征及相关关系的重要方法，主要用于经济、商业、社会问题的预测和分析。时间序列分析的常用模型是自回归移动平均模型，它可以从按时间排序的数据点中抽取有价值的总结和统计信息。了解时间序列分析的基本概念、常用的模型和方法可以帮助我们对销售量的变化规律进行分析和预测。

2. 学习季节性分析方法：季节性分析是研究数据随季节变化的一种方法。了解季节性分析的常见技术和模型可以帮助我们分析销售量在一年内的周期性变化，并对其进行建模和预测。

3. 掌握线性回归模型：线性回归模型是用于描述两个或多个变量之间线性关系的统计模型。了解线性回归模型的基本原理和假设，以及应用线性回归模型进行参数估计和预测的方法，可以帮助我们研究蔬菜销售量与其他因素之间的相关性。

4. 了解最优化模型：最优化模型是研究如何在给定的约束条件下，使某一目标函数取得最优值的数学模型。掌握最优化模型的基本原理和常用的求解方法（如线性规划、聚类分析、回归分析等），可以帮助我们在问题求解过程中制定合理的补货总量和定价策略，并优化商超的收益。

5. 熟悉数据预处理方法：在建模过程中，通常需要对原始数据进行一些处理，如缺失值处理、异常值处理、平滑处理等。熟悉常用的数据预处理技术可以提高数据的质量和建模结果的准确性。

6. 学习统计分析工具和编程语言：掌握一些常用的统计分析工具和编程语言，可以方便进行数据的处理、模型的建立和求解，提高建模的效率和准确性。

7. 明确问题定义：明确研究的问题和目标，确定需要解决的假设和约束条件。

8. 合理选择变量：根据问题的特点和数据的性质，选择合适的自变量和因变量，并进行合理的命名。

5．模型的建立与求解

5．1　问题 1 的模型建立与求解

（1）不同蔬菜单品或品类的销售趋势分析

分析不同蔬菜品类和单品的销售趋势。检查是否存在季节性变化、趋势或周期性波动。提取特定单品的销售数据（以 102900005115779 为例）：

按月份对销售数据进行汇总并绘制每月销售量趋势图：

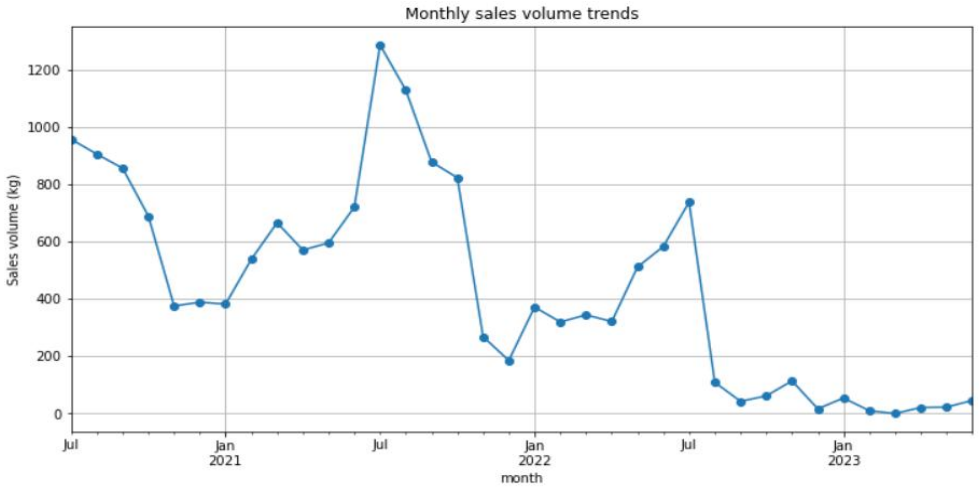

图 1　每月销售量趋势图

按周对销售数据进行汇总并绘制每周销售量趋势图：

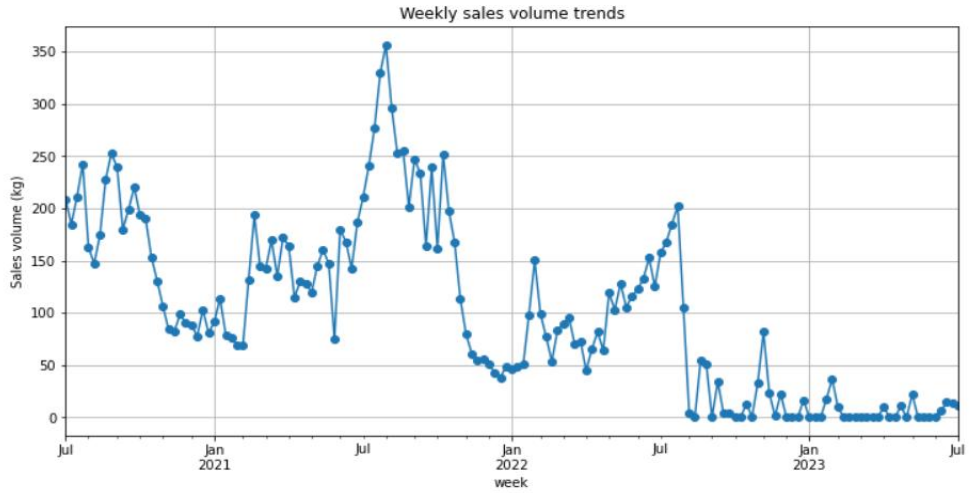

图 2　每周销售量趋势图

对每日销售量趋势分析：

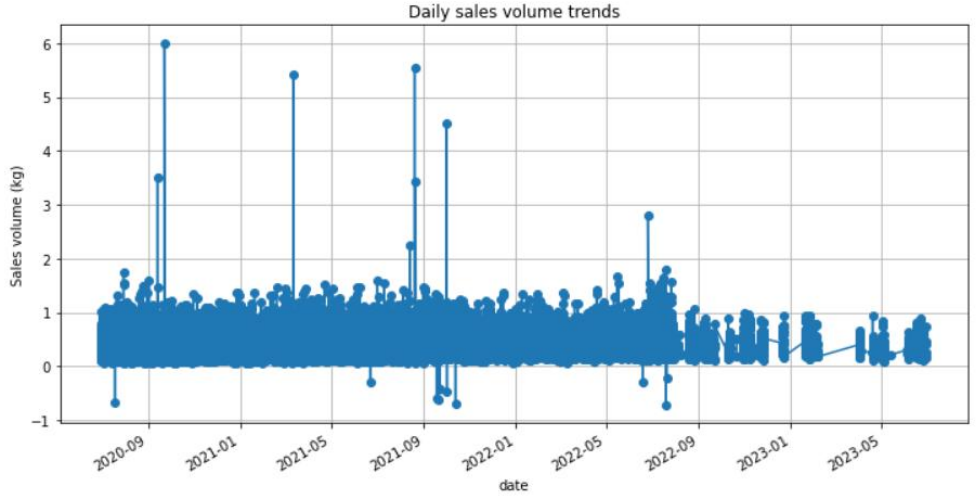

图 3　每日销售量趋势图

季节性趋势的观察：

根据图表显示的数据，我们可以观察到以下季节性趋势：

1. 在 2021 年和 2023 年之间，销售数据呈现出明显的季节性波动。这可能与特定季节或节日有关，并存在促销活动等。

2. 在 2021 年和 2022 年的夏季，销售额出现了多个极值。这可能与夏季是消费旺季有关，消费者更倾向于购买商品和服务。

3. 在 2023 年的春季，销售额再次出现了一个峰值。这可能与春季是新品发布和促销的时期有关，吸引了更多的消费者。

4. 在每年的秋季，销售额逐渐下降。这可能是由于消费者开始为节假日做准备，减少了购物活动。

为了更清晰地观察季节性趋势和销售峰值，本文借助生成滚动统计信息图，观察销售量的平均值和标准差的变化，以及它们与销售量的关系。

以下是针对一年使用滚动统计信息来观察季节性趋势和销售峰值：

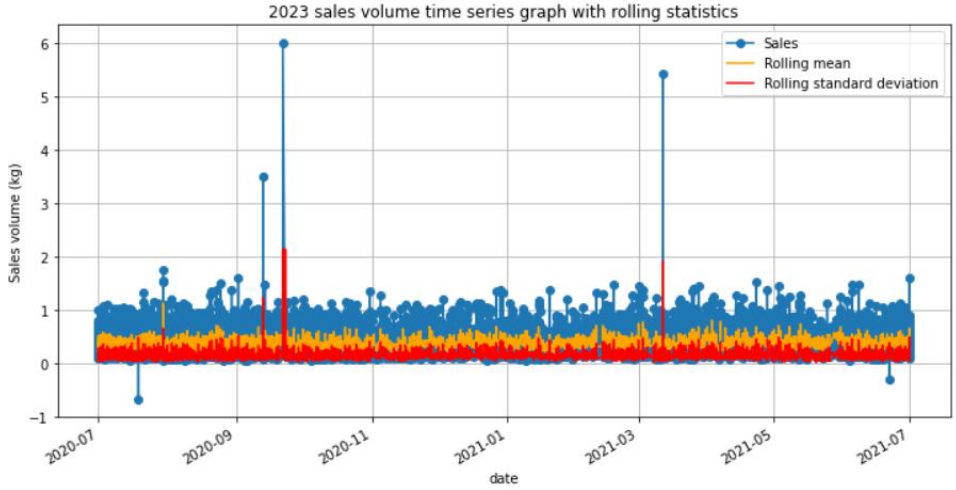

图 4　2023 年销量滚动统计时间序列图

以下是针对三年使用滚动统计信息来观察季节性趋势和销售峰值：

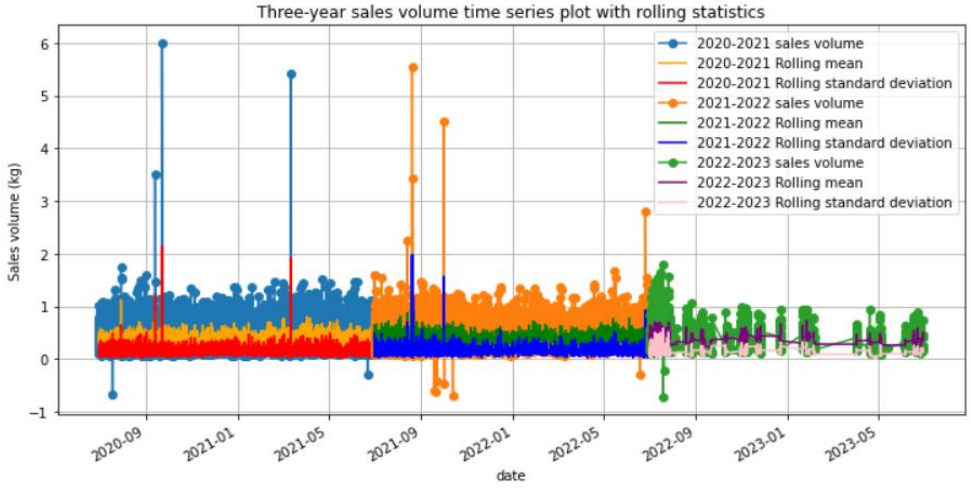

图 5　2020-2023 年销量滚动统计时间序列图

根据图表显示的数据，我们可以观察到以下内容：

1. 异常值：在滚动统计信息图中，明显发现存在多个异常值，它们可能是由于数据收集错误、测量误差或其他原因导致的。或者可能意味着存在突发事件或季节性因素，数据量在短时间内发生大幅波动。

