

# 基于PSO-BP和MILP模型的蔬菜类商品定价与补货策略优化

## 摘要

据大数据统计,蔬菜类商品的鲜度和品相随时间增加会变差,导致隔日无法再售。由于商超所售蔬菜种类繁多,且产地各异,而进货时间通常为凌晨3:00到5:00,所以商家难以作出最优的补货决策和定价决策。本文通过使用Apriori算法和时间序列预测模型对蔬菜类商品数据进行分析,然后使用PSO-BP神经网络优化模型和ARIMA模型对销售量进行数据拟合预测。在销售空间有限的情况下,构建MILP模型,对7月1日的单品定价进行预测,从而优化了各单品的补货量和定价策略,有利于商超经济的持续稳定发展。

**针对问题一:** 通过运用**Apriori**关联规则算法,分析了蔬菜品类销售量与单品销售量的相互关系。首先,进行数据预处理,删除异常值。对于分布规律,采用频数分布直方图的统计方法,据图分析,发现品类销售量主要集中在花叶类蔬菜上,且各品类蔬菜销售量之间差别较大;同时,基于时间与销售量两大维度,进行了**时间序列分析**,从而发现单品销售量具有冬夏季上升、春秋季下降的周期性交叉变换规律;对于相关关系,运用Apriori关联规则算法,发现辣椒类和食用菌分别与其他品类的销售量存在较强的关联。

**针对问题二:** 构建基于粒子群算法优化BP神经网络(**PSO-BP**)的模型,从而对未来一周各蔬菜品类的销售总量进行预测。对于第一小问,首先将、数据正则化处理,再利用Logistic回归模型,得出各蔬菜品类销售总量和成本加成定价的关系;对于第二小问,首先运用**ARIMA**预测模型,预测未来一周的销售总量,其次使用**PSO-BP**时间序列预测模型进行了优化预测;对于第三小问,综合应用前两小问的结论,利用销售总量和成本加成定价的回归方程,根据未来一周各蔬菜品类销售总量的预测值,合理分析出其定价的预测值,商超据此定价标准并结合市场实际需求量制定定价策略。

**针对问题三:** 运用混合整数线性规划模型(**MILP**),综合约束条件对于各单品的补货量和定价策略进行优化。首先,进行数据预处理,提取2023年6月23到30日的数据,计算各蔬菜单品的利润,根据利润额和供应的季节性提取28个蔬菜单品;其次,建立混合整数线性规划模型,在可售单品数量限制、最小陈列量限制、产品需求多样化约束条件下,考虑收益最大化目标与所选单品的定价、进货量的关系,规划了强化约束的预测定价标准,指导商超进行补货决策和定价决策的优化,从而提高了经济效益。

**针对问题四:** 考虑“客户需求、市场分析数据”,“商品原产地数据”,“环境因素数据”这三大因素,通过线性相关模型和动态分析模型进行数据分析与产品规划,帮助商超更准确地预测市场需求,例如:季节性和趋势数据、供应链数据、客户购买行为数据等。根据上述的信息集合来商超可以定制更有效的定价策略,实现最大化收益,降低成本与沉没成本,并且以多元化的信息考量商品的供应,从而更好地融入市场、满足商超的可持续发展、满足客户的实际需求、优化蔬菜商品的供应链,致力于实现商超收益最大化、浪费最小化。

综上所述,本文依据各题给出的条件全面分析了蔬菜类商品的补货模型和定价模型,并给出了限制陈列量条件下蔬菜类商品最佳的定价策略,实现了商超收益最大化。经过分析验证,本文的模型可以面向金融工程、数据科学领域进行推广,具有合理性和一定的现实意义。

**关键词:** 蔬菜类商品 成本加成定价 PSO-BP模型 MILP模型 Python

# 一、问题重述

## 1.1 问题背景

生鲜超市的蔬菜类商品保鲜期短，且品相随时间增加会变差，若当日未售出，则隔日无法再售。所以商超补货主要依据商品的历史销售和需求状况。

由于所售蔬菜种类繁多、产地各异，而蔬菜的进货时间通常在凌晨3:00到4:00，因此，商家须在具体单品和进货价格不明确的情况下，作出当日各蔬菜品类的补货决策。蔬菜定价采用“成本加成定价法”，运损和品相变差的商品打折出售。正确的市场需求分析有利于作出合理的补货决策和定价决策。针对需求侧，蔬菜类商品的销售量与时间具有相关关系；针对供给侧，蔬菜的供应品种在4月到10月较为丰富，而商超销售空间的有限性使得合理的销售布局尤为重要。

## 1.2 问题提出

附件1给出了某商超销售的6种蔬菜品类的商品信息；附件2和附件3分别给出了该商超2020年7月1日至2023年6月30日各商品的销售流水明细和批发价格的相关数据；附件4给出了各商品近期的损耗率数据。

依据附件中的相关数据和实际情况进行分析建模，解决以下问题：

- 问题一：蔬菜类商品不同品类或不同单品之间可能存在一定的关联关系，请分析蔬菜各品类及单品销售量的分布规律及相互关系。
- 问题二：考虑商超以品类为单位做补货计划，请分析各蔬菜品类的销售总量与成本加成定价的关系，并给出各蔬菜品类未来一周(2023年7月1-7日)的日补货总量和定价策略，使得商超收益最大。
- 问题三：因蔬菜类商品的销售空间有限，商超希望进一步制定单品的补货计划，要求可售单品总数控制在27-33个，且各单品订购量满足最小陈列量2.5千克的要求。根据2023年6月24-30日的可售品种，给出7月1日的单品补货量和定价策略，在尽量满足市场对各品类蔬菜商品需求的前提下，使得商超收益最大。
- 问题四：为了更好地作出蔬菜商品的补货量及定价策略，商超还需搜集哪些相关数据，这些数据如何解决商超的收益最大化问题，请给出你们的意见和理由。

## 二、问题分析

### 2.1研究综述

生鲜超市对蔬菜类商品的销售状况及进货价格给出了6种品类共计251种单品近4年的销售流水明细和批发价格的相关数据，题目要求据此制定合理的补货决策和定价决策。然而由于生鲜产品本身具有易腐性的特点，其新鲜度和营养价值会随着时间的流逝而不断下降，合理有效的定价方式也已经成为了众多生鲜零售商在生鲜产品销售中需要考虑的重点问题<sup>[7]</sup>。因此，针对不同的蔬菜类单品，为达到到商超收益最大化，有必要挑选出利润贡献率高的单品，并考虑销售空间的有限性，作出最优的补货决策和定价决策，使得商超的收益达到最大。

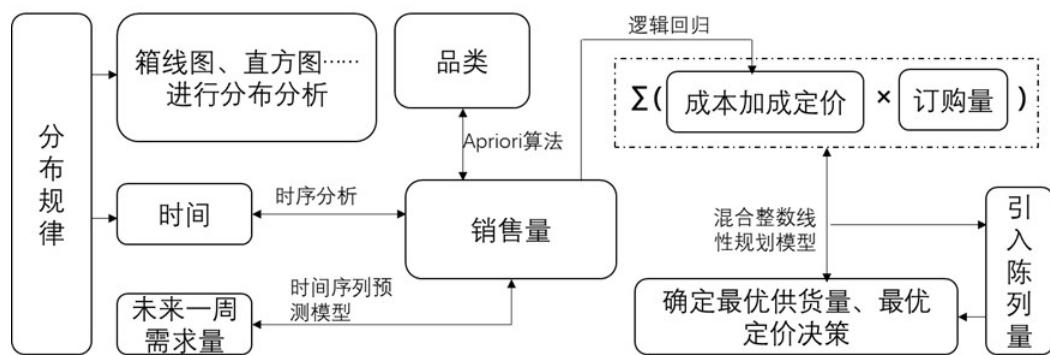


图 1: 总体框架图

由图1可以看出，本文就蔬菜类商品的定价问题，建立了涉及时间、需求量、销售量蔬菜品类及单品、成本加成定价等指标的Apriori算法预测模型、时间序列预测模型、逻辑回归模型、混合整数线性规划模型。为商超指定最优的补货决策和定价策略提供指导，帮助其实现收益最大化。

### 2.2问题剖析

#### 2.2.1分析蔬菜各品类及单品销售量的分布规律及相互关系

问题一主要探究蔬菜类商品不同品类及单品销售量的分布规律及相关关系。首先，对数据进行预处理，处理缺失值和异常值，删除销售量小于零（即退货）的单品交易列，同时提取相关特征，按照销售时间将附件1、2、3、4各单品编码对应合并。其次，对于销售量的分布规律分析，由于其与时间的密切关联，采用时序分析考察各品类及单品销售量随时间的变化趋势，并运用直方图分析销售量的分布情况。最后，针对关联性分析，利用Apriori算法找

出频繁购买的品类和单品组合。

### 2.2.2建立周预测模型和最优定价模型

问题二要求分析各蔬菜品类的销售总量与成本加成定价的关系，并给出各蔬菜品类未来一周日补货总量和定价策略，使得商超收益最大。首先，对数据进行正则化处理，消除数据的不平衡性，考虑到环境对销售量影响的强度随时间增加而增加，且数据量较大，采用Logistic回归建立销售总量和成本加成定价的关系。其次，为给出未来一周的日补货量，先运用ARIMA模型进行预测，再使用粒子群算法结合BP神经网络（PSO-BP）模型优化预测精度，为作出合理的捕获总量决策做指导。最后，题目要求给出定价策略，使得商超收益最大，可以根据本题前半部分得出的销售总量关于成本加成定价的逻辑回归模型和未来一周的销售总量，反推出各蔬菜品类的定价方案。

### 2.2.3销售空间有限性条件下商超收益最大化

问题三要求在将可售单品总数控制在 27-33 个，且各单品订购量满足最小陈列2.5千克的条件下，制定7月1号的日补货量和定价策略，使得商超收益最大。首先，进行数据预处理，选择2023年6月24到30日的数据，计算出各单品的利润，筛选出利润排名前40的蔬菜单品，同时考虑到季节性，选择7月份的蔬菜单品，最终确定28个蔬菜单品。其次，题目要求给出单品补货量和定价策略，本题强化了约束条件，限制可售单品数量、限制最小陈列量、满足市场多样化单品需求，建立混合整数线性规划模型，考虑收益最大化目标与所筛选单品的定价、进货数量的关系，以此制定出最优的补货决策和定价策略，实现商超收益的最大化。

### 2.2.4搜集其他相关数据优化补货、定价模型

问题四要求考虑其他相关数据，进一步优化问题二、问题三制定的销售量预测模型和定价模型，并给出意见和理由。首先，通过查阅文献可以找出其他影响收益的特征因素。其次，将搜集的数据整合清洗有。再次，在对数据作探索性研究，分析一般规律后，构建合适的数学模型。最后，对莫i选哪个结果进行分析，得出最优的补货、定价策略，指导商超进一步扩大收益。

## 三、模型假设

为了更好的建立模型我们做出如下假设：

1. 销售数据稳定性假设:假设销售数据在短期内是相对稳定的，且不受突发事件的影响(例如：地震、龙卷风等天气因素，疫情、流感等病理因素)。
2. 市场竞争稳定性假设:假设市场上的其他对手行为相对稳定，他们的价格策略和促销活动不会剧烈波动，不会造成突然的竞争。
3. 价格影响假设:由市场普遍规律可以建立以下假设：价格上涨时可能导致销售额度下降，价格下降时可能导致销售额度增加。