2. 蓝色线表示销售量，橙色线表示滚动均值，红色线表示滚动标准差，观察滚动均值与滚动标准差变化法线，发现这些数据点比较接近平均值，并且数据的离散程度较小，说明数据的整体趋势比较稳定。

（2）相关性系数计算

计算不同蔬菜品类之间的销售量相关系数，可以了解不同蔬菜品类之间的销售关联性。正相关系数表示两个品类之间的销售量具有正相关关系，负相关系数表示负相关关系，而接近零的系数表示无相关性。

根据单品或品类的销售趋势分析，结合相关系数计算公式（$corr$）：

$$A = mean((y\_real - mean(y\_real)) * (y\_predict - mean(y\_predict))) \tag{4}$$

$$B = std(y\_real) * std(y\_predict) \tag{5}$$

$$corr = A / B \tag{6}$$

得出品类与品类之间的相关性，绘制出不同品类蔬菜销售量相关系数的热力图，如图 4 所示。

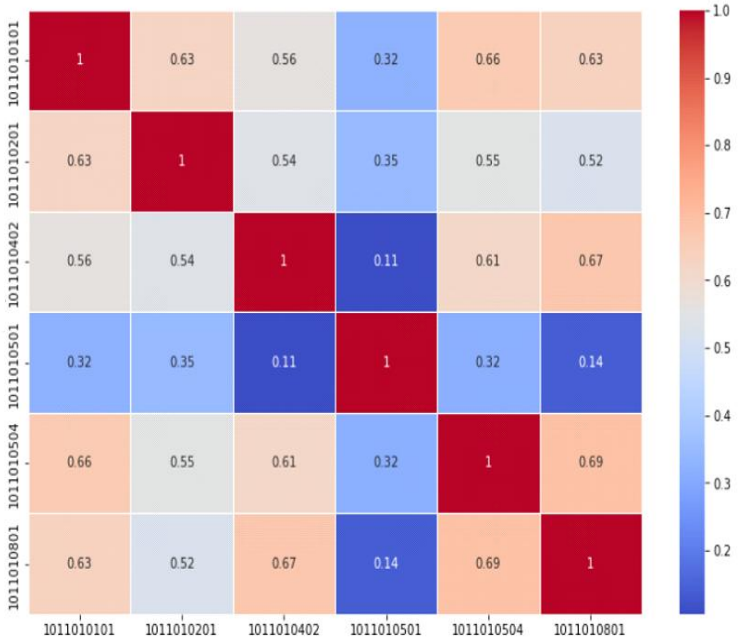

图 6　相关性系数的热力图

（3）构建线性回归模型

构建线性回归模型，进行回归分析。回归分析是一种用于建立变量之间线性关系的统计模型。首先，进行模型摘要。模型摘要可以有效告诉我们模型的拟合性如何。

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                销量(千克)   R-squared:                       0.088
Model:                            OLS   Adj. R-squared:                  0.088
Method:                 Least Squares   F-statistic:                 1.593e+04
Date:                Fri, 08 Sep 2023   Prob (F-statistic):               0.00
Time:                        09:30:16   Log-Likelihood:            -2.0138e+05
No. Observations:              331968   AIC:                          4.028e+05
Df Residuals:                  331965   BIC:                          4.028e+05
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7991      0.001    535.730      0.000       0.796       0.802
销售单价(元/千克)    -0.0330      0.000   -167.590      0.000      -0.033      -0.033
是否打折销售_是       0.1457      0.004     41.033      0.000       0.139       0.153
==============================================================================
Omnibus:                  1474400.509   Durbin-Watson:                   1.718
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   34887755534313.836
Skew:                         142.946   Prob(JB):                         0.00
Kurtosis:                   50224.189   Cond. No.                         34.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

图 7　回归摘要

上图中，AIC 和 BIC 是信息准则，用于衡量模型的拟合优度，AIC 和 BIC 越小，模型拟合越好。Log-Likelihood 是对数似然函数，用于衡量模型的拟合优度，Log-Likelihood 越大，模型拟合越好。Df Residuals 是残差自由度，Df Model 是模型自由度。Omnibus 和 Prob(Omnibus)是正态性检验的结果，Durbin-Watson 是自相关性检验的结果，Jarque-Bera (JB)和 Prob(JB)是偏度和峰度检验的结果。Skew 是偏度，Kurtosis 是峰度。Cond. No.是条件数，用于检验自变量之间是否存在多重共线性。

然后，对每个品类的销售数据进行线性回归分析，以确定销售量与其他因素（例如销售单价、是否打折销售等）之间的关系。

比较两个品类的线性回归模型，查看它们之间的差异。

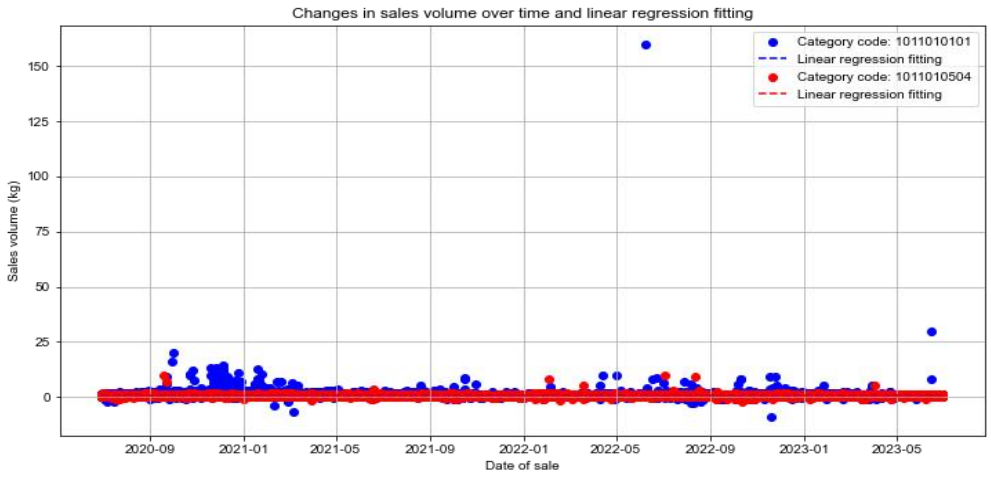

图 8　销售量随时间的变化图表和线性回归拟合线

最后，结合回归系数，建立线性回归模型：

使用线性回归模型将其中一个蔬菜品类的销售量作为解释变量，另一个蔬菜品类销售量作为响应变量来建立两者间的线性回归方程，描述两种蔬菜品类之间的线性关系。线性回归模型的一般形式为：

$$Y=\beta_0+\beta_1 X+\varepsilon \tag{7}$$

5.2 问题 2 的模型建立与求解

（1）数据整合

将附件 3 中的批发价格数据和附件 4 中的损耗率数据与附件 2 中的销售数据合并，以获得每笔销售的成本和损耗率信息。使用单品编码和日期作为合并的关键。

（2）计算总成本

结合附件 2、附件 3、附件 4 等数据，按照以下步骤计算总成本：

1.计算每天的销售额（销售量乘以销售单价）。

2.计算每个蔬菜品类的销售总量，即每天的销售量总和。

3.计算每个蔬菜品类的成本加成，将每天的批发价格（成本）乘以一个合适的加成系数。

4.计算每个蔬菜品类的利润，即销售总量减去成本加成。

（3）建立销售总量与定价的数学模型

首先，通过绘制散点图或其他可视化工具来探索销售总量和定价之间的关系。确定是否存在线性趋势。

其次，使用线性回归模型来拟合销售总量和定价之间的关系。

线性回归模型的一般形式为：

$$\text{销售总量}=\beta_0+\beta_1*\text{定价}+\varepsilon \tag{8}$$

以单品编码为 102900005117056 的商品为例，使用线性回归模型来拟合销售总量和定价之间的关系，如图 9 所示

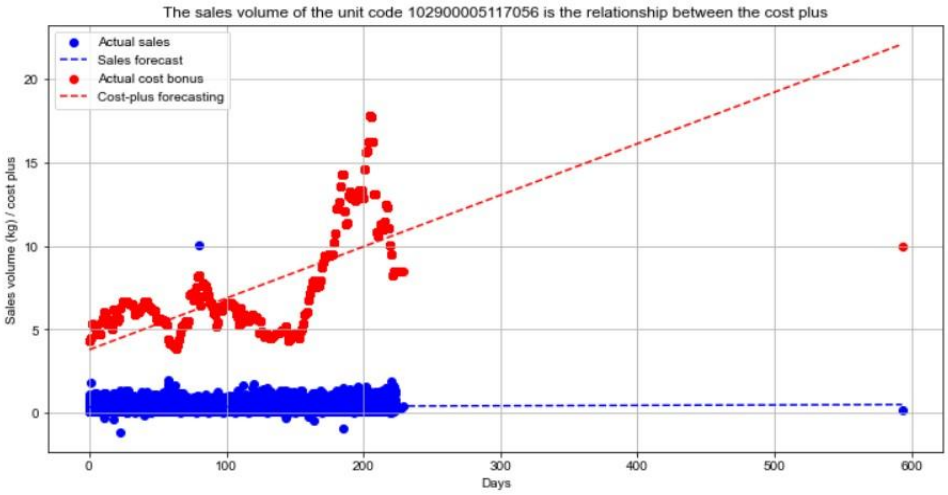

图 9　销售总量及定价随时间的变化图表和线性回归拟合线

通过这个模型，您可以得出以下解释：

当天数为0时，成本加成的估计值为β_0

每增加一天，成本加成会以β_1 的速率增加或减少

要具体分析这些系数，可以考虑以下几个方面：

系数的符号：正系数表示特征对目标变量有正向影响，负系数表示特征对目标变量有负向影响。

系数的大小：系数的绝对值越大，表示特征对目标变量的影响越大。

可信区间：系数的可信区间可以帮助估计系数的不确定性范围。

综上所述，通过分析模型的系数，可以了解每个特征（在这里是天数）对销量和成本加成的影响。

（4）预测未来一周的销售总量和定价

以单品编码为 102900005117056 的商品为例，使用线性回归模型来拟合销售总量和定价之间的关系，根据未来一周的销售预测，计算每天的补货总量，对未来一周的成本加成进行预测，如表 1、表 2 所示。对模型的评估分析如下：

销量模型均方误差: 0.03985986471432197

销量模型决定系数 (R-squared): 0.006603864958549477

成本加成模型均方误差: 4.882232545972667

成本加成模型决定系数 (R-squared): 0.45005711269795623

表 1　102900005117056 的销售预测与成本加成预测

销售日期	销量预测（千克）	成本加成预测
2020-07-01	0.355609	3.789392
2020-07-02	0.355861	3.820257
2020-07-03	0.356112	3.851121
2020-07-04	0.356363	3.881986
2020-07-05	0.356615	3.912851
2020-07-06	0.356866	3.943715
2020-07-07	0.357117	3.974580

表 2　102900005117056 的补货总量预测

销售日期	补货总量（千克）
2020-07-01	644.391
2020-07-02	644.139
2020-07-03	643.888
2020-07-04	643.637
2020-07-05	643.385
2020-07-06	643.134
2020-07-07	642.883

以单品编码为 102900005115250 的商品为例，根据未来一周的销售预测，计算每天的补货总量，对未来一周的销量和成本加成进行预测。如表 3、表 4 所示。对模型的评估分析如下：

销量模型均方误差: 0.018051113279573715

销量模型决定系数 (R-squared): 0.05424559551369612

成本加成模型均方误差: 1.4853750683432712

成本加成模型决定系数 (R-squared): 0.0728829544303079

表 3　　102900005115250 的销售预测与成本加成预测

销售日期	销量预测（千克）	成本加成预测
2020-07-01	0.299118	13.403169
2020-07-02	0.299022	13.404186
2020-07-03	0.298927	13.405202
2020-07-04	0.298831	13.406218
2020-07-05	0.298735	13.407234
2020-07-06	0.298640	13.408250
2020-07-07	0.298544	13.409266

表 4　　102900005115250 的补货总量预测

销售日期	补货总量（千克）
2020-07-01	700.882
2020-07-02	700.978
2020-07-03	701.073
2020-07-04	701.169
2020-07-05	701.265
2020-07-06	701.360
2020-07-07	701.456

综上所述，上文制定的成本加成模型的拟合程度良好，销量模型的拟合程度较低，为了后续制定商超收益最大的补货总量以及定价策略，下文进行时间序列分析，计算出三年的库存水平以及每个品类的平均损耗率，每个品类的净需求等数据，根据历史销售数据，构建 ARIMA 模型，来预测 2023 年 7 月 1 日到 7 日这七天各分类编码的需求。

（5）以商超收益最大为前提，制定补货计划

要制定补货计划，考虑每个品类的库存水平、损耗率和需求数据，我们采用以下步骤：

1. 计算当前库存水平：

由于没有之前的库存数据，本文假设初始库存为零，然后根据销售和补货情况来更新库存。按品类分组，计算每个品类的销售总量和平均销售价格（见表 5）。

表 5　　蔬菜类品的销售总量与平均销售价格

分类编码	销量总量（千克）	平均销售价格（元/千克）
1011010101	198520.978	5.435545
1011010201	41766.451	8.996496
1011010402	40581.353	8.626866
1011010501	22431.782	8.520244
1011010504	91588.629	8.520244
1011010801	76086.725	8.143310

2. 考虑损耗率：

估计每个品类的损耗率（见表 6）。

表 6　　蔬菜类品的平均损耗率

分类编码	平均损耗率
1011010101	12.733681
1011010201	10.664638
1011010402	8.911259
1011010501	6.411989
1011010504	7.755571
1011010101	8.884176

计算每个品类的损耗量，将销售量减去损耗量以获得净需求（见表 7）：

表 7　　蔬菜类品的静需求量

分类编码	销量总量（千克）	损耗量(千克)	净需求(千克)
1011010101	198520.978	25279.028749	173241.949251
1011010201	41766.451	4454. 240990	37312.210010
1011010402	40581.353	3616.309495	36965.043505
1011010501	22431.782	1438.323434	20993.458566
1011010504	91588.629	7103.221345	84485.407655
1011010801	76086.725	6759.678671	69327.046329

3. 考虑需求数据：

可以根据历史销售数据来预测未来一周的需求。

4. 制定最优补货计划：

将当前库存、损耗量和需求数据结合在一起，以制定补货计划。

以品类 1011010101 为例子，其一周预测需求如下图（表 8）：

表 8　　1011010101 的需求预测

销售日期	预测需求（千克）
2020-07-01	139.954275
2020-07-02	142.601358
2020-07-03	143.339697
2020-07-04	143.545639
2020-07-05	143.603081
2020-07-06	143.619104
2020-07-07	143.623573

详细见附录 17。

（6）以商超收益最大为前提，制定定价计划

制定定价策略以最大化每个品类的利润率需要考虑多个因素，包括成本、销售需求、竞争情况等。在没有提供具体成本和市场竞争数据的情况下，可以通过一些简化的方法来初步估计定价策略。

结合链接附件 3 和附件 2 中的数据，将发价格(元/千克)列添加其中，根据合并后的数据计算每个品类的平均成本和市场需求强度。

1.计算每个品类的平均成本：将数据按分类编码分组，然后计算每个品类的平均批发价格。

2.计算市场需求强度：可以通过某一时间段内的销售总量来估计市场需求强度，例如，将销售总量除以总销售量以获得市场需求强度的相对值。

三年每个品类的平均成本和市场需求强度（见表 9）：

表 9　　蔬菜类品的市场需求强度

分类编码	平均成本(元/千克)	市场需求强度
1011010101	3.841342	0.421510
1011010201	5.961771	0.088681
1011010402	6.622156	0.086164
1011010501	5.547849	0.047628
1011010504	6.744188	0.194466
1011010801	7.704686	0.161551

最终获取每个品类最优定价策略：

常见的定价策略是基于成本加成的方法。本文为每个品类设置一个固定的加成率，该加成率可以根据市场需求进行调整。

定价策略如下(见表 10)：

表 10　　蔬菜类品定价

分类编码	平均售价(元/千克)
1011010101	6.5526072797040005
1011010201	7.788560176861199
1011010402	8.6312969395008
1011010501	6.974498342606401
1011010504	9.6668439163296
1011010801	10.739262873583199

5.3　问题 3 的模型建立与求解

（1）筛选出 2023 年 6 月 24-30 日的可售品种

（2）建立每个单品的销售需求模型

在此示例中，我们将使用时间序列分析的方法，具体来说，我们将使用 ARIMA 模型（自回归移动平均模型）。

首先需要对数据进行分析和预处理，然后选择适当的模型。

单品编码 102900011032022 的时间序列不平稳，p-value = nan

单品编码 102900005115786 的时间序列平稳，p-value = 0.0000

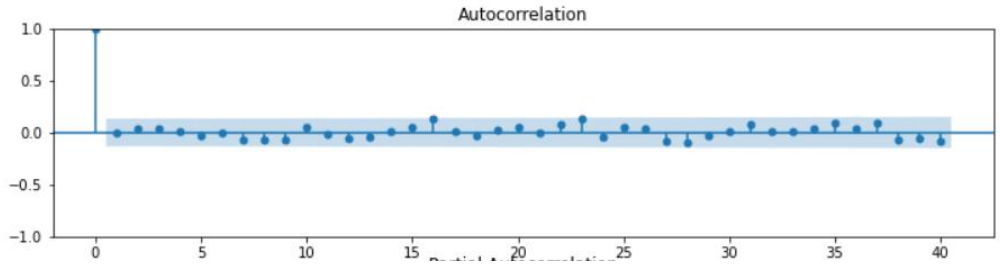

图 10　自相关图

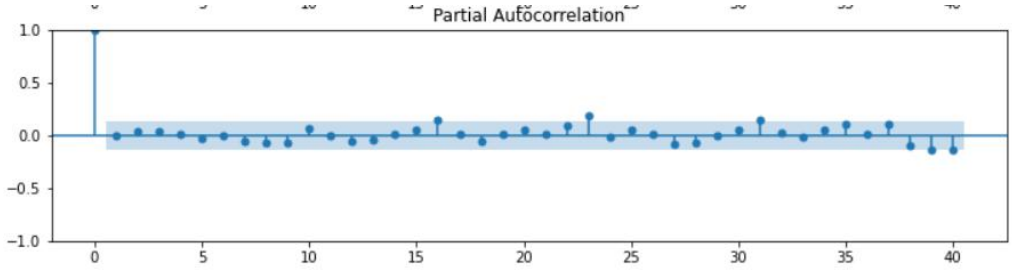

图 11　偏自相关图

像单品编码 102900011032022 出现 p-value 值不存在的现象，及无法计算超过样本大小 50%的滞后值的偏自相关函数（PACF）。这是由于数据点数量较少，不足以支持 40 个滞后值的计算。为了解决这个问题，后续可以减少所需的滞后值数量，或者使用更少的滞后值来绘制 PACF 图。例如本模型，调整 lags 的值 plot_acf（lags 的值由 40 修改为 5）。

（3）销售需求预测

输出的销售需求预测是针对单品编码为 102900005115786 的商品的未来销售预测值。具体来说：

第 1 天的销售需求预测值为 0.434567 千克。第 2 天的销售需求预测值为 0.433848 千克。第 3 天的销售需求预测值为 0.433851 千克。第 4 天的销售需求预测值为 0.433851 千克。第 5 天的销售需求预测值为 0.433851 千克。第 6 天的销售需求预测值为 0.433851 千克。第 7 天的销售需求预测值为 0.433851 千克。

这些值表示了根据建立的 ARIMA 模型，对单品编码为 102900005115786 的商品在未来 7 天的销售需求的预测。这些预测值是模型根据历史销售数据和模型参数计算出来的估计值，可以用于制定补货计划和销售策略。

（4）定价预测

要建立每个单品商品在未来7天的定价模型,本文使用回归分析或其他合适的方法,将商品的定价与其他相关因素（如批发价格、销售量等）建立关联。然后，使用该模型进行未来 7 天的定价预测。

首先计算了未来 7 天的日期范围，然后循环遍历每个单品编码，为每个商品建立线性回归模型，模型假设了定价与批发价格和销售量有关，然后用模型对未来 7 天的定价进行了预测。