## 四、符号说明

符号	意义	单位
$t$	时间周期	次序
$i$	蔬菜单品排名	次序
$X_i$	第 <i>i</i> 个单品的定价	元
$m_i$	第 <i>i</i> 个单品订购量	千克
$Y_i$	第 <i>i</i> 个单品的销量	千克
$Y_{t-i}$	第 <i>i</i> 个单品的历史销售量	千克
$Y_i^e$	第 <i>i</i> 个单品的预测销售量	千克
$P_s$	销售价格	元
$P_c$	批发价格	元
$l$	损耗率	%
$E$	总收益	元

注：表中未出现的符号在文中均有详细说明。

## 五、模型的建立与求解

### 5.1问题一模型的建立与求解

#### 5.1.1模型的分析

问题一主要探究蔬菜类商品不同品类及单品销售量的分布规律及相关关系。在对数据进行预处理，删除销售量小于零（即退货）的单品交易列后，提取相关特征，按照销售时间

将附件1、2、3、4各单品对应合并。然后，对于销售量的分布规律分析，由于销售量属于连续性变量，且与时间的密切关联，考虑采用时序分析考察各品类及单品销售量随时间的变化趋势，并采用直方图分析销售量的分布情况。最后，各品类销售量的关联性分析，利用Apriori算法找出频繁购买的品类和单品组合。

## 5.1.2模型的原理

### (1) 时序分析模型

#### (A) 基本原理

时间序列也称动态序列，是指将某种现象的指标数值按照时间顺序排列而成的数值数列，包括两个要素：时间要素和数值要素。一般的，时间序列具有3种效应：趋势性、周期性（季节性）、随机性，除此之外，还有节日效应。时间序列3种效应的组合方式有：

Tt代表趋势效应，St代表季节效应，It代表随机效应

加法模型： $X_t = Tt + St + It$

乘法模型： $X_t = Tt \times St \times It$

为了确定没有随机趋势或确定趋势，避免“伪回归”问题，需要先做平稳性检验，达到销售量不随时间改变，只依赖于K这个时间跨度，不依赖于时间点t本身。

即：

$$E(Z_t) = \text{constant} \quad (1)$$

$$\text{Var}(Z_t) = \text{constant}$$

$$\text{Cov}(Z_t, Z_{t-1}) = \text{constant}$$

#### (B)DF检验

$x_t = x_{t-1} + u_t$ 是非平稳的，其中 $u_t$ 是白噪声（即零均值，常方差的稳定随机序列）。该序列可以看成 $x_t = \rho * x_{t-1} + u_t$ 的参数 $\rho = 1$ 的序列，此时 $x_t$ 存有一个单位根，即该时间序列是随机游走序列，为非平稳的。可对其进行差分处理：

$$\Delta x_t = (\rho - 1) * x_{t-1} + u_t, \delta = \rho - 1 \quad (2)$$

（原假设  $H_0$ : 序存在单位根，即参数  $\delta = 0$ ）

检验时间序列的平稳性，一般可通过检验带有截距项的一阶自回归模型：

$$\Delta x_t = \alpha + \delta x_{t-1} + \mu_t \quad (3)$$

对上式可通过进行普通最小二乘法的t检验完成（即若P值 $< 0.05$ ，则拒绝原假设，证明 $\delta \neq 0$ ，序列平稳。）

### (2) Apriori算法

关联规则分析(Association Rule Analysis)是为了发掘数据背后的关联关系而诞生的。对于Apriori算法，我们使用支持度来作为我们判断频繁项集的标准。Apriori算法的目标是找到最大的K项频繁集。

支持度为：(以两数据为例)

$$Support(XY) = P(XY) = \frac{num(XY)}{num(AllSamples)}$$

置信度为：(以两数据为例)

$$Confidence(X \Leftarrow Y) = P(X|Y) = P(XY)/P(Y)$$

提升度为：(以两数据为例)

$$Lift(X \Leftarrow Y) = P(X|Y)/P(X) = Confidence(X \Leftarrow Y)/P(X)$$

Apriori算法采用了迭代的方法，先搜索出候选1项集及对应的支持度，剪枝去掉低于支持度的1项集，得到频繁1项集。然后对剩下的频繁1项集进行连接，得到候选的频繁2项集，筛选去掉低于支持度的候选频繁2项集，得到真正的频繁二项集，以此类推，迭代下去，直到无法找到频繁k+1项集为止，对应的频繁k项集的集合即为算法的输出结果。

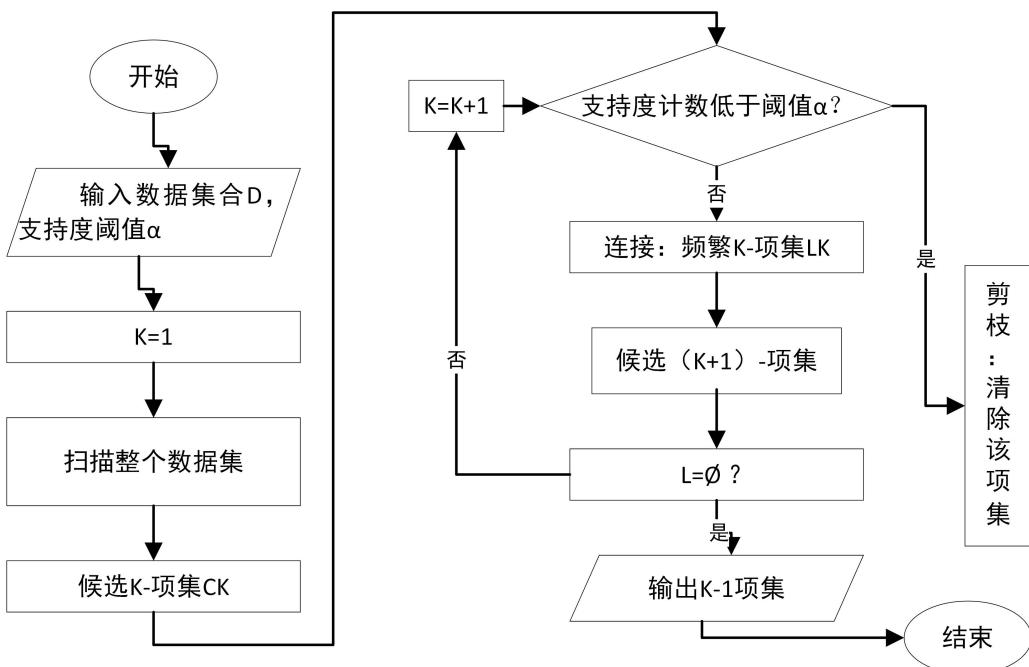


图 2: Apriori算法流程图

### 5.1.3模型的建立与求解

#### (1) 分布规律

### (A) 频数分布直方图

通过Python将附件1、2、3、4整合在一张表格中，并统计出6种蔬菜品类整个时间跨度的销售总量，画出如图3所示的频数分布直方图。

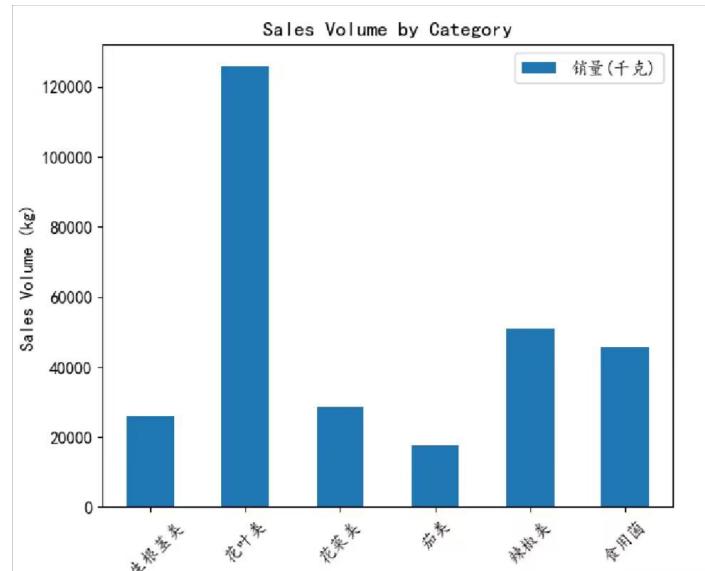


图 3: 蔬菜各品类销售量频数分布直方图

从图3蔬菜各品类销售量频数分布直方图可以看出,花叶类蔬菜的销售总量呈现明显的高峰, 茄类蔬菜销售总量最少。销售总量以: 花叶类、辣椒类、食用菌类、花菜类、水生根茎类、茄类的顺序逐渐降低。由此可以得出: 商超应该增加花叶类和辣椒类蔬菜的订购量, 减少茄类蔬菜的订购量, 在有限的订购类别中实现销售量的最大化, 从而获得最大收益。

### (B) 时间序列图

将处理后的数据导入Python, 分析销售量随时间的变化趋势。分别以日和月为时间单位(月销售量为当月销售量总和), 作出了销售量关于时间的时间序列图。

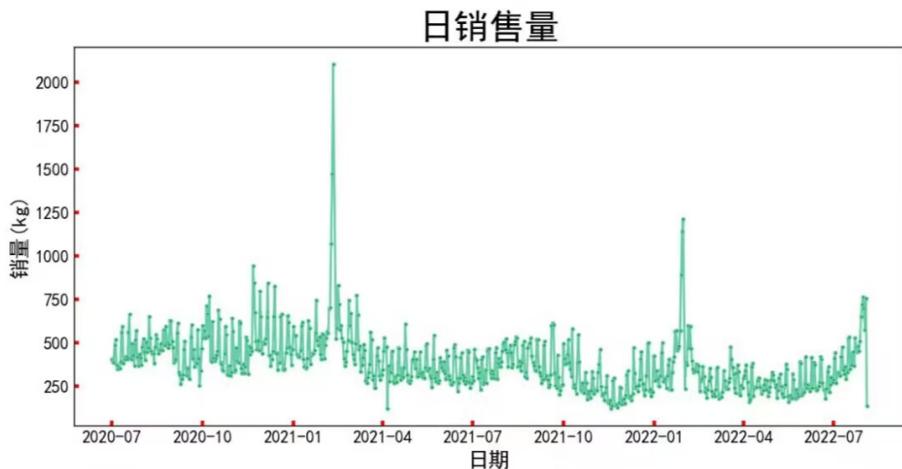


图 4: 日销售量时间序列图

由图4日销售量时间序列图可以看出，从2020年7月份到2021年初，销售量有所增加，在2021年的2月份左右急速增加；随后到2021年末缓慢下降，达到最低值；2022年初销售量又出现一个小的峰值，随后开始缓慢下降。（由于数据量太多，这里只选取了2020年7月至2022年7月的销售量）

由此可以得到，销售量呈现年周期性，即每年年初销售量会出现一个峰值，而后缓慢回落，直至下一年年初再次上升。这可能跟年初居民消费意愿增强，消费能力上升有关。商超可以在年初增加订购量，以满足市场丰富的消费需求。