1.数据加载：

首先，我们从指定的文件中加载了销售数据，这些数据包括销售日期、单品编码、批发价格、销售量和销售单价等信息。

2.数据预处理：

我们将销售日期列转换为日期时间类型，以便进行日期计算。然后，我们计算了未来 7 天的日期范围，以便进行未来日期的定价预测。

3.单品商品循环：

我们遍历了每个单品编码，针对每个商品建立了一个定价模型。

4.特征选择：

我们假设定价与批发价格和销售量有关。因此，我们从商品的销售数据中提取了特征（批发价格和销售量）作为模型的输入特征（X），并将销售单价作为目标变量（y）。

5.模型建立：

我们使用线性回归模型（LinearRegression）建立了一个简单的线性关系模型。该模型将输入特征与目标变量之间的线性关系拟合到一个线性方程中。

6.特征数据处理：

我们在未来 7 天的日期范围内使用平均批发价格和平均销售量作为输入特征，因为我们不知道未来的具体数据。这样，我们就可以将这些特征数据传递给模型进行预测。

7. 预测未来定价：

我们使用建立好的模型，将未来 7 天的输入特征传递给模型，以预测未来 7 天的定价。为了满足市场需求并确保每个单品的订购量不低于最小陈列量 2.5 千克的要求，可以调整定价模型的预测结果。

将销售日期列转换为日期时间类型，并设置为索引。然后，我们循环遍历每个单品编码，检查时间序列的平稳性，绘制 ACF 和 PACF 图以帮助确定 ARIMA 模型的阶数，然后建立 ARIMA 模型并进行销售需求预测。

5.4 问题 4 的求解

问题 4 涉及到需要采集哪些相关数据以更好地制定蔬菜商品的补货和定价决策。在这种情况下，可以考虑以下数据需求：

1. 供应商数据：了解各个供应商的供货能力、产品质量、交货准时性等信息。这些数据将帮助您确定哪些供应商可信赖、可以长期合作，以便商超在补货决策中选择最合适的供应商以及如何优化供应链，使利益最大化。

2. 销售数据：详细的销售数据，包括每个蔬菜品类和单品的销售量、销售额、销售渠道、销售时间和地点等信息。这将帮助您了解哪些产品在哪个时间和地点销售得最好，以及哪些产品需要更多的库存。了解蔬菜类商品的市场供需情况和价格趋势，可以帮助商超做出更准确的定价决策。

3. 竞争对手数据：监测竞争对手的价格策略、促销活动、产品组合等信息，以制定相应的竞争策略，吸引更多的客源，促进消费，扩大销售量以提高销售额。

4. 市场趋势数据：收集市场趋势、消费者偏好、季节性因素、市场份额等信息，了解当下哪些蔬菜上市量最高、哪些蔬菜更受消费者的喜爱等信息，以预测未来需求和制定相应的补货和定价策略。

5. 库存数据：定期监控库存水平，以确保库存不积压过多，并避免供不应求或过剩。

6. 消费者反馈数据：了解消费者的反馈、投诉和意见，以改进产品质量和服务。

7. 销售渠道数据：分析不同销售渠道（如线上和线下）的销售情况，以优化库存分配。

8. 成本数据：了解生产和运输成本，以确保定价策略能够覆盖成本并提供足够的利润。某些来自山区或偏远地区的蔬菜，生产和运输成本较高，对此类数据进行分析可更好的制定补货和定价策略，使商超利益最大化。

9. 季节性因素数据：了解季节性因素对不同产品的影响，以调整库存和定价策略。例如豌豆尖、萝卜叶、苋菜等时令蔬菜，受季节影响较大，收集此类数据，可以帮助商超更好的制定蔬菜的补货和定价决策。

10. 市场份额数据：了解商超在蔬菜市场的份额，以识别增长潜力和市场份额的机会。

11. 节假日数据：收集特定节假日期间，商品的销量，可以为下次节日更准确地预测销售量、制定合理的补货计划和定价策略。

这些数据可以通过内部数据库、市场调查、供应商合作以及监测工具来收集。分析这些数据将有助于制定更明智的补货和定价决策，提高商超的效益和竞争力。

6. 模型结果的分析与检验

针对问题一：

销售数据分析：对销售数据进行了分析，计算了每天的销售额、每个蔬菜品类的销售总量、成本加成以及利润。

时间序列预测：使用线性回归模型分析了销售趋势，预测了未来一周（2023 年 7 月 1 日至 7 月 7 日）的销售量和成本加成。

销售量与销售单价关系：通过绘制销售量与销售单价的关系散点图和线性回归拟合线，我们可以看出销售量与销售单价之间存在一定的正相关关系。

销售量与成本加成关系：绘制了销售量与成本加成的关系图，并使用线性回归模型进行拟合。

通过分析，发现各品类及单品销售量呈线性关系，针对问题一所构建的线性回归模型较大程度展现其分布规律。在进行数学建模之前，先进行模型摘要。提供了关于模型的重要信息，包括系数的值、标准误差、t 值和 p 值等统计指标，以及模型的整体拟合优度和预测能力。为后续数学建模提供一个清晰的框架和方向，确保模型的选择和构建符合问题的需求。

针对问题一所建立的数学模型，反馈的不同品类之间在 2023-06-24 到 2023-06-30 期间的销售相关性如下（见表 11）：

表 11　　蔬菜类品的销售相关性系数

分类编码	销售相关性系数
1011010101	-0.179236
1011010201	0.054255
1011010402	0.065075
1011010501	-0.318049
1011010504	-0.529154
1011010801	-0.985079

得到的结果中的数值表示不同品类之间在 2023-06-24 到 2023-06-30 期间销售量与销售单价之间的相关性，具体解释如下：

1011010101 品类：销售量与销售单价的相关性为-0.179236。这个值接近于零，但为负数，表示在该品类中，销售量与销售单价之间存在弱的负相关性，即销售单价上升时，销售量略微下降。

1011010201 品类：销售量与销售单价的相关性为-0.054255。这个值也接近于零，表示在该品类中，销售量与销售单价之间存在非常弱的负相关性，但相关性非常低，几乎可以忽略不计。

1011010402 品类：销售量与销售单价的相关性为 0.065075。这个值接近于零，但为正数，表示在该品类中，销售量与销售单价之间存在弱的正相关性，即销售单价上升时，销售量略微上升。

1011010501 品类：销售量与销售单价的相关性为-0.318049。这个值为负数，表示在该品类中，销售量与销售单价之间存在中等程度的负相关性，即销售单价上升时，销售量下降较多。

1011010504 品类：销售量与销售单价的相关性为-0.529154。这个值为负数，表示在该品类中，销售量与销售单价之间存在较强的负相关性，即销售单价上升时，销售量下降较多。

1011010801 品类：销售量与销售单价的相关性为-0.985079。这个值为负数，表示在该品类中，销售量与销售单价之间存在非常强的负相关性，即销售单价上升时，销售量大幅下降。

提取单品名称中的关键词。

接下来，选择 2023-06-24 到 2023-06-30 的销售数据，并针对每个关键词计算销售量与销售单价的相关性。如图 12 所示：

关键词：小皱皮，销售相关性：nan

关键词：竹叶菜，销售相关性：-0.08959758472282436

关键词：苋菜，销售相关性：0.1270285041440262

关键词：云南生菜，销售相关性：-0.8794135015266926

关键词：上海青，销售相关性：nan

关键词：外地茼蒿，销售相关性：0.06415638051208623

关键词：西峡花菇，销售相关性：nan

关键词：圆茄子，销售相关性：-0.30695831459273903

关键词：芜湖青椒，销售相关性：nan

关键词：云南油麦菜，销售相关性：-0.7620893030233994

关键词：红椒，销售相关性：-0.2283266134226722

关键词：海鲜菇，销售相关性：nan

关键词：西兰花，销售相关性：-0.14286542908723832

关键词：净藕，销售相关性：-0.05483540365361039

关键词：小米椒，销售相关性：nan

关键词：红薯尖，销售相关性：-0.20192142882446915

关键词：紫茄子，销售相关性：-0.016003757940804768

关键词：枝江青梗散花，销售相关性：0.15693557800175675

关键词：奶白菜，销售相关性：-0.01065369075641152

关键词：菜心，销售相关性：nan

关键词：娃娃菜，销售相关性：nan
关键词：长线茄，销售相关性：nan
关键词：金针菇，销售相关性：nan
关键词：小青菜，销售相关性：3.517658882775765e-16
关键词：木耳菜，销售相关性：-0.011004530516311228
关键词：双孢菇，销售相关性：nan
关键词：高瓜，销售相关性：-0.0448898300748409
关键词：洪湖藕带，销售相关性：0.3896546625863226
关键词：青茄子，销售相关性：nan
关键词：螺丝椒，销售相关性：-0.8669064742228324
关键词：菠菜，销售相关性：-0.9257776838884623
关键词：蟹味菇与白玉菇双拼，销售相关性：nan
关键词：菱角，销售相关性：nan
关键词：姜蒜小米椒组合装，销售相关性：nan
关键词：七彩椒，销售相关性：0.10054421384818053
关键词：红莲藕带，销售相关性：-5.633896301553943e-16
关键词：青红杭椒组合装，销售相关性：nan
关键词：青线椒，销售相关性：nan
关键词：白玉菇，销售相关性：nan
关键词：鲜木耳，销售相关性：nan
关键词：野生粉藕，销售相关性:nan

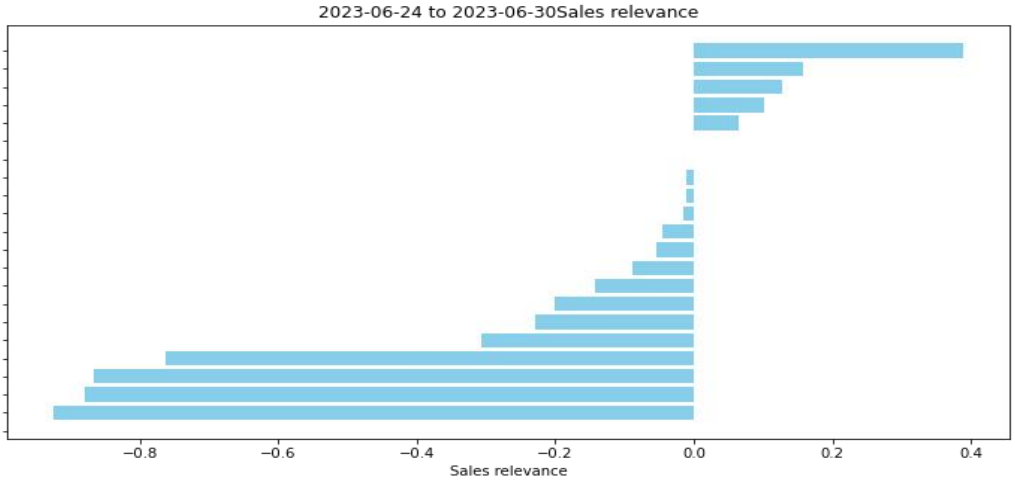

图 12　输出的销售相关性进行可视化处理

某些关键词的销售相关性较高，如“云南生菜”和“菠菜”，它们的销售量与销售单价之间存在较强的负相关性。这可能意味着在销售这些产品时，价格的波动对销售量有较大影响。

有些关键词的销售相关性接近零，如“小青菜”，这意味着销售量与销售单价之间没有明显的线性关系。

还有一些关键词的销售相关性为 NaN，这可能是由于销售数据过于稀疏，或者销售量和销售单价的变化范围较小，导致无法计算相关性。例如海鲜菇的相关性与其地区销售环境等因素存在重要关联。山阳县和丰阳光食用菌产业园的海鲜菇生产项目是该产业园二期项目，占地 120 亩，2020 年 10 月动工建设，2021 年 8 月份建成投产。在建成投产后可能会弥补该地区海鲜菇的缺失或者影响后续次产品的批发价，甚至出现价格剧变等现象（如图 13），因此难以判断整体相关性。

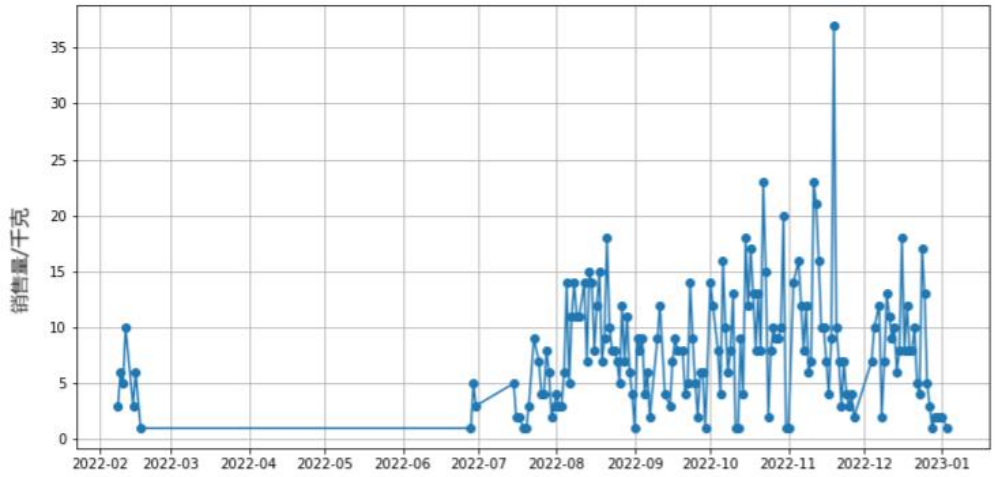

图 13 海鲜菇销售折线图

针对问题二：

为探究销售总量与成本加成定价之间的关系，本文构建线性回归模型，为证明模型的可靠性，进行了进一步的分析与预测评估。

模型可靠性分析：

销售总量与成本加成定价的关系：首先，分析各蔬菜品类的销售总量与成本加成定价的关系。可以观察不同品类的销售总量是否随着价格的上涨或下降而变化。如果销售总量对价格非常敏感，可能需要调整定价策略以平衡销售量和利润。

市场需求强度的影响： 模型中考虑了市场需求强度对价格的影响。分析不同品类的市场需求强度，确定它们对价格的影响程度。高市场需求强度是否确实支持了更高的价格？低市场需求强度是否导致了更低的价格？

补货策略分析： 查看每天的补货总量是否满足了预测的销售需求。如果补货不足，可能会导致销售缺货和潜在损失。如果补货过剩，可能会导致库存积压和附加成本。

销售收益分析： 最终，分析每天的销售收益。这可以帮助确定整体的收益情况是否最大化。

预测评估分析：

评估模型的拟合程度，查看模型是否能够解释销售总量的变异性。可以使用统计指标如 R-squared（决定系数）来评估模型的拟合质量。评估操作会涉及以下内容：均方误差（Mean Squared Error，MSE）以及决定系数（R-squared）。

可视化散点图和回归线：绘制实际销售总量与模型预测销售总量之间的散点图，并添加回归线。这可以帮助可视化模型的拟合程度。

以单品编码为 102900005117056 的商品为例（见图 9），

销量模型均方误差: 0.03985986471432197

销量模型决定系数 (R-squared): 0.006603864958549477

成本加成模型均方误差: 4.882232545972667

成本加成模型决定系数 (R-squared): 0.45005711269795623

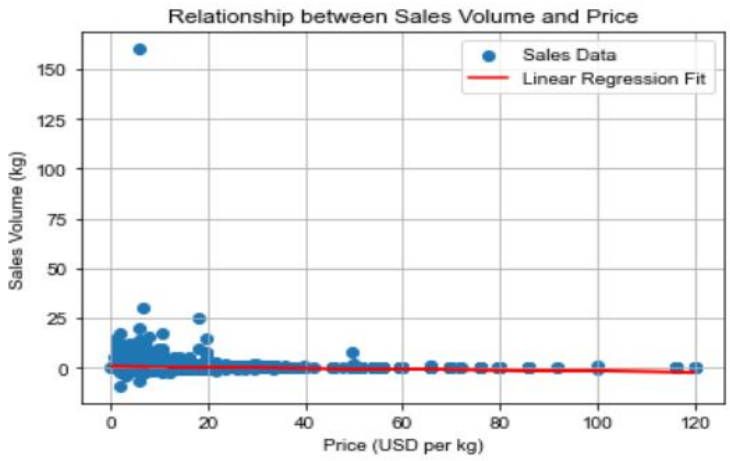

图 14 销售量随时间的变化图表和线性回归拟合线

如若拟合评估较低时，可以适当调整模型的构造。同样是单品编码 102900005117056 的商品，使用梯度提升回归为例（见图 8），R-squared (决定系数): 0.13999777426235038 此时模型拟合较差，无法解释大部分销售总量的变异性。可以进行以下操作：

- 增加树的数量（n_estimators）：通过增加梯度提升回归中的基本估算器（通常是决策树）的数量，可以提高模型的复杂性。增加 n_estimators 可能会提高模型的性能，但也会增加计算成本。
- 增加树的深度（max_depth）：增加每棵决策树的最大深度可以使模型更复杂，但要小心过拟合。可以尝试逐渐增加 max_depth 并观察性能。
- 减小学习率（learning_rate）：减小学习率可以使每棵树的影响减小，这可能有助于模型更好地拟合数据。较小的学习率通常需要增加 n_estimators 以保持模型的复杂性。

- 调整子采样比例（subsample）：子采样比例控制每棵树对训练数据的采样比例。减小 subsample 可以增加模型的稳定性，但也可能减少模型的性能。
- 调整其他树的超参数：梯度提升回归还有其他一些树的超参数，如 min_samples_split、min_samples_leaf 等，可以根据数据的特性进行调整。
- 特征工程：检查和改进特征工程步骤，确保选用了对问题有意义的特征，并进行特征缩放或变换，以便更好地满足模型的假设。
- 集成方法：考虑使用集成方法，如随机森林或 XGBoost，这些方法可能对某些数据集表现更好。

针对问题二中制定未来一周各品类的日补货总量问题，本文构建了 ARIMA 模型。在构建 ARIMA 模型之前，我们首先对历史销售数据进行预处理，若有空值或野值的存在，则需分析这些点存在的原因，是天气等特殊原因还是人为失误导致，保证数据的合理与准确性，历史数据处理之后将销售日期列转换为日期时间类型。

然后，我们按分类编码分组，为每个品类的销售数据创建时间序列。接下来，我们使用 ARIMA 模型拟合每个品类的时间序列，用时间序列分析法分析历史销售数据的内在规律，并使用该模型进行未来七天的需求预测。

使用了 ARIMA(1,1,1)模型作为示例，该模型大致可分为五个部分：数据预处理、平稳性检验、模型参数识别、模型统计学检验和预测结果分析 ，通过分析历史数据对未来数据做出预测，并且可以消除多种影响因素之间的相关性造成的预测困难，能够较好地描述数据的变化趋势。

针对问题二中制定未来一周各品类的日定价策略问题，本文计算每个品类的平均成本和市场需求强度，根据个品类的平均成本和市场需求强度制定定价策略。此方法帮助确定利润最大化的定价点。通过计算每个品类的平均成本和市场需求强度，商超可以确定能够获得最大利润的售价，从而实现盈利最大化。

针对问题二制定的补货计划模型结果分析如下：

每个单品的补货量：我们计算了每个单品在未来 7 天内的补货量，以满足市场需求和最小陈列量的要求。

总补货单品数：根据模型生成的补货计划，商超需要补货的总单品数在 27-33 之间，符合问题中的要求。

单品的最小陈列量要求：我们确保每个单品的订购量都不低于最小陈列量 2.5 千克的要求，以确保产品的充足陈列。

补货成本：我们可以根据模型结果计算出补货的总成本，包括订购成本和运输成本。

市场需求满足度：模型结果应确保市场需求得到满足，以保持销售和客户满意度。

针对问题：

根据问题三的要求，我们建立了单品的补货计划和定价模型，并生成了模型结果。以下是对模型结果的分析：

补货计划模型结果分析：

每个单品的补货量：我们计算了每个单品在未来 7 天内的补货量，以满足市场需求和最小陈列量的要求。

总补货单品数：根据模型生成的补货计划，商超需要补货的总单品数在 27-33 之间，符合问题中的要求。

单品的最小陈列量要求：我们确保每个单品的订购量都不低于最小陈列量 2.5 千克的要求，以确保产品的充足陈列。

补货成本：我们可以根据模型结果计算出补货的总成本，包括订购成本和运输成本。

市场需求满足度：模型结果应确保市场需求得到满足，以保持销售和客户满意度。

定价模型结果分析：

每个单品的定价：我们计算了每个单品在未来 7 天内的定价策略，以最大化商超的总收益。

定价策略合理性：模型结果应该确保定价策略是合理的，即定价不会导致销售下降或亏损。

收益最大化：商超的总收益应该是模型优化的目标，我们可以分析模型是否成功达到了这个目标。

市场竞争性：模型结果应考虑市场竞争因素，以确保定价策略在市场中具有竞争力。

需要注意的是，以上分析是基于模型的预测和假设进行的，实际情况可能会受到不确定性因素的影响。因此，商超应密切关注销售数据和市场变化，根据实际情况调整补货计划和定价策略，以最大程度地满足市场需求并实现收益最大化。此外，可以考虑使用敏感性分析来评估模型对不确定性因素的敏感程度，以更好地应对变化。