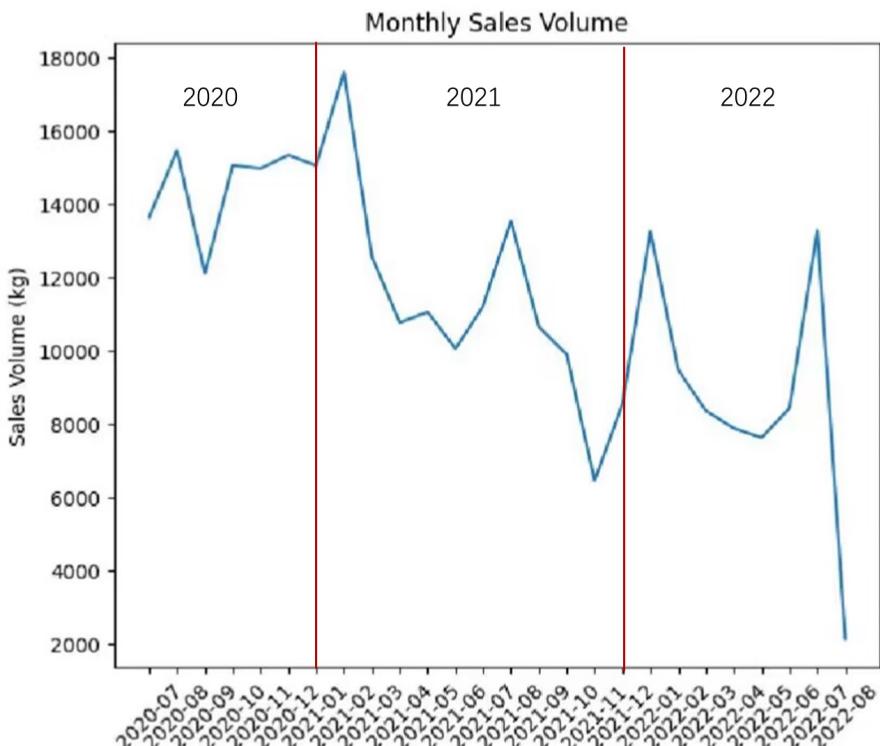


图 5: 月销售量时间序列图

由图5月销售量时间序列图可以看出，从2020年到2022年销售量绝对值呈现波动下降趋势。其中2020年波动趋势相对较小，从2021年初开始急剧下降，直至2022年月销售量跌到最低值，这可能是受新冠疫情爆发的影响，商超的蔬菜类商品供给较少，居民消费意愿减弱。对比2020、2021和2022年可以看出，每年的7、8月份销售量会有明显的上升趋势，因为此时蔬菜的供应品中较为丰富，选择的多样性也增加了蔬菜类商品的销售量。

## (2) Apriori算法分析关联性

对不同蔬菜单品的销售量进行分析，得到各品类销量的关联规则，如下表1所示。（表1只列出了按交易时间前后的前13组单品关联组）

表 1: 蔬菜各单品销售量关联表

蔬菜单品	关联蔬菜单品	置信度
东门口小白菜	奶白菜	0.017
大龙茄子	泡泡椒 (精品)	0.232
金针菇 (1)	娃娃菜	0.399
小米椒	红杭椒	0.088
小米椒	青线椒	0.528
平菇	牛首油菜	0.258
茼蒿	平菇	0.513
金针菇 (1)	平菇	0.031
枝江红菜苔	牛首油菜	0.022
茼蒿	牛首油菜	0.301
红尖椒	螺丝椒	0.065
青线椒	红杭椒	0.800
红椒 (1)	青尖椒	0.661

由表1蔬菜各单品销售量关联表可以看到, (大龙茄子) -泡泡椒 (精品)、金针菇 (1) - 娃娃菜、小米椒-红杭椒、平菇-牛首油菜、茼蒿-平菇、泡泡椒(精品) 和西峡香菇 (1) - 净藕 (1)……(由于篇幅有限, 各蔬菜单品关联规则在此不再详细列出, 全部关联规则及置信度见附录5) 按照蔬菜品类划分, 做出了各品类间的关联关系, 见下图6。

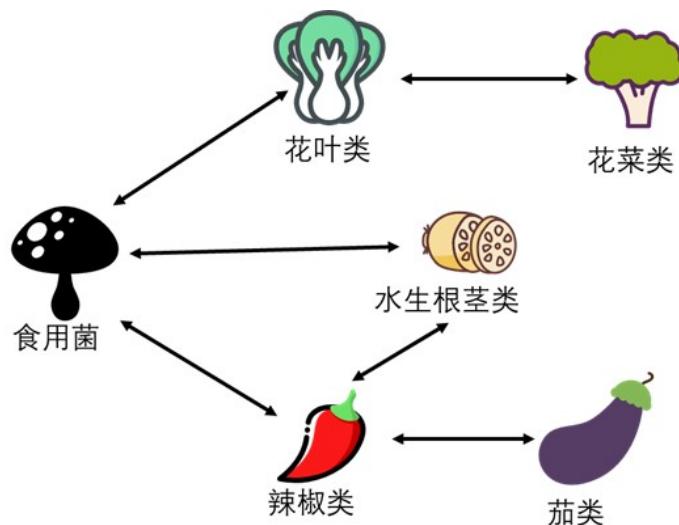


图 6: 蔬菜各品类销售量关联规则示意图

具有关联的蔬菜品类其销售量存在相互促进效应, 观察图6蔬菜各品类销售量关联规则

示意图可以发现辣椒类蔬菜和食用菌在关联关系中出现的最频繁：食用菌的销售量与花叶类、辣椒类、水生根茎类、花菜类蔬菜销售量均有关联，而辣椒类蔬菜与茄类、水声根茎类、花菜类、食用菌蔬菜销售量均有关联，这说明食用菌和辣椒类数次啊几乎可以与其他任何一种蔬菜搭配，商超可以将这两类蔬菜商品摆放在距离其他各品类蔬菜都比较近的地方，以增加消费者的购买欲望。同时，食用菌、辣椒类蔬菜、水生根茎类蔬菜以及花叶类蔬菜、食用菌、花菜类蔬菜三者的销售量也有一定的关联关系，商超可以考虑整合这样的三类蔬菜商品地摆放位置，便于消费者购买，同时适当地对具有关联的蔬菜品类合并折扣出售，以提高经济效益。

## 5.2问题二模型的建立与求解

### 5.2.1模型的分析

问题二要求分析各蔬菜品类的销售总量与成本加成定价的关系，并给出各蔬菜品类未来一周日补货总量和定价策略，使得商超收益最大。首先，将数据正则化处理，防止过度拟合，然后采用Logistic回归建立销售总量和成本加成定价的关系。其次，为给出未来一周的日补货量，需要预测未来一周的销售量，先运用ARIMA模型进行预测，再使用粒子群算法结合BP神经网络（PSO-BP）模型优化预测精度。最后，题目要求给出未来一周的的补货量和定价策略。在本题第一小问已经拟合出销售总量和成本加成定价的关系，本题第二小问预测了未来一周的销售量，据此可以得出未来一周蔬菜商品的成本加成定价，该定价是综合销量等因素而制定，可以扩大商超的利润空间。

### 5.2.2模型的原理

#### (1) 正则化

正则化就是对原函数添加一个惩罚函数（损失函数），在对数据集中的数据点赋值时，通过原函数的相关参数加一定的惩罚函数，来使得函数逐渐逼近最优解。并且在不同环境下可以改变惩罚参数，使得训练出来的模型分布更加稀疏，不集中在某一区域。

惩罚参数记作 $\Omega(\theta)$ ，正则化之后的损失函数记作 $\tilde{J}$ ，则

$$\tilde{J}(\theta, X, y) = J(\theta, X, y) + \alpha\Omega(\theta) \quad (4)$$

其中 $\alpha \in [0, +\infty]$ 是惩罚函数的权重，越大的 $\alpha$ 对应越多的正则化程度。

#### (2) Logistic回归模型

Logistic回归模型可以用一下微分方程去描述（其中 $N_m$ 表示理论上的最大值）

$$\frac{dN}{dt} = r(1 - \frac{N}{N_m}), N(t_0) = N_0 \quad (5)$$

将微分方程分离变量并代入初值为:

$$N = \frac{1}{\frac{1}{N_m} + (\frac{1}{N_0} - \frac{1}{N_m})e^{-r(t-t_0)}} \quad (6)$$

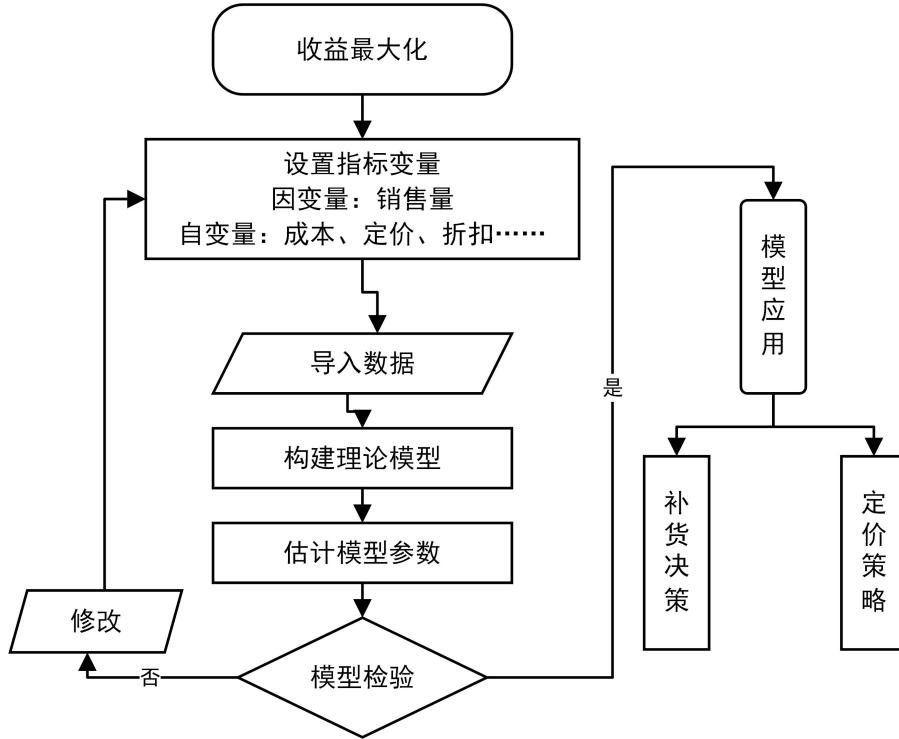


图 7: Logistic 回归分析流程图

### (3) ARIMA 时间序列预测模型

ARIMA 模型包含 3 个部分，即自回归(AR)、差分(I)和移动\*均(MA)，它们的含义分别是：

- AR 表示自回归(Auto Regression)
- I 表示单整阶数(Integration)，时间序列必须是\*稳的，才能建立计量模型。对时间序列进行单位根检验，如果是非\*稳序列，那么需要通过差分转化为\*稳序列，经过几次差分转化为\*稳序列，就称为几阶单整
- MA 表示移动\*均模型(Moving Average)

ARIMA 模型记作 ARIMA(p, d, q)，p 为自回归项数；q 为滑动\*均项数，d 为使之成为\*稳序列所做的差分次数(阶数)。“差分”是关键步骤，采用 ARIMA 模型预测的时序数据，必须是稳定的(\*稳定性)，不稳定的数据，是无法捕捉到时序规律的。

ARIMA原理：将非平稳时间序列转化为平稳时间序列然后将因变量仅对它的滞后值以及随机误差项的现值和滞后值进行回归所建立的模型。

p阶自回归过程的公式定义为：

$$y_t = \mu + \sum_{i=1}^p \gamma_i y_{t-i} + \varepsilon_t \quad (7)$$

其中 $y_t$ 是当前值， $\mu$ 是常数项， $p$ 是阶数， $\gamma_i$ 是自相关系数， $\varepsilon_t$ 是误差， $y_{t-i}$ 是历史值。