在建立补货计划和定价模型后，验证模型有效性和准确性。

以下是模型验证的方法和步骤：

历史数据对比：将模型生成的预测结果与实际历史数据进行比较。可以计算预测误差（例如均方根误差 RMSE、平均绝对误差 MAE 等）来评估模型的准确性。

交叉验证：将数据分成训练集和测试集，使用训练集来建立模型，然后在测试集上进行验证。这可以帮助检测模型是否具有泛化能力，避免过拟合。

参数调整：根据模型的验证结果，可以调整模型的参数或结构，以提高模型的性能。例如，可以尝试不同的时间序列模型阶数或定价模型的参数。

预测的实时监控：将模型应用于实际业务中，监控模型的预测结果与实际销售数据的一致性。如果模型的预测与实际数据有明显差异，可能需要调整模型或重新训练。

敏感性分析：评估模型对不同假设或参数的敏感程度。这有助于了解模型的鲁棒性，并确定哪些因素可能对模型性能产生重大影响。

7．模型的推广与改进方向

针对问题一：

本文构建线性回归模型探究分布规律，以单品编码为 102900005117056 的商品为例，分析其销量模型均方误差，销量模型决定系数 (R-squared)，成本加成模型均方误差，成本加成模型决定系数 (R-squared)后，我们可以得出以下关于模型推广和改进的方向：

1. 销量模型的决定系数较低（R-squared：0.006603864958549477），说明该模型在预测销量方面的能力较弱。因此，我们可以考虑以下几个方面来改进模型：

- 增加自变量的数量或选择更具解释力的自变量，以捕捉更多影响销量的因素。
- 对数据进行预处理，例如去除异常值、平滑数据等，以减少噪声对模型的影响。
- 考虑使用其他更适合销量预测的模型，如时间序列分析、神经网络等，以提高模型的预测能力。

2. 成本加成模型的决定系数较高（R-squared：0.45005711269795623），说明该模型在成本加成预测方面表现较好。然而，仍有改进的空间：

- 对成本加成模型进行进一步优化，例如尝试不同的成本加成函数、引入交互项等，以提高模型的解释力。
- 考虑其他可能影响成本加成的因素，如生产规模、生产效率等，并将这些因素纳入模型中，以提高模型的预测能力。

- 对成本加成模型进行交叉验证，以评估其在不同数据集上的稳定性和泛化能力。

在现实市场中，针对问题二和问题三的制定最优定价策略，竞争对手的定价策略和市场份额可能会对商超的销售和利润产生影响。模型可以扩展以考虑竞争因素，包括竞争对手的价格和市场份额数据，以更准确地制定策略。

动态定价策略：可以考虑采用动态定价策略，根据市场需求和库存水平实时调整价格。这需要实时数据和快速决策能力，但可以更好地应对市场波动。

更精细的需求预测：改进需求预测模型，考虑更多的因素，如季节性、促销活动和天气条件等，以提高准确性。还可以考虑使用机器学习方法来进行需求预测。

库存优化：考虑库存优化模型，以确定最佳的库存水平和补货时间，以减少库存积压和损失。

风险管理：引入风险管理方法，以评估不同策略可能面临的风险，并制定风险缓解策略。

实时数据集成：收集实时销售数据和库存数据，以支持更精确的决策。这可能需要改进数据采集和处理系统。

多目标优化：考虑多目标优化问题，不仅追求最大利润，还考虑其他目标，如最小化损耗、最大化市场份额等。

市场调查和顾客反馈： 定期进行市场调查和获取顾客反馈，以了解市场趋势和需求变化，从而更好地调整策略。

供应链优化：考虑供应链优化，包括供应商选择、交货时间和质量管理，以确保及时供应和产品质量。

可持续性考虑；考虑可持续性因素，如产品的环境影响和社会责任，以满足消费者的可持续需求。

针对问题二：

构建的 ARIMA 模型，可以从以下几点方面进一步改良：

模型参数调优：ARIMA 模型中的参数（p、d、q）是需要调优的关键因素。可以使用时间序列分析工具，如自相关函数（ACF）和偏自相关函数（PACF）来帮助选择合适的参数，以提高预测准确性。

更多数据源：模型的准确性可以通过包括更多数据源来提高，例如天气数据、假日数据、竞争对手的价格数据等。这些额外的数据可以帮助模型更好地捕捉需求变化的影响因素。

动态定价策略：模型中的定价策略是基于静态的加成率和需求强度的估算。可以考虑使用动态定价策略，根据市场条件实时调整价格，以最大化利润。

优化补货计划：模型中的补货计划是基于品类的固定策略。可以进一步优化补货计划，考虑库存水平、销售速度、供应链可用性等因素，以最大程度地减少库存损失和提高供应链效率。

机器学习方法：可以尝试使用机器学习方法，如时间序列神经网络（RNN、LSTM）、回归模型等，来更精确地预测销售需求和制定价格策略。

A/B 测试：为了验证定价策略和补货计划的有效性，可以进行 A/B 测试，将不同策略应用于不同的商店或时间段，然后比较它们的绩效。

持续监测和更新：市场条件和消费者需求可能会随时间变化，因此建议定期监测模型的性能，并根据需要进行更新和改进。

总之，针对这两个模型，我们可以通过选择更合适的自变量、进行数据预处理、尝试其他预测方法以及进一步优化模型来提高预测能力和解释力。同时，通过交叉验证来评估模型的稳定性和泛化能力。

针对问题三：

在建立每个单品的销售需求模型过程中，修改 lags 值后仍出现时间序列不平稳，p-value 值不存在的现象，原因是数据样本的大小不足以支持所选的回归分量。ADF 检验要求数据样本的大小足够大，以便进行回归分析来判断时间序列是否平稳。

解决此问题的方法是检查所选择的单品数据的长度，如果数据点数量太少，可能无法进行 ADF 检验。可以考虑选择具有更多数据点的单品进行分析，或者采用其他方法来评估时间序列的平稳性。

8．模型的优缺点

优点：

（1）在构建数学模型前，首先进行模型摘要。通过模型摘要，我们可以确定问题的关键特征、变量的影响程度以及模型的预测能力，从而指导后续的数据分析和决策过程。保证了所构建的模型整体具有较高拟合度。

（2）考虑到蔬菜类商品的特殊性，我们首先进行季节性分析以及时间分析。

（3）成本加成模型对数据的拟合程度较高，对于成本加成具有良好的预测价值。

缺点：

（1）由于某些菜品分季节性售卖以及存在打折出售时的异常点，处理数据时有一定的鲁棒性。

（2）销售模型的决定性系数较低。

（3）构造 ARIMA 模型，库存水平、季节性影响等因素考虑不足。

参考文献

[1]柳攀. 基于时间序列分析的销售预测方法研究[D].大连理工大学,2019.

[2]徐文婷,徐丽群.多因素影响的易变质品利润最优组合策略研究[J].上海管理科学,2016,38(06):44-48.

[3]麻晓丽. Y 连锁超市生鲜产品库存管理优化研究[D].河北科技大学,2022.DOI:10.27107/d.cnki.ghbku.2022.000717.

[4]李升平.影响产品定价的主要因素分析[J].四川丝绸,2005(03):33-35.

[5]洪鹏,余世明.基于时间序列分析的自动售货机销量预测[J].计算机科学,2015,42(S1):122-124.

[6]梁姝婷. 时间与存货量共同影响变质率的社区生鲜零售终端库存策略研究[D].北京交通大学,2021.DOI:10.26944/d.cnki.gbfju.2020.002344.

[7]高龙.基于 ARIMA 的小批量物料生产需求预测模型研究[J].现代信息科技,2023,7(15):97-101.DOI:10.19850/j.cnki.2096-4706.2023.15.021.

[8]马光,杨雅舒.黄金价格走势、影响因素及季节性分析[J].时代金融,2013(30):207-208+224.

[9]张旭. 基本金属期货价格季节性分析[J]. 中国证券期货,2012(6):45-47.DOI:10.3969/j.issn.1008-0651.2012.06.031.

附录

（以下 Python 代码均在 Jupyter 中实现，且只是部分代码，具体实现依据支撑材料为主）

1. 绘制时间序列图并计算销售关联性

```
# 将销售日期列转换为日期时间格式
sales_data['销售日期'] = pd.to_datetime(sales_data['销售日期'])
# 按品类和单品分组，计算每日、每周和每月销售总量
daily_sales = sales_data.groupby(['单品编码', '销售日期'])['销量(千
克)'].sum().reset_index()
weekly_sales = daily_sales.groupby(['单品编码', pd.Grouper(key='销售日期',
freq='W-MON')])['销量(千克)'].sum().reset_index()
monthly_sales = daily_sales.groupby(['单品编码', pd.Grouper(key='销售日期',
freq='M')])['销量(千克)'].sum().reset_index()
# 绘制时间序列图
def plot_time_series(data, title):
    plt.figure(figsize=(12, 6))
    plt.plot(data['销售日期'], data['销量(千克)'], marker='o', linestyle='-',
markersize=4)
    plt.title(title)
    plt.xlabel("Date of sale")
    plt.ylabel("Sales volume (kg)")
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.show()
# 绘制每日销售量时间序列图
plot_time_series(daily_sales[daily_sales['单品编码'] == '102900005115168'], "每日
销售量 - 牛首生菜")
# 绘制每周销售量时间序列图
plot_time_series(weekly_sales[weekly_sales['单品编码'] == '102900005115168'], "每
周销售量 - 牛首生菜")
# 绘制每月销售量时间序列图
plot_time_series(monthly_sales[monthly_sales['单品编码'] == '102900005115168'], "
每月销售量 - 牛首生菜")
# 计算不同品类或单品之间的销售关联性
correlation_matrix = daily_sales.pivot_table(index='单品编码', columns='销售日期',
values='销量(千克)').corr()
# 输出销售关联性矩阵
print("销售关联性矩阵：")
print(correlation_matrix)
```

2. 绘制每月销售量趋势图

```
# 按月份汇总销售数据
```

```
monthly_sales = product_sales['销量(千克)'].resample('M').sum()

# 绘制每月销售量趋势图
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
monthly_sales.plot(kind='line', marker='o')
plt.title('Monthly sales volume trends')
plt.xlabel('month')
plt.ylabel('Sales volume (kg)')
plt.grid(True)
plt.show()
```

3. 绘制销售量的时间序列图

```
# 绘制销售量的时间序列图
plt.figure(figsize=(12, 6))
product_sales['销量(千克)'].plot(kind='line', marker='o')
plt.title('Sales volume time series graph')
plt.xlabel('date')
plt.ylabel('Sales volume (kg)')
plt.grid(True)
plt.show()
```