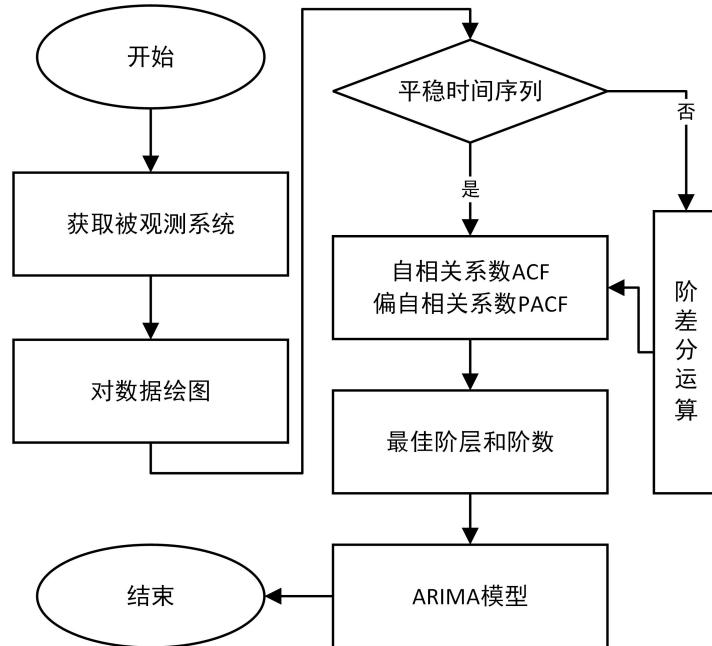


图 8: 时间序列建模流程图

时间序列预测建模一般步骤为：首先导入被观测数据，其次使得所有序列平稳化，再次求出自相关系数和偏自相关系数，得到最佳阶层，最后对所得ARIMA模型进行检验。

#### (4) PSO-BP时间序列预测模型

基于粒子群优化算法（PSO）优化BP神经网络时间序列预测模型可以将BP神经网络的权重和阈值作为粒子的位置表示，以均方误差或其他适当的评价指标作为适应度函数。通过不断地更新粒子的位置和速度，使得BP神经网络的预测误差逐渐减小，得到更好的预测结果。

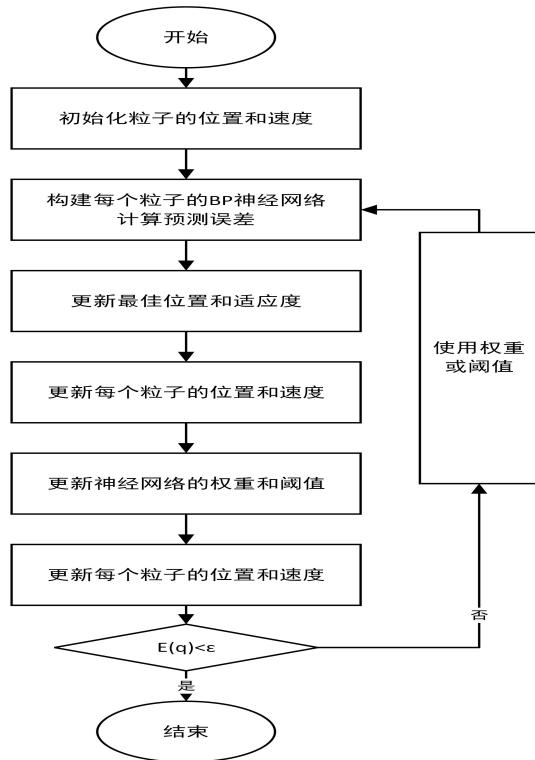


图 9: PSO-BP时间序列预测模型流程图

一般来说，模型的主要步骤如下：

初始化粒子群中每个粒子的位置和速度。

对于每个粒子，使用当前位置构建BP神经网络，并计算预测误差。

更新每个粒子的最佳位置和最佳适应度。

更新每个粒子的速度和位置。

对于每个粒子，更新BP神经网络的权重和阈值。

重复步骤2-5，直到满足终止条件（例如达到最大迭代次数或预测误差小于某个阈值）。

### 5.2.3模型的建立与求解

#### (1) Logistic回归模型分析关系

首先根据成本加成定价中“定价=（1+成本利润率）\*单位成本”计算各蔬菜品类的成本加成定价，同时计算出各蔬菜品类销售总量；其次为防止分析销售总量和成本加成定价关系时过度拟合，应用Python对销售总量数据和各品类成本定价数据进行正则化处理，运行代码见附录，得出表2。

表 2: 正则化处理-销售总量和定价表

变量	数值					
成本加成定价X	0.84189586	0.53964003	0.82622734	0.56333682	0.73461938	0.67847946
销售总量Y	0.97857705	0.20588094	0.87519365	0.48377275	0.76919981	0.63900834

对正则化处理过的稳定的数据进行Logistic回归分析，用 $X_i$ 表示各蔬菜品类成本加成定价， $Y_i$ 表示各蔬菜品类销售总量。其中 $i = 1, 2, \dots, 6$ 。（分别代表花菜类、花叶类、水生根茎类、茄类、辣椒类蔬菜和食用菌）

使用Matlab回归得出模型的未知参数，可以写出回归方程为

$$Y = \frac{438.8462}{1 + 1.0521e^{-9.6835X}} \quad (8)$$

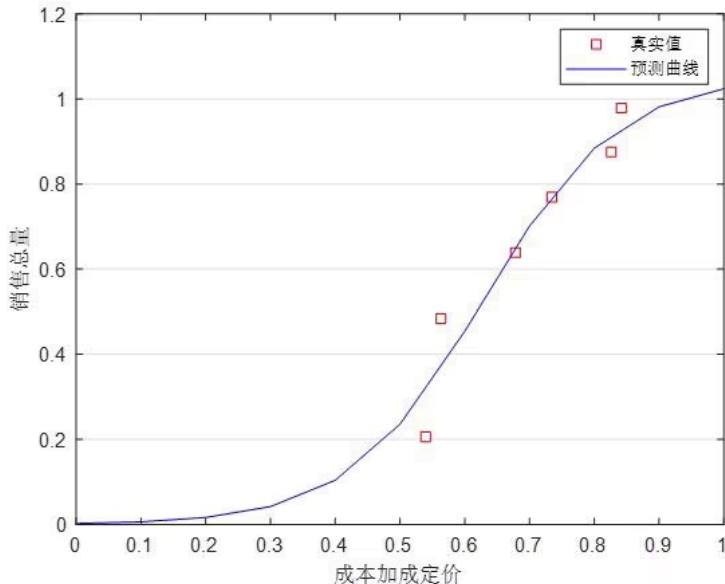


图 10: 销售总量-成本加成定价的回归拟合图

同时画出了回归拟合曲线，由图10销售总量-定价的回归拟合图可以看出，销售曲线的真实值和预测曲线基本一致，说明拟合的回归曲线可以反映销售总量和成本加成定价的关系。前期成本加成定价对销售总量的影响较大，但随着环境因素影响力不断增强，成本加成定价对销售总量的影响不断减弱，销售总量在不断增加的过程中存在一个上限，即销售总量不可能无限增加。这要求商超作出合理的定价决策以增加销售总量，同时考虑其他因素对销售总量的影响，谋求更大利润。

## (2) ARIMA建立周预测模型

### (A)绘制各蔬菜品类历史销售量的时间序列图

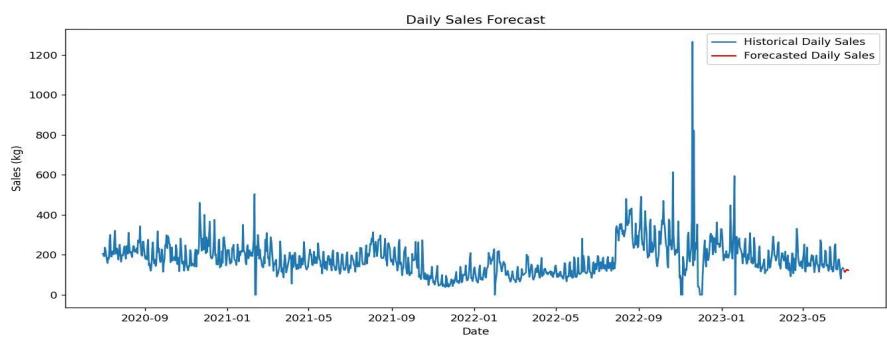


图 11: 花叶类蔬菜时间序列图

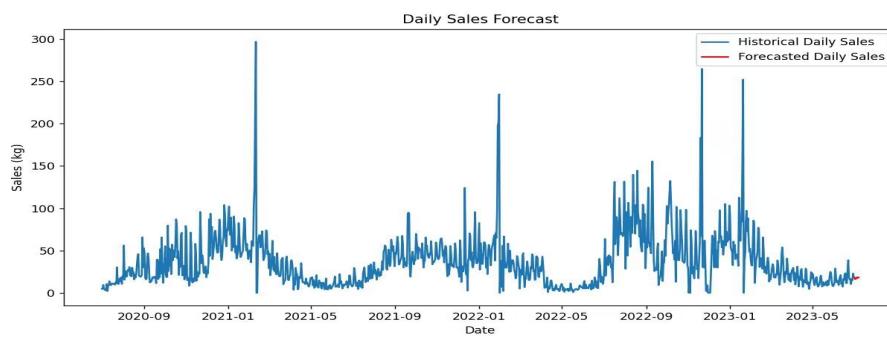


图 12: 花菜类蔬菜时间序列图

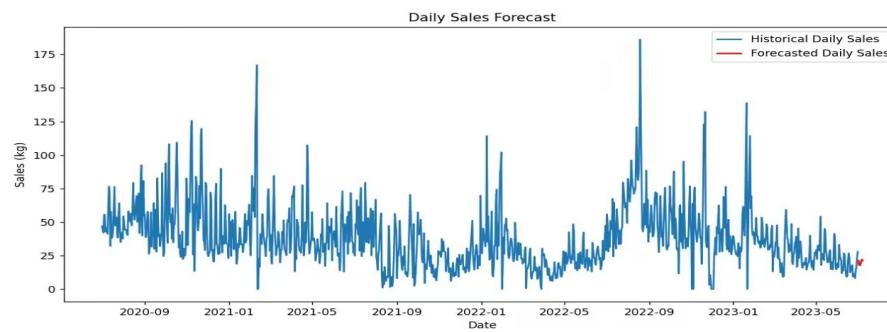


图 13: 水生根茎类蔬菜时间序列图

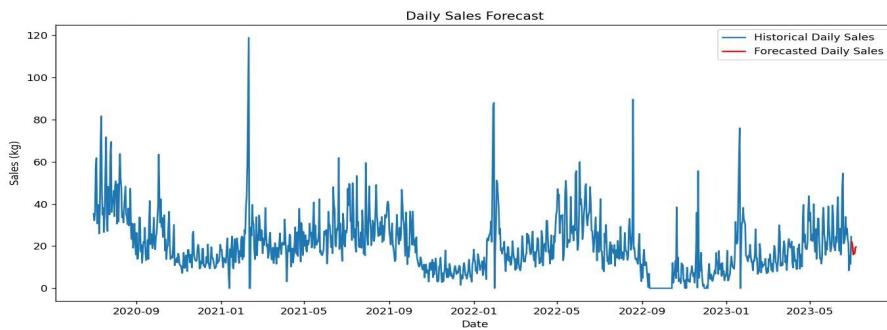


图 14: 茄类蔬菜时间序列图

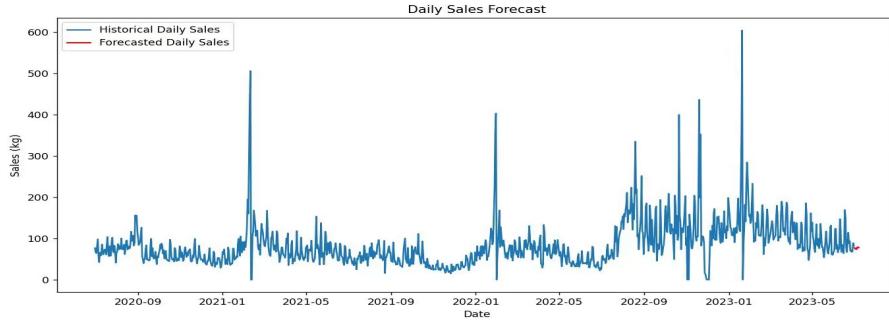


图 15: 辣椒类蔬菜时间序列图

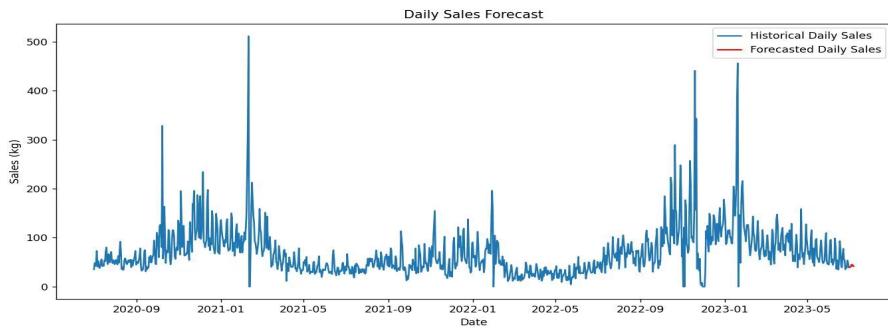


图 16: 食用菌蔬菜时间序列图

由各蔬菜品类历史销售总量时间序列分布图可以看出，各年份销售总量围绕一个值上下波动。且有明显的季节波动性，销售总量在冬季显著上升，进入春季逐渐回落，夏季开始又波动上升，进入秋季上升趋势逐渐减弱。

综上所述，该商超蔬菜类商品在夏冬两季为销售旺季，在春秋两季为销售淡季。

#### (B)建立预测模型

确定模型阶数后，预测销售总量 $Y_t$ 在未来一周的值，针对时间序列预测模型的一般形式

$$y_t = \mu + \sum_{i=1}^p \gamma_i y_{t-i} + \varepsilon_t$$

运用Python计算出模型里的常数项 $\mu$ ，历史销售量 $y_{t-i}$ ，自相关系数 $\gamma_i$ ，误差 $\varepsilon_t$ ，进而预测出蔬菜各品类未来一周的销售量。运行代码见附录，预测结果如表3所示。

表 3: 各蔬菜品类未来一周销售量预测

	花叶类	花菜类	水生根茎类	茄类	辣椒类	食用菌
1日	119.788238	21.345175	18.360545	22.018674	77.360113	39.417055
2日	113.881616	17.908934	16.366054	19.706889	74.840902	40.338509
3日	120.367779	17.725709	16.872867	15.982792	73.921305	40.790728
4日	122.824793	18.602839	17.125373	16.701725	74.112690	44.914085
5日	126.231984	20.808670	18.601610	16.460376	78.723609	44.979535
6日	125.775287	22.096681	18.434664	19.267003	79.047416	42.716349
7日	122.963259	21.179842	18.173555	19.595051	77.910977	41.771704

### (3)PSO-BP时间序列预测模型

使用Matlab建立PSO-BP模型，分析各蔬菜品类历史销售量，通过PSO优化神经网络权值阈值，改进了神经网络的预测精度，得出如图17到图21的结果图。(由于篇幅有限，在此只展示花菜类蔬菜的POS-BP预测结果图，具体预测值及其他5种蔬菜品类的预测结果图见附录4)

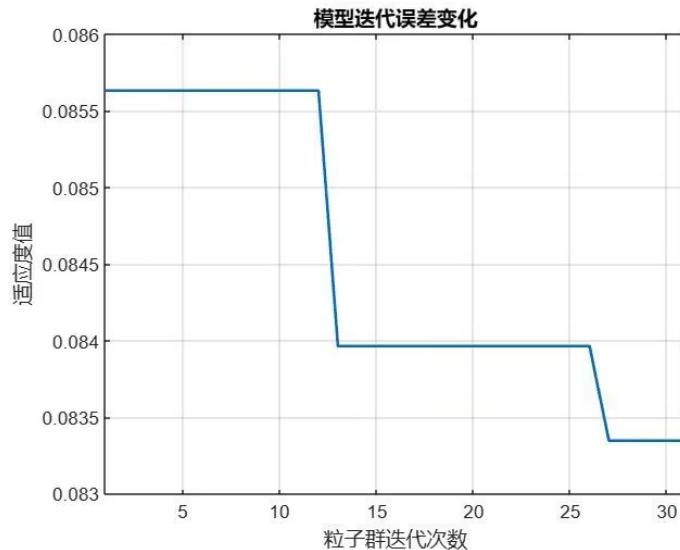


图 17: 模型迭代误差变化图

从图17模型迭代误差变化图可以看出，随着迭代次数不断增加，模型的误差在不断下降。

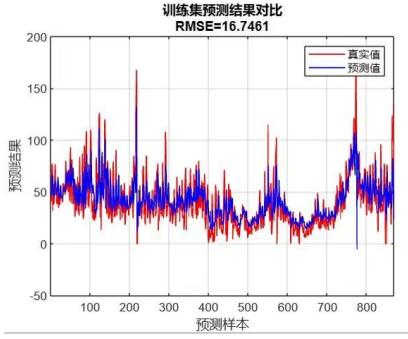


图 18: 训练集预测结果对比图

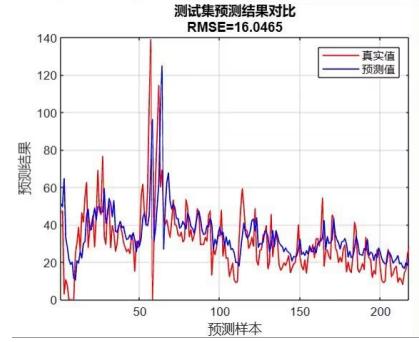


图 19: 测试集预测结果对比图

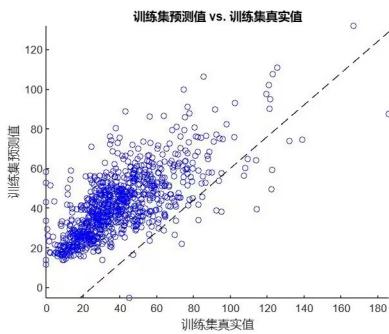


图 20: 训练集预测值-训练集真实值

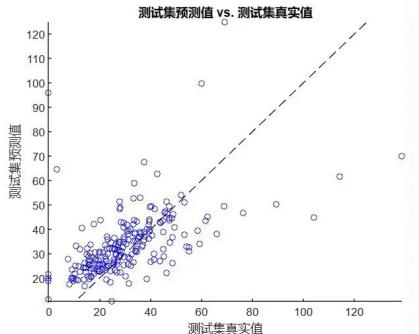


图 21: 测试集预测值-测试集真实值

观察图18、图19、图20、图21，对照训练集和测试集的预测结果和实际结果，可以发现粒子群算法改进的BP神经网络预测效果精确度很好。此预测模型可作为 ARIMA时间序列预测模型的补充，用于提高未来一周销售量预测值的准确度，为商超制定补货计划并作出合理的定价决策提供可靠的指导。

#### (4) 定价策略

使用Python将表3的数据放入拟合的销售总量关于成本加成定价的模型（方程8）中，反推出未来一周各蔬菜单品的成本加成定价，运行代码见附录，运行结果如表4展示。

表 4: 各蔬菜品类定价策略

	花叶类	花菜类	水生根茎类	茄类	辣椒类	食用菌
1日	0.1918	0.6036	0.6362	0.5969	0.3079	0.4678
2日	0.2061	0.6416	0.6609	0.6209	0.3162	0.4626
3日	0.1905	0.6438	0.6544	0.6660	0.3193	0.4600
4日	0.1847	0.6334	0.6512	0.6566	0.3186	0.4380
5日	0.1768	0.6092	0.6334	0.6597	0.3036	0.4377
6日	0.1779	0.5961	0.6354	0.6258	0.3025	0.4495
7日	0.1844	0.6053	0.6384	0.6222	0.3062	0.4546

由表4各蔬菜品类定价策略可以发现，花菜类蔬菜和水生根茎类蔬菜的定价标准应当着重考虑，结合表3，发现这两种品类蔬菜未来一周的销售量在所有品类中处于最低规模。花叶类蔬菜的成本加成定价相对较低，这与其未来一周高销量预测由密切关系。结合问题一的关联规则分级，还可以注意到辣椒类蔬菜这种与其他品类蔬菜关联性较强的商品，该品类的定价标准不高，但在所有品类蔬菜中，其未来一周预测销售量排在第二位，因为辣椒类蔬菜可以作为配菜，搭配在大部分蔬菜品类中，商超可以关注这类关联品类蔬菜，提高消费者联合购买概率。

在定价策略上，商超可以适当地开展一些打折促销活动，例如，对销售量较低的蔬菜品类可以给出一定折扣，或者将该蔬菜品类与其关联商品一起给出折扣，以此吸引消费者购买意愿，提升该蔬菜品类的销售量。

## 5.3问题三模型的建立与求解

### 5.3.1模型的分析

问题三考虑到商超销售空间的有限性，加入陈列量、可售品种数量两个约束条件，并满足市场对各品类蔬菜商品需求，制定7月1号的日补货量和定价策略，使得商超收益最大。首先，进行数据预处理，提取2023年6月24到30日的销售量数据，计算此时间段各蔬菜单品的利润，筛选出利润贡献率排名前40的蔬菜单品，同时考虑到季节性，必须选择7月份的蔬菜单品，最终确定33个蔬菜单品。其次，建立混合整数线性规划模型，在可售单品数量限制、最小陈列量限制、产品需求多样等约束条件下，考虑收益最大化目标与所选单品的定价、进货量的关系，指导商超进行补货决策和定价决策，提高经济效益。