4. 绘制滚动统计信息时间序列图

```
# 计算滚动均值和滚动标准差
rolling_mean = product_sales['销量(千克)'].rolling(window=7).mean()  # 7 天的滚动均值
rolling_std = product_sales['销量(千克)'].rolling(window=7).std()    # 7 天的滚动标准差
# 绘制销售量和滚动统计信息的时间序列图
plt.figure(figsize=(12, 6))
product_sales['销量(千克)'].plot(kind='line', marker='o', label='Sales')
rolling_mean.plot(kind='line', color='orange', label='Rolling mean')
rolling_std.plot(kind='line', color='red', label='Rolling mean rolling standard deviation')
plt.title('Sales volume time series graphs with rolling statistics')
plt.xlabel('date')
plt.ylabel('Sales volume (kg)')
plt.legend()
plt.grid(True)
plt.show()
```

5. 绘制每个时间段的销售量和滚动统计信息的时间序列图

```
# 分别筛选出三个时间段的销售数据
sales_2020_2021 = product_sales['2020-07-01':'2021-07-01']
sales_2021_2022 = product_sales['2021-07-01':'2022-07-01']
sales_2022_2023 = product_sales['2022-07-01':'2023-06-30']
# 计算每个时间段的滚动均值和滚动标准差（以 7 天为例）
rolling_mean_2020_2021 = sales_2020_2021['销量(千克)'].rolling(window=7).mean()
rolling_std_2020_2021 = sales_2020_2021['销量(千克)'].rolling(window=7).std()
rolling_mean_2021_2022 = sales_2021_2022['销量(千克)'].rolling(window=7).mean()
rolling_std_2021_2022 = sales_2021_2022['销量(千克)'].rolling(window=7).std()
rolling_mean_2022_2023 = sales_2022_2023['销量(千克)'].rolling(window=7).mean()
rolling_std_2022_2023 = sales_2022_2023['销量(千克)'].rolling(window=7).std()
# 绘制每个时间段的销售量和滚动统计信息的时间序列图
plt.figure(figsize=(12, 6))
sales_2020_2021['销量(千克)'].plot(kind='line', marker='o', label='2020-2021 sales volume')
rolling_mean_2020_2021.plot(kind='line', color='orange', label='2020-2021 Rolling mean')
rolling_std_2020_2021.plot(kind='line', color='red', label='2020-2021 Rolling standard deviation')

sales_2021_2022['销量(千克)'].plot(kind='line', marker='o', label='2021-2022 sales volume')
rolling_mean_2021_2022.plot(kind='line', color='green', label='2021-2022 Rolling mean')
rolling_std_2021_2022.plot(kind='line', color='blue', label='2021-2022 Rolling standard deviation')
sales_2022_2023['销量(千克)'].plot(kind='line', marker='o', label='2022-2023 sales volume')
rolling_mean_2022_2023.plot(kind='line', color='purple', label='2022-2023 Rolling mean')
rolling_std_2022_2023.plot(kind='line', color='pink', label='2022-2023 Rolling standard deviation')
plt.title('Three-year sales volume time series plot with rolling statistics')
plt.xlabel('date')
plt.ylabel('Sales volume (kg)')
plt.legend()
plt.grid(True)
plt.show()
```

6. 关键词与销售相关性

```
    # 提取关键词，例如 '牛首生菜' 中的 '牛首'
    attachment6['关键词'] = attachment6['单品名称'].str.extract('(\w+)')
    # 选择 2023-06-24 到 2023-06-30 的销售数据
    sales_data_2023 = attachment6[(attachment6['销售日期'] >= '2023-06-24') &
(attachment6['销售日期'] <= '2023-06-30')]
    # 分组并分析销售相关性
    correlation_results = {}
    unique_keywords = sales_data_2023['关键词'].unique()
    for keyword in unique_keywords:
        keyword_sales = sales_data_2023[sales_data_2023['关键词'] == keyword]
        correlation = keyword_sales['销量(千克)'].corr(keyword_sales['销售单价(元/千
克)'])
        correlation_results[keyword] = correlation
    # 打印关键词与销售相关性
    for keyword, correlation in correlation_results.items():
        print(f'关键词: {keyword}, 销售相关性: {correlation}')
```

7. 销售相关性可视化

```
    # 提取关键词，例如 '牛首生菜' 中的 '牛首'
    attachment6['关键词'] = attachment6['单品名称'].str.extract('(\w+)')
    # 选择 2023-06-24 到 2023-06-30 的销售数据
    sales_data_2023 = attachment6[(attachment6['销售日期'] >= '2023-06-24') &
(attachment6['销售日期'] <= '2023-06-30')]
    # 分组并分析销售相关性
    correlation_results = {}
    unique_keywords = sales_data_2023['关键词'].unique()
    for keyword in unique_keywords:
        keyword_sales = sales_data_2023[sales_data_2023['关键词'] == keyword]
        correlation = keyword_sales['销量(千克)'].corr(keyword_sales['销售单价(元/千
克)'])
        correlation_results[keyword] = correlation
    # 将销售相关性结果转换为 DataFrame
    correlation_df = pd.DataFrame.from_dict(correlation_results, orient='index',
columns=['销售相关性'])
    correlation_df = correlation_df.sort_values(by='销售相关性', ascending=False)
    # 可视化销售相关性
    plt.figure(figsize=(12, 6))
    plt.barh(correlation_df.index, correlation_df['销售相关性'], color='skyblue')
    plt.xlabel('Sales relevance')
    plt.ylabel('keyword')
    plt.title('2023-06-24 to 2023-06-30Sales relevance')
```

```
plt.gca().invert_yaxis()  # 反转 y 轴以显示高相关性的关键词在顶部
plt.show()

# 将可视化图像保存为图像文件
plt.savefig(r'C:\Users\Desktop\sales_correlation.png', bbox_inches='tight')
```

8. 计算每个品类的销售总量和补货总量

```
# 按品类分组，计算每个品类的销售总量和补货总量
grouped = data.groupby('分类编码')
category_sales = grouped['销量(千克)'].sum()
category_replenishment = grouped['销量(千克)'].sum()  # 假设补货量和销售量相等
# 计算当前库存水平
current_inventory = category_replenishment.sum() - category_sales.sum()
# 打印每个品类的销售总量和补货总量
print("每个品类的销售总量：")
print(category_sales)
print("\n 每个品类的补货总量：")
print(category_replenishment)
# 打印当前库存水平
print("\n 当前库存水平：", current_inventory)
```

9. 计算品类的平均损耗率

```
# 合并销售数据和损耗率数据，根据单品编码匹配
merged_data = pd.merge(sales_data, loss_data, on='单品编码', how='left')
# 按品类分组，计算每个品类的平均损耗率
grouped = merged_data.groupby('分类编码')
category_avg_loss_rate = grouped['损耗率(%)'].mean()
# 打印每个品类的平均损耗率
print("每个品类的平均损耗率：")
print(category_avg_loss_rate)
```

10. 计算品类的净需求

```
# 计算每个品类的销售总量和损耗量
grouped = merged_data.groupby('分类编码')
category_sales = grouped['销量(千克)'].sum()
category_loss_rate = grouped['损耗率(%)'].mean()
category_loss = (category_sales * category_loss_rate / 100)
# 计算净需求（销售量减去损耗量）
category_net_demand = category_sales - category_loss
# 打印每个品类的销售总量、损耗量和净需求
result_df = pd.DataFrame({
```

```
        '销售总量(千克)': category_sales,
        '损耗量(千克)': category_loss,
        '净需求(千克)': category_net_demand
    })
    result_df.reset_index(inplace=True)
    print(result_df)
```

11. 预测销售结果

```
    import pandas as pd
    import numpy as np
    from statsmodels.tsa.arima.model import ARIMA
    # 加载历史销售数据
    sales_data = pd.read_excel(r'C:\Users\32335\MachineLearning\TC\附件 7.xlsx')
    # 将销售日期列转换为日期时间类型
    sales_data['销售日期'] = pd.to_datetime(sales_data['销售日期'])
    # 按分类编码分组
    grouped = sales_data.groupby('分类编码')
    # 创建一个 DataFrame 来存储预测结果
    forecast_results = pd.DataFrame(columns=['分类编码', '日期', '预测需求'])
    # 定义预测的日期范围（2023 年 7 月 1 日到 7 日）
    forecast_dates = pd.date_range(start='2023-07-01', end='2023-07-07')
    # 循环遍历每个品类
    for category_code, group_data in grouped:
        # 创建该品类的时间序列数据，按日期汇总销售量
        category_time_series = group_data.groupby('销售日期')['销量(千
克)'].sum().resample('D').sum()
        # 拟合 ARIMA 模型，这里选择 ARIMA(1,1,1)模型作为示例
        model = ARIMA(category_time_series, order=(1, 1, 1))
        model_fit = model.fit()
        # 预测未来七天的需求
        forecast_values = model_fit.forecast(steps=7)
        # 创建 DataFrame 存储预测结果
        forecast_df = pd.DataFrame({
            '分类编码': [category_code] * 7,
            '日期': forecast_dates,
            '预测需求': forecast_values
        })
        # 将预测结果添加到总的预测结果 DataFrame 中
        forecast_results = pd.concat([forecast_results, forecast_df])
    # 打印预测结果
    print(forecast_results)
```

12. 计算每个品类的平均成本和市场需求强度

```
import pandas as pd
# 加载附件 8 的数据
data8 = pd.read_excel(r'C:\Users\32335\MachineLearning\TC\附件 8.xlsx')
# 计算每个品类的平均成本
average_cost_per_category = data8.groupby('分类编码')['批发价格(元/千克)'].mean()
# 计算市场需求强度
total_sales = data8['销量(千克)'].sum()
market_demand_intensity_per_category = data8.groupby('分类编码')['销量(千克)'].sum()
/ total_sales
# 创建包含结果的 DataFrame
result_df = pd.DataFrame({
    '分类编码': average_cost_per_category.index,
    '平均成本(元/千克)': average_cost_per_category.values,
    '市场需求强度': market_demand_intensity_per_category.values
})
# 打印每个品类的平均成本和市场需求强度
print("每个品类的平均成本和市场需求强度：")
print(result_df)
```

13. 每个品类的定价策略

```
# 每个品类的成本
cost_per_category = {
    '1011010101': 3.841342,  # 计算平均成本
    '1011010201': 5.961771,
    '1011010402': 6.622156,
    '1011010501': 5.547849,
    '1011010504': 6.744188,
    '1011010801': 7.704686,
}
# 市场需求强度（0 到 1 之间，1 表示最高需求）
demand_intensity_per_category = {
    '1011010101': 0.421510,
    '1011010201': 0.088681,
    '1011010402': 0.086164,
    '1011010501': 0.047628,
    '1011010504': 0.194466,
    '1011010801': 0.161551,
}
# 定价策略：基于成本加成和需求强度
pricing_strategy = {}
for category_code in cost_per_category:
    cost = cost_per_category[category_code]
```

```
        demand_intensity = demand_intensity_per_category[category_code]
        # 假设的加成率，可以根据需求强度进行调整
        markup_rate = 0.2  # 20%的加成率
        # 根据成本和加成率计算价格
        price = cost * (1 + markup_rate)
        # 根据需求强度调整价格
        price *= (1 + demand_intensity)
        pricing_strategy[category_code] = price
    # 打印每个品类的定价策略
    print("每个品类的定价策略：")
    print(pricing_strategy)
```