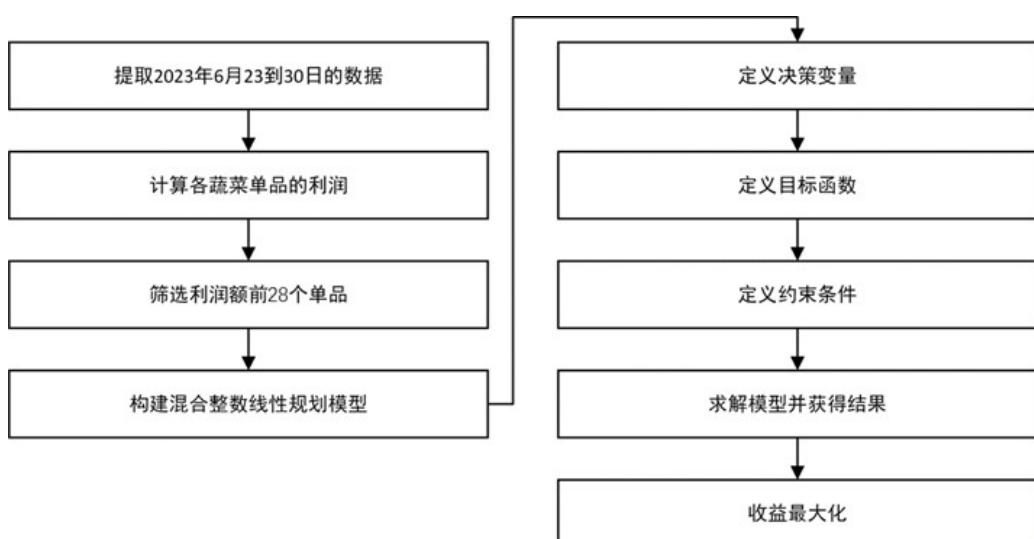


图 22: 问题三分析框架图

### 5.3.2模型的原理

混合整数线性规划模型也叫做分支限界法，是指目标函数是线性的，约束条件也是线性的，部分决策变量要求必须是整数的线性规划问题，也就是说此优化问题不止有条件约束，还有整数约束。

混合整数线性规划的基本思路是：

- 将原混合整数线性规划问题改进为行的松弛问题，不断地用单纯形法求解
- 通过增加约束来进行分支求解
- 直到整数最优解出现在新的改进后的松弛问题的一个顶点

通过割平面法增加最优解的边界能力，割平面法的核心思想是通过增加约束改进松弛最优解，但不会消除可行解。从松弛解中选择取值分数的变量，经过如下转换可以得到一个切割：

$$\sum (a_{1j} - [a_{1j}])x_j \geq b_1 - [b_1] \quad (9)$$

从而得到新约束。重新求解增加新约束的松弛解。

割平面法的思路为：

- 求解线性松弛问题
- 选择取值为分数的变量，增加Gomory切割
- 使用对偶单纯形法获得新的最优解
- 迭代直至解满足整数要求或者不再有可行解

混合整数线性规划模型的一般形式为：（其中， $x_j$ 为整数）

$$\begin{aligned} \max(\min) Z = & \sum_{j=1}^n c_j x_j \\ s.t. \left\{ \begin{array}{ll} \sum_{j=1}^n a_{ij} x_j \leq (= \geq) b_i, & (i = 1, 2, \dots, m) \\ x_j \geq 0, & (j = 1, 2, \dots, n) \end{array} \right. \end{aligned} \quad (10)$$

### 5.3.3模型的建立与求解

#### (1) 蔬菜单品的筛选

首先，使用Python计算2023年6月24日到30日各单品的利润额，题目要求销售单品数量处于27-33之间，先按照利润额筛选出排名前40的蔬菜单品，考虑到有些蔬菜单品供应具有季节性，如小青菜、红薯尖等，需要去除这部分蔬菜单品。

最终确定28个蔬菜单品，分别为：洪湖藕带、赤松茸、丝瓜尖、鸡枞菌、小米椒、黑皮鸡枞菌、螺丝椒(份)、黄花菜、七彩椒(2)、外地茼蒿、小米椒(份)、水果辣椒(橙色)、七彩椒(1)、蔡甸藜蒿、高瓜(1)、螺丝椒、菱角、菠菜、竹叶菜、红椒(1)、小白菜、苋菜、银耳(朵)、西兰花、平菇、红灯笼椒(2)、净藕(1)、鲜藕带(袋)，共计28个蔬菜单品。

## (2) 商超收益最大化——混合整数线性规划模型

### (A) 定义模型相关指标

a. 目标函数：最大化收益

b. 决策变量：单品的订购量、单品的定价

c. 约束条件：单品最小订购量为2.5千克、可售单品总数在27-34之间、订购量不超过预测的需求量

将上述各个指标转化为数学符号表示为：

a. 目标函数：

$$E = \sum_{i=1}^n [Y_i^e \times (P_s - P_c) \times (1 - l)] \Rightarrow \max \quad (11)$$

b. 决策变量： $m_i$ 、 $X_i$

c. 约束条件：

$$m_i \geq 2.5$$

$$i \in [27, 34] \quad (12)$$

$$\sum_{i=1}^n m_i \leq Y_i^e$$

### (B) 求解模型并分析结果

引入问题二的蔬菜单品定价模型，对提取的28个蔬菜单品预测价格，但本题给出了更为严格的约束条件，要求在单品最小订购量为2.5千克、可售单品总数在27-34之间、订购量不超过预测的需求量三个约束条件下，做出定价决策。导入所筛选单品已处理过的数据，使用Python建立定价标准、订购量的混合整数线性规划模型，并预测出7月1日的单品定价。

表 5: 7月1日各单品定价预测表

单品名称	定价(元)	补货量(千克)
洪湖藕带	81.845	2.5
赤松茸	66.77	2.5
鸡枞菌	131.34	2.5
小米椒	184.113	2.5
黑皮鸡枞菌	111.885	2.5
螺丝椒(份)	26.116	2.5
黄花菜	110.703	2.5
七彩椒(2)	45.209	2.5
外地茼蒿	32.664	2.5
小米椒	20.08	2.5
水果辣椒(橙色)	32.772	2.5
七彩椒(1)	42.587	2.5
蔡甸藜蒿	34.255	2.5
高瓜(1)	37.982	2.5
螺丝椒	36.574	2.5
菱角	29.731	2.5
菠菜	33.155	2.5
竹叶菜	26.656	2.5
红椒(1)	43.488	2.5
小白菜	27.485	2.5
苋菜	23.864	2.5
银耳(朵)	17.074	2.5
西兰花	24.916	2.5
平菇	31.128	2.5
红灯笼椒(2)	34.758	2.5
净藕(1)	23.846	2.5
鲜藕带(袋)	20.457	2.5

## 5.4问题四模型的建立与求解

第一点：客户需求、市场分析数据

(1) 通过发布线上或线下调研，并以优惠卷为奖励机制来收集顾客的购买爱好，经济水平、家庭人口等信息，分析顾客的消费水平能力来决定商品定价。

(2) 收集周边其他商家的价格数据，用于定价策略的优化。据上一条分析出本商超受欢迎的产品，并适当打折扣，以吸引更多顾客人流量。

(3) 不同的时间和日期可能会有不同的客流量，这与蔬菜的销售量有直接关联。利用线性相关模型和动态分析模型，做出每日时间与人流量的折线图，有利于了解高峰和低谷时段，可以帮助商家调整陈列策略、促销活动时间以及进货量。

### 第二点：商品原产地数据

(1) 收集各品类蔬菜的供应地信息，其中包括：蔬菜的产地和产季信息、地方仓库库存量信息，可以用于补货策略的优化；地方蔬菜价格更优惠、交通运输开销比较等信息，致力于降低成本与沉没成本，以多元化的信息考量商品的供应。

(2) 由于各地季节温度可能有较大差异，对脆弱的蔬菜进行合理的保鲜技术投资，从而减少沉没成本、保证蔬菜质量；也有利于面向顾客树立信任形象，使企业获得更加持续性更大的利润回报。

### 第三点：环境因素数据

(1) 从环境方面，需要增加各地区季节性因素，每日天气和节假日数据，分析其对销售量的影响。已知不同的季节和气候差异，会对促销活动等其他活动不同的影响，例如在阴雨天、暴风雨天不适合进行促销折扣活动。

(2) 从政策方面，收集政府的政策信息，如商品补贴、商品优惠政策、价格限制额度等。

(3) 从储存方面，需要增加库存状况数据，一方面可以用于补货策略的优化，例如某地盛产某蔬菜，那么在该蔬菜资源紧缺时，优先联系此仓库保持商品及时供应；另一方面也有助于当某些蔬菜资源紧缺时，可以适量需要调高价格，也属于政策优化的一种。

## 六、模型的评价与改进与推广

### 6.1 模型的优点

(1) 问题一的模型利用了时间序列预测模型和Apriori关联规则挖掘方法，前者通过时序来分析得出各商品类和商品单品的销售量随时间的变化趋势；后者则通过较高的相关系数，找出频繁购买的品类和单品组合，实现销售联动上涨。

(2) 问题二首先在分析成本加成定价的关系上采用了多元线性回归模型（MRM），“多元”即“销售量、成本、定价”，由各类商品的历史数据结合“销售量、成本、定价”三元，

从而确定优化定价策略；然后，采用时间序列分析方法的ARIMA模型来预测未来一周内的销售量，易建立线性规划模型（LP）求解出符合各约束条件的目标函数最优解，接着取得补货量和定价策略的最优解，保证商超利益值的最大化。

（3）问题三同样采用问题二的模型，在问题二的基础上结合混合整数线性规划模型(MILP)，其精准算法能够求得模型的精确最优解定价策略。

（4）问题四在问题一、二、三的基础上，参考更多的环境因素、客观因素，例如：蔬菜供应链的产地、不同季节的应季蔬菜信息、不同地区土壤的微量元素决定有机蔬菜与无机蔬菜等，通过构建线性规划模型（LP）来确定商超最优的补货时间、补货量和定价策略。

（5）本文采用了大量的数据计算，并建立大量的基本数学模型，可以确保数据的有效性与精确性，从而更精准的预测未来商超的收益趋势，并做出相应的策略。

## 6.2模型的缺点

（1）混合整数线性规划模型(MILP)缺点：虽然能够处理决策变量较多的问题，但其得到的最优解为近似最优解，且比较容易陷入局部最优解，所求得的近似最优解与实际最优解之间的差距无法衡量和估计。

（2）线性规划模型（LP）缺点：该方法在建立方程时非常简单快速，但不利于人工计算，在想预估出数据处理结果时较难以实现。

（3）建模过程中，对约束条件和普遍性规律的研究和计算不够细致。

（4）没有考虑其他自然环境因素的对蔬菜所造成的影响（如：土壤湿度、气温气候、土壤含微量元素不同等），使得不同地区的数据结论可能具有较大误差，具有单一性、片面性。

## 6.3模型的推广

本文所用算法和模型可广泛使用于各种商业产品的销售领域，其先将问题拆分成若干子问题，再寻求最优解，对研究求解过程中较复杂多变问题具有重要意义。同时此算法简单易懂，实用性很强，灵活性很高对于优化类情境具有现实性的意义，适合推广到金融工程，数据科学等相关研究工作中。

## 七、参考文献

- [1]司守奎,孙玺菁.数学建模算法与应用(第3版)[M].国防工业出版社,2021.
- [2]Michael G ,Marina Á M ,Agustín B . Compositional Data Analysis of Microbiome and Any-Omics Datasets: A Validation of the Additive Logratio Transformation [J]. Frontiers in Microbiology,2021,12.
- [3]Frank R. Giordano, William P.Fox, Steven. 数学建模[M].机械工业出版社,2019.
- [4]Mark M.Meerschaert. 数学建模方法与分析[M].机械工业出版社, 2015.
- [5]卓金武. MATLAB在数学建模中的应用[M].北京航空航天大学出版社, 2011.
- [6]韩中庚. 数学建模方法及其应用(第3版)[M].高等教育出版社,2017.
- [7]黎莎,修睿,计明军.基于新鲜度动态变化的冷链物流库存分配与运输路径联合优化[J].系统工程,2021,39(05):69-80.
- [8]魏杰,姜云超.考虑供应商保鲜努力的生鲜电商销售模式选择研究[J].控制与决策,2023,1-9.
- [9]Wang B ,Moayedi H ,Nguyen H , et al. Feasibility of a novel predictive technique based on artificial neural network optimized with particle swarm optimization estimating pullout bearing capacity of helical piles[J]. Engineering with Computers,2019,36(4).
- [10]贾润达,李志奇,张树磊等.基于混合整数线性规划的浓密脱水过程协调优化[J].控制与决策,2023,1-7.

# 附录

## 附录列表

- 附录1：问题一的源代码
- 附录2：问题二的源代码
- 附录3：问题三的源代码
- 附录4：PSO-BP模型的相关预测图
- 附录5：关联规则结果图

## 附录1：问题一的源代码

---

```
1 问题一代码
2
3 #第一题第一小问1.
4 import pandas as pd
5 import os
6
7 # 定义文件路径
8 file_path = r"E:\py_works"
9
10 # 通过os.path.拼接路径并读取数据join
11 attachment_1 = pd.read_excel(os.path.join(file_path, "附件1.xlsx"))
12 attachment_2 = pd.read_excel(os.path.join(file_path, "附件2.xlsx"))
13 attachment_3 = pd.read_excel(os.path.join(file_path, "附件3.xlsx"))
14 attachment_4 = pd.read_excel(os.path.join(file_path, "附件4.xlsx"))
15
16 # 检查数据类型
17 print("附件1 单品编码数据类型:", attachment_1[‘单品编码’].dtypes)
18 print("附件2 单品编码数据类型:", attachment_2[‘单品编码’].dtypes)
19 print("附件3 单品编码数据类型:", attachment_3[‘单品编码’].dtypes)
20 print("附件4 小分类编码数据类型:", attachment_4[‘小分类编码’].dtypes)
21
22 # 如果数据类型不同, 将它们转换为相同的数据类型
23 attachment_1[‘单品编码’] = attachment_1[‘单品编码’].astype(str)
24 attachment_2[‘单品编码’] = attachment_2[‘单品编码’].astype(str)
25 attachment_3[‘单品编码’] = attachment_3[‘单品编码’].astype(str)
```

```

26 attachment_4[‘小分类编码’] = attachment_4[‘小分类编码’].astype(str)
27 attachment_1[‘分类编码’] = attachment_1[‘分类编码’].astype(str) # 新添加的代码行
28
29 # 合并数据
30 merged_data = pd.merge(attachment_2, attachment_1, on=‘单品编码’, how=‘left’)
31
32 merged_data = pd.merge(merged_data, attachment_3, left_on=[‘单品编码’, ‘销售日期’], right_on=[‘单品编码’, ‘日期’], how=‘left’)
33
34 merged_data = pd.merge(merged_data, attachment_4, left_on=‘分类编码’, right_on=‘小分类编码’, how=‘left’)
35
36 merged_data.to_csv(os.path.join(file_path, “merged_data_new.csv”), index=False)
37
38 print(merged_data.head())
39
40 #第一题第小问2.2
41
42 import pandas as pd
43 import matplotlib.pyplot as plt
44 import os
45
46
47 file_path = r“E:\\pythonProject”
48 merged_data = pd.read_csv(os.path.join(file_path, “merged_data_new.csv”), encoding=‘gbk’)
49 # 读取连接中的数据①
50 merged_data[‘销售日期’] = pd.to_datetime(merged_data[‘销售日期’])
51 # 将转换销售日期列为日期格式②
52
53 # 日销售量分析③
54 daily_sales = merged_data.groupby(‘销售日期’)[‘销量千克()’].sum().reset_index()
55 plt.figure()
56 plt.plot(daily_sales[‘销售日期’], daily_sales[‘销量千克()’])
57 plt.title(‘Daily Sales Volume’)
58 plt.xlabel(‘Date’)
59 plt.ylabel(‘Sales Volume (kg)’)

```

```

60 plt.show()#展示出第一个图:: 日销售量分析
61
62 # 月销售量分析@@@
63 merged_data['YearMonth'] = merged_data['销售日期'].dt.to_period('M')
64 monthly_sales = merged_data.groupby('YearMonth')[‘销量千克’].sum().reset_index()
65 plt.figure()
66 plt.plot(monthly_sales['YearMonth'].astype(str), monthly_sales[‘销量千克’])
67 plt.title('Monthly Sales Volume')
68 plt.xlabel('Month')
69 plt.ylabel('Sales Volume (kg)')
70 plt.xticks(rotation=45)
71 plt.show()#展示出第一个图: 月销售量分析
72
73 # 商品类别销售量分析@@
74 category_sales = merged_data.groupby('分类名称')[‘销量千克’].sum().reset_index()
75 plt.figure()
76 category_sales.plot(x='分类名称', y='销量千克', kind='bar')
77 plt.title('Sales Volume by Category')
78 plt.xlabel('Category')
79 plt.ylabel('Sales Volume (kg)')
80 plt.xticks(rotation=45)
81 plt.show()
82
83 # 单品销售量分析@@@
84 product_sales = merged_data.groupby('单品名称')[‘销量千克’].sum().reset_index()
85 plt.figure()
86 product_sales.plot(x='单品名称', y='销量千克', kind='bar')
87 plt.title('Sales Volume by Product')
88 plt.xlabel('Product')
89 plt.ylabel('Sales Volume (kg)')
90 plt.xticks(rotation=90)
91 plt.show()
92
93
94 #第一题第小问3.3
95

```

```

96 import pandas as pd
97 from mlxtend.frequent_patterns import apriori, association_rules
98
99 # 步骤 1: 数据准备
100 file_path = "C:\\\\Users\\\\31711\\\\Desktop高教社
    杯\\\\2023\\\\CUMCM2023Problems\\\\题C\\\\merged_data_new.csv"
101 data = pd.read_csv(file_path, encoding='gbk')
102
103 # 步骤 2: 数据转换
104 # 将数据转换为每个交易的购物篮列表
105 transactions = data.groupby(['销售日期', '扫码销售时间'])['单品名
    称'].apply(list).values.tolist()
106
107 # 使用转换数据TransactionEncoder
108 from mlxtend.preprocessing import TransactionEncoder
109 te = TransactionEncoder()
110 te_ary = te.fit(transactions).transform(transactions)
111 df = pd.DataFrame(te_ary, columns=te.columns_)
112
113 # 步骤 3: 关联规则分析
114 # 生成频繁项集
115 frequent_itemsets = apriori(df, min_support=0.0001, use_colnames=True)
116
117 # 输出频繁项集
118 print(frequent_itemsets)
119
120 # 如果有频繁项集, 再生成关联规则
121 if not frequent_itemsets.empty:
122     rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
123 # 输出结果
124 print(rules)
125 else:
126     print("No found")# 表示没有生成关联规则

```

---

## 附录2：问题二的源代码

---

```
1 问题二代码
2
3 #问题二第小问1.1
4
5 import pandas as pd
6 from sklearn.linear_model import LinearRegression
7 from sklearn.model_selection import train_test_split
8
9 # 设置一个函数用来修复无效时间@@@
10 def huifuwuxiaoshijian(t_str):
11     if t_str.startswith("24"):
12         return "00" + t_str[2:]
13     return t_str
14 # 返回的值t_str
15
16 DATE = pd.read_csv('E:\\pythonProject\\merged_data_new.csv', encoding='gbk')
17 #读取文件路径下的内容@@@
18
19 DATE['销售日期'] = pd.to_datetime(DATE['销售日期'])
20 DATE['是否打折销售'] = DATE['是否打折销售'].replace({'是': 1, '否': 0})
21
22 #是打折销售时，返回值，不是打折销售时，返回值10
23 DATE['扫码销售时间'] = DATE['扫码销售时间'].astype(str).apply(huifuwuxiaoshijian)
24
25 #提取扫码销售时间
26 DATE['扫码销售时间'] = pd.to_datetime(DATE['扫码销售时间'],
27                                         errors='coerce', format='%Y-%m-%d %H:%M:%S')
28
29 ## 多元线性回归模型
30 variables = ['销量千克()', '批发价格元千克(/)', '销售单价元千克(/)', '是否打折销售',
31               '平均损耗率(%)小分类编码不同值__']
32 DATE_model = DATE[variables]
33 X = DATE_model.drop('销量千克()', axis=1)
34 y = DATE_model['销量千克()']
35 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
36                                                 random_state=42)
```

```

34
35
36 model = LinearRegression()# 创建模型,
37 model.fit(X_train, y_train)# 训练分析模型结果
38 print('Model_coefficients:', model.coef_)
39
40
41 #问题二第小问2.2
42
43 import pandas as pd
44 from statsmodels.tsa.arima.model import ARIMA
45 import matplotlib.pyplot as plt
46 import os
47
48 # 首先, 文件路径加载数据加载数据@@@
49 file_path = r"E:\pythonProject"
50 data = pd.read_csv(os.path.join(file_path, "merged_data_new.csv"),
51 encoding='gbk')
52 data['日期'] = pd.to_datetime(data['日期'])
53 data.set_index('日期', inplace=True)
54
55 # 分析蔬菜品, 在这里, 我们取出类名为@@@食用菌的分类,,
56 shucaishuju = data[data['分类名称'] == '食用菌']
57
58 # 根据日期聚合销售量数据@@@
59 everyday_sale_num = shucaishuju.resample('D')[‘销量千克()’].sum()
60
61 # 建立模型@@@ARIMA
62 model = ARIMA(everyday_sale_num, order=(5,1,0))
63 model_fit = model.fit()
64 steps = 7# 意为步数, 可以表示未来一周的销售额度step
65 forecast = model_fit.forecast(steps=steps)
66
67 # 用的库将上述结果可视化输出pythonplt
68 plt.plot(everyday_sale_num.index, everyday_sale_num.values, label='Historical
Daily Sales')

```

```

68 plt.plot(pd.date_range(start=everyday_sale_num.index[-1], periods=steps+1)[1:],
69         forecast, label='Forecasted Daily Sales', color='red')
70 #横坐标@@x
71 plt.xlabel('Date')
72 #纵坐标@@y
73 plt.ylabel('Sales (kg)')
74 #带上单位@@kg
75 plt.title('Daily Sales Forecast')
76 plt.legend()
77 plt.show()

78 # 输出预测值
79 print('Forecasted Sales for the Next Week:', forecast)问题二补充代码（优化部分使
80 用）

81
82 matplotlib
83 clear
84 clc
85 x=[0.84189586 0.53964003 0.82622734 0.56333682 0.73461938 0.67847946];
86 y=[0.97857705 0.20588094 0.87519365 0.48377275 0.76919981 0.63900834];

87
88
89 % syms b c
90 % [b,c]=solve(5000/(1+b*exp(-c*1))==43,5000/(1+b*exp(-c*2))==45,b,c)
91 c0=[1 ];
92 % c0=[6000 145.0271 0.0458];c0=[10000 242.3770 0.0457];
93 fun=inline('1./(1+c(1).*exp(-c(2).*x))','c','x');
94 b=nlinfit(x,y,fun,c0);disp("各参数的值(c(1) c(2) c(3)):");disp(b);
95 t=0:0.1:1;
96 plot(x,y,'rs',t,fun(b,t),'b');set(gca,'ygrid','on');
97 legend("真实值","预测曲线");xlabel("成本加成定价");ylabel("销售总量");
98
99 %% 清空环境变量
100 warning off % 关闭报警信息