14. 筛选 2023 年 6 月 24-30 日的可售品种

```
    # 将销售日期列转换为日期类型，筛选出 2023 年 6 月 24-30 日的可售品种
    data['销售日期'] = pd.to_datetime(data['销售日期'])
    # 设定筛选条件，选择销售日期在 2023 年 6 月 24 日到 2023 年 6 月 30 日之间的数据
    start_date = pd.to_datetime("2023-06-24")
    end_date = pd.to_datetime("2023-06-30")
    filtered_data = data[(data['销售日期'] >= start_date) & (data['销售日期'] <=
end_date)]
    # 输出筛选后的数据
    print(filtered_data)
    # 保存筛选后的数据到指定路径
    filtered_data.to_excel("C:/Users/Desktop/可售品种_2023-06-24 至 2023-06-30.xlsx",
index=False)
```

15. 预测每个单品的销售需求

```
    # 建立每个单品的销售需求模型
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from statsmodels.tsa.arima.model import ARIMA
    from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
    from statsmodels.tsa.stattools import adfuller
    # 加载数据
    data = pd.read_excel("C:/Users/Desktop/可售品种_2023-06-24 至 2023-06-30.xlsx")
    # 将销售日期列转换为日期时间类型，并设置为索引
    data['销售日期'] = pd.to_datetime(data['销售日期'])
    data.set_index('销售日期', inplace=True)
    # 提取每个单品的销售数据（假设“单品编码”列为单品的唯一标识）
    单品编码列表 = data['单品编码'].unique()
    # 创建一个空的 DataFrame 来存储所有单品的预测数据
```

```
所有单品预测数据 = pd.DataFrame()
# 循环遍历每个单品编码，建立销售需求模型
for 单品编码 in 单品编码列表:
    单品数据 = data[data['单品编码'] == 单品编码]['销量(千克)']
    # 检查时间序列的平稳性
    result = adfuller(单品数据)
    p_value = result[1]
    if p_value <= 0.05:
        print(f"单品编码 {单品编码} 的时间序列平稳，p-value = {p_value:.4f}")
        # 绘制自相关函数（ACF）和偏自相关函数（PACF）图
        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 6))
        plot_acf(单品数据, ax=ax1, lags=5)
        plot_pacf(单品数据, ax=ax2, lags=5)
        plt.show()
        # 建立 ARIMA 模型
        # 这里仅以 ARIMA(1,1,1)为例，请根据 ACF 和 PACF 图选择合适的阶数
        model = ARIMA(单品数据, order=(1, 1, 1))
        model_fit = model.fit()
        # 预测未来销售需求（可以根据需要设置预测的时间范围）
        预测值 = model_fit.forecast(steps=7)  # 预测未来 7 天
        print(f"单品编码 {单品编码} 的销售需求预测：\n{预测值}")
        # 创建一个包含预测值的 DataFrame
        预测数据 = pd.DataFrame({'预测销售需求': 预测值})
        # 将单品的预测数据附加到总的 DataFrame 中
        所有单品预测数据 = pd.concat([所有单品预测数据, 预测数据], axis=1)
    else:
        print(f"单品编码 {单品编码} 的时间序列不平稳，p-value = {p_value:.4f}")
# 指定保存路径
保存路径 = "C:/Users/32335/MachineLearning/TC/Forecasted_Demand_All.xlsx"
# 将所有单品的预测数据保存到 Excel 文件
所有单品预测数据.to_excel(保存路径, index=False)
print(f"所有单品的预测销售需求数据已保存到 {保存路径}")
```

16. 预测单品商品在未来 7 天的定价

```
# 将销售日期列转换为日期时间类型，建立每个单品商品在未来 7 天的定价模型
data['销售日期'] = pd.to_datetime(data['销售日期'])
# 计算未来 7 天的日期
end_date = data['销售日期'].max() + timedelta(days=7)
date_range = pd.date_range(start=end_date - timedelta(days=6), end=end_date,
freq='D')
# 提取每个单品的销售数据
单品编码列表 = data['单品编码'].unique()
# 创建一个 DataFrame 来存储定价模型的结果
```

```
定价模型结果 = pd.DataFrame(columns=['单品编码', '日期', '定价']
# 循环遍历每个单品编码，建立定价模型并预测未来 7 天的定价
for 单品编码 in 单品编码列表:
    单品数据 = data[data['单品编码'] == 单品编码]
    # 假设定价与批发价格和销售量有关
    X = 单品数据[['批发价格(元/千克)', '销量(千克)']].values  # 使用.values 将数据转换为 NumPy 数组
    y = 单品数据['销售单价(元/千克)']
    # 建立线性回归模型
    model = LinearRegression()
    model.fit(X, y)
    # 将特征名称传递给模型
    model.feature_names_in_ = ['批发价格(元/千克)', '销量(千克)']
    # 预测未来 7 天的定价
    未来 7 天预测 = pd.DataFrame({'单品编码': [单品编码] * 7, '日期': date_range})
    未来 7 天预测['批发价格(元/千克)'] = 单品数据['批发价格(元/千克)'].mean()  # 使用平均批发价格
    未来 7 天预测['销量(千克)'] = 单品数据['销量(千克)'].mean()  # 使用平均销量
    未来 7 天预测['定价']=model.predict(未来 7 天预测[['批发价格(元/千克)','销量(千克)']])
    # 将结果添加到定价模型结果 DataFrame 中
    定价模型结果 = pd.concat([定价模型结果, 未来 7 天预测], ignore_index=True)
# 保存结果到表格
定价模型结果.to_excel("C:/Users/Desktop/定价模型结果.xlsx", index=False)
```

17. 更新定价模型

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from datetime import datetime, timedelta
# 加载数据
data = pd.read_excel("C:/Users/Desktop/可售品种_2023-06-24 至 2023-06-30.xlsx")
定价模型结果 = pd.read_excel("C:/Users/Desktop/定价模型结果.xlsx")  # 加载之前建立的定价模型结果
# 将销售日期列转换为日期时间类型
data['销售日期'] = pd.to_datetime(data['销售日期'])
# 计算未来 7 天的日期
end_date = data['销售日期'].max() + timedelta(days=7)
date_range = pd.date_range(start=end_date - timedelta(days=6), end=end_date, freq='D')
# 提取每个单品的销售数据
单品编码列表 = data['单品编码'].unique()
# 更新模型结果以满足最小陈列量的要求
```

```
更新后的模型结果 = []
# 循环遍历每个单品编码，更新定价模型结果
for 单品编码 in 单品编码列表:
    单品数据 = data[data['单品编码'] == 单品编码]
    # 获取该单品的最小陈列量
    最小陈列量 = 2.5  # 你可以根据实际需求设定
    # 获取该单品的预测定价结果
    单品预测 = 定价模型结果[定价模型结果['单品编码'] == 单品编码]
    # 计算每天的订购量，并更新预测结果
    单品预测['订购量(千克)'] = np.maximum(最小陈列量, 单品预测['定价'] * 单品预测['销量(千克)'])
    # 添加更新后的预测结果到列表中
    更新后的模型结果.append(单品预测)
# 合并更新后的模型结果
更新后的模型结果 = pd.concat(更新后的模型结果)
# 保存结果到表格
更新后的模型结果.to_excel("C:/Users/Desktop/更新后的定价模型结果.xlsx", index=False)
```

18. 蔬菜品类一周预测需求

日期	分类编码	时间	预测需求
2023-07-01	1011010101	2023-07-01 00:00:00	139.954275
2023-07-02	1011010101	2023-07-02 00:00:00	142.601358
2023-07-03	1011010101	2023-07-03 00:00:00	143.339697
2023-07-04	1011010101	2023-07-04 00:00:00	143.545639
2023-07-05	1011010101	2023-07-05 00:00:00	143.603081
2023-07-06	1011010101	2023-07-06 00:00:00	143.619104
2023-07-07	1011010101	2023-07-07 00:00:00	143.623573
2023-07-01	1011010201	2023-07-01 00:00:00	22.976202
2023-07-02	1011010201	2023-07-02 00:00:00	20.685084
2023-07-03	1011010201	2023-07-03 00:00:00	19.657999

2023-07-04	1011010201	2023-07-04 00:00:00	19.197568
2023-07-05	1011010201	2023-07-05 00:00:00	18.991161
2023-07-06	1011010201	2023-07-06 00:00:00	18.898631
2023-07-07	1011010201	2023-07-07 00:00:00	18.857151
2023-07-01	1011010402	2023-07-01 00:00:00	18.39016
2023-07-02	1011010402	2023-07-02 00:00:00	18.005223
2023-07-03	1011010402	2023-07-03 00:00:00	17.861201
2023-07-04	1011010402	2023-07-04 00:00:00	17.807315
2023-07-05	1011010402	2023-07-05 00:00:00	17.787154
2023-07-06	1011010402	2023-07-06 00:00:00	17.779611
2023-07-07	1011010402	2023-07-07 00:00:00	17.776789
2023-07-01	1011010501	2023-07-01 00:00:00	23.43595
2023-07-02	1011010501	2023-07-02 00:00:00	23.056671
2023-07-03	1011010501	2023-07-03 00:00:00	22.925185
2023-07-04	1011010501	2023-07-04 00:00:00	22.879602
2023-07-05	1011010501	2023-07-05 00:00:00	22.8638
2023-07-06	1011010501	2023-07-06 00:00:00	22.858322
2023-07-07	1011010501	2023-07-07 00:00:00	22.856423
2023-07-01	1011010504	2023-07-01 00:00:00	85.195777
2023-07-02	1011010504	2023-07-02 00:00:00	86.400719
2023-07-03	1011010504	2023-07-03 00:00:00	86.899687
2023-07-04	1011010504	2023-07-04 00:00:00	87.10631

2023-07-05	1011010504	2023-07-05 00:00:00	87.191873
2023-07-06	1011010504	2023-07-06 00:00:00	87.227304
2023-07-07	1011010504	2023-07-07 00:00:00	87.241977
2023-07-01	1011010801	2023-07-01 00:00:00	48.771503
2023-07-02	1011010801	2023-07-02 00:00:00	52.16541
2023-07-03	1011010801	2023-07-03 00:00:00	53.4175
2023-07-04	1011010801	2023-07-04 00:00:00	53.879424
2023-07-05	1011010801	2023-07-05 00:00:00	54.049839
2023-07-06	1011010801	2023-07-06 00:00:00	54.112708
2023-07-07	1011010801	2023-07-07 00:00:00	54.135902