```

```

102 close all          % 关闭开启的图窗
103 clear             % 清空变量
104 clc               % 清空命令行
105
106 %% 导入数据（时间序列的单列数据）
107 result = xlsread('数据集.xlsx');
108
109 %% 数据分析
110 num_samples = length(result); % 样本个数
111 kim = 15;                  % 延时步长（个历史数据作为自变量）kim
112 zim = 1;                   % 跨个时间点进行预测zim
113
114 %% 构造数据集
115 for i = 1: num_samples - kim - zim + 1
116 res(i, :) = [reshape(result(i: i + kim - 1), 1, kim), result(i + kim + zim -
117 1)];
118 end
119
120 %% 划分训练集和测试集
121 temp = 1: 1: 922;
122
123 P_train = res(temp(1: 700), 1: 15)';
124 T_train = res(temp(1: 700), 16)';
125 M = size(P_train, 2);
126
127 P_test = res(temp(701: end), 1: 15)';
128 T_test = res(temp(701: end), 16)';
129 N = size(P_test, 2);
130
131 %% 数据归一化
132 [p_train, ps_input] = mapminmax(P_train, 0, 1);
133 p_test = mapminmax('apply', P_test, ps_input);
134
135 [t_train, ps_output] = mapminmax(T_train, 0, 1);
136 t_test = mapminmax('apply', T_test, ps_output);

```

```

137 %% 节点个数
138 inputnum = size(p_train, 1); % 输入层节点数
139 hiddennum = 5; % 隐藏层节点数
140 outputnum = size(t_train, 1); % 输出层节点数
141
142 %% 建立网络
143 net = newff(p_train, t_train, hiddennum);
144
145 %% 设置训练参数
146 net.trainParam.epochs = 1000; % 训练次数
147 net.trainParam.goal = 1e-6; % 目标误差
148 net.trainParam.lr = 0.01; % 学习率
149 net.trainParam.showWindow = 0; % 关闭窗口
150
151 %% 参数初始化
152 c1 = 4.494; % 学习因子
153 c2 = 4.494; % 学习因子
154 maxgen = 30; % 种群更新次数
155 sizepop = 5; % 种群规模
156 Vmax = 1.0; % 最大速度
157 Vmin = -1.0; % 最小速度
158 popmax = 2.0; % 最大边界
159 popmin = -2.0; % 最小边界
160
161 %% 节点总数
162 numsum = inputnum * hiddennum + hiddennum * outputnum + outputnum;
163
164 for i = 1 : sizepop
165 pop(i, :) = rands(1, numsum); % 初始化种群
166 V(i, :) = rands(1, numsum); % 初始化速度
167 fitness(i) = fun(pop(i, :), hiddennum, net, p_train, t_train);
168 end
169
170 %% 个体极值和群体极值
171 [fitnesszbest, bestindex] = min(fitness);
172 zbest = pop(bestindex, :); % 全局最佳

```

```

173 gbest = pop; % 个体最佳
174 fitnessgbest = fitness; % 个体最佳适应度值
175 BestFit = fitnesszbest; % 全局最佳适应度值
176
177 %% 迭代寻优
178 for i = 1 : maxgen
179 for j = 1 : sizepop
180
181 % 速度更新
182 V(j, :) = V(j, :) + c1 * rand * (gbest(j, :) - pop(j, :)) + c2 * rand * (zbest
183 - pop(j, :));
184 V(j, (V(j, :) > Vmax)) = Vmax;
185 V(j, (V(j, :) < Vmin)) = Vmin;
186
187 % 种群更新
188 pop(j, :) = pop(j, :) + 0.2 * V(j, :);
189 pop(j, (pop(j, :) > popmax)) = popmax;
190 pop(j, (pop(j, :) < popmin)) = popmin;
191
192 % 自适应变异
193 pos = unidrnd(numsum);
194 if rand > 0.95
195 pop(j, pos) = rands(1, 1);
196 end
197
198 % 适应度值
199 fitness(j) = fun(pop(j, :), hiddennum, net, p_train, t_train);
200
201 end
202 for j = 1 : sizepop
203
204 % 个体最优更新
205 if fitness(j) < fitnessgbest(j)
206 gbest(j, :) = pop(j, :);
207 fitnessgbest(j) = fitness(j);

```

```

208 end

209

210 % 群体最优更新

211 if fitness(j) < fitnesszbest
212 zbest = pop(j, :);
213 fitnesszbest = fitness(j);
214 end

215

216 end

217

218 BestFit = [BestFit, fitnesszbest];
219 end

220

221 %% 提取最优初始权值和阈值

222 w1 = zbest(1 : inputnum * hiddennum);
223 B1 = zbest(inputnum * hiddennum + 1 : inputnum * hiddennum + hiddennum);
224 w2 = zbest(inputnum * hiddennum + hiddennum + 1 : inputnum * hiddennum ...
225 + hiddennum + hiddennum * outputnum);
226 B2 = zbest(inputnum * hiddennum + hiddennum + hiddennum * outputnum + 1 : ...
227 inputnum * hiddennum + hiddennum + hiddennum * outputnum + outputnum);

228

229 %% 最优值赋值

230 net.Iw{1, 1} = reshape(w1, hiddennum, inputnum);
231 net.Lw{2, 1} = reshape(w2, outputnum, hiddennum);
232 net.b{1}      = reshape(B1, hiddennum, 1);
233 net.b{2}      = B2';

234

235 %% 打开训练窗口

236 net.trainParam.showWindow = 1;      % 打开窗口

237

238 %% 网络训练

239 net = train(net, p_train, t_train);

240

241 %% 仿真预测

242 t_sim1 = sim(net, p_train);
243 t_sim2 = sim(net, p_test );

```

```

244
245 %% 数据反归一化
246 T_sim1 = mapminmax('reverse', t_sim1, ps_output);
247 T_sim2 = mapminmax('reverse', t_sim2, ps_output);
248
249 %% 均方根误差
250 error1 = sqrt(sum((T_sim1 - T_train).^2) ./ M);
251 error2 = sqrt(sum((T_sim2 - T_test ).^2) ./ N);
252
253 %% 绘图
254 figure
255 plot(1: M, T_train, 'r-', 1: M, T_sim1, 'b-', 'LineWidth', 1)
256 legend('真实值', '预测值')
257 xlabel('预测样本')
258 ylabel('预测结果')
259 string = {'训练集预测结果对比'; ['RMSE=' num2str(error1)]};
260 title(string)
261 xlim([1, M])
262 grid
263
264 figure
265 plot(1: N, T_test, 'r-', 1: N, T_sim2, 'b-', 'LineWidth', 1)
266 legend('真实值', '预测值')
267 xlabel('预测样本')
268 ylabel('预测结果')
269 string = {'测试集预测结果对比'; ['RMSE=' num2str(error2)]};
270 title(string)
271 xlim([1, N])
272 grid
273
274 %% 误差曲线迭代图
275 figure
276 plot(1: length(BestFit), BestFit, 'LineWidth', 1.5);
277 xlabel('粒子群迭代次数');
278 ylabel('适应度值');
279 xlim([1, length(BestFit)])

```

```

280 string = {'模型迭代误差变化'};
281 title(string)
282 grid on
283
284 %% 相关指标计算
285 % R2
286 R1 = 1 - norm(T_train - T_sim1)^2 / norm(T_train - mean(T_train))^2;
287 R2 = 1 - norm(T_test - T_sim2)^2 / norm(T_test - mean(T_test ))^2;
288
289 disp(['训练集数据的为: R2', num2str(R1)])
290 disp(['测试集数据的为: R2', num2str(R2)])
291
292 % MAE
293 mae1 = sum(abs(T_sim1 - T_train)) ./ M ;
294 mae2 = sum(abs(T_sim2 - T_test )) ./ N ;
295
296 disp(['训练集数据的为: MAE', num2str(mae1)])
297 disp(['测试集数据的为: MAE', num2str(mae2)])
298
299 % MBE
300 mbe1 = sum(T_sim1 - T_train) ./ M ;
301 mbe2 = sum(T_sim2 - T_test ) ./ N ;
302
303 disp(['训练集数据的为: MBE', num2str(mbe1)])
304 disp(['测试集数据的为: MBE', num2str(mbe2)])
305
306 %% 绘制散点图
307 sz = 25;
308 c = 'b';
309
310 figure
311 scatter(T_train, T_sim1, sz, c)
312 hold on
313 plot(xlim, ylim, '--k')
314 xlabel('训练集真实值');
315 ylabel('训练集预测值');

```

```

316 xlim([min(T_train) max(T_train)])
317 ylim([min(T_sim1) max(T_sim1)])
318 title('训练集预测值 vs. 训练集真实值')
319
320 figure
321 scatter(T_test, T_sim2, sz, c)
322 hold on
323 plot(xlim, ylim, '--k')
324 xlabel('测试集真实值');
325 ylabel('测试集预测值');
326 xlim([min(T_test) max(T_test)])
327 ylim([min(T_sim2) max(T_sim2)])
328 title('测试集预测值 vs. 测试集真实值')
329
330
331 function error = fun(pop, hiddennum, net, p_train, t_train)
332
333 %% 节点个数
334 inputnum = size(p_train, 1); % 输入层节点数
335 outputnum = size(t_train, 1); % 输出层节点数
336
337 %% 提取权值和阈值
338 w1 = pop(1 : inputnum * hiddennum);
339 B1 = pop(inputnum * hiddennum + 1 : inputnum * hiddennum + hiddennum);
340 w2 = pop(inputnum * hiddennum + hiddennum + 1 : ...
341 inputnum * hiddennum + hiddennum * outputnum);
342 B2 = pop(inputnum * hiddennum + hiddennum + hiddennum * outputnum + 1 : ...
343 inputnum * hiddennum + hiddennum + hiddennum * outputnum + outputnum);
344
345 %% 网络赋值
346 net.Iw{1, 1} = reshape(w1, hiddennum, inputnum );
347 net.Lw{2, 1} = reshape(w2, outputnum, hiddennum);
348 net.b{1}     = reshape(B1, hiddennum, 1);
349 net.b{2}     = B2';
350
351 %% 网络训练

```

```
352     net = train(net, p_train, t_train);  
353  
354 %% 仿真测试  
355 t_sim1 = sim(net, p_train);  
356  
357 %% 适应度值  
358 error = sqrt(sum((t_sim1 - t_train) .^ 2) ./ length(t_sim1));
```

---

## 附录3：问题三的源代码

```
1 import pandas as pd  
2 import matplotlib.pyplot as plt  
3 from sklearn.linear_model import LinearRegression  
4 from sklearn.model_selection import train_test_split  
5 from sklearn.metrics import mean_squared_error  
6 import numpy as np  
7  
8 # 加载提供的文件  
9 attachment_1 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\1.xlsx")  
10 attachment_2 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\2.xlsx")  
11 # Display the first few rows of each dataset for a preliminary inspection  
12 attachment_1.head(), attachment_2.head()  
13  
14 # 合并“单品编码”上的数据集  
15 merged_data = pd.merge(attachment_2, attachment_1, on="单品编码", how="left")  
16 # 显示合并数据集的前几行  
17 merged_data.head()  
18 merged_data.to_excel(r"C:\Users\86136\Desktop\merged_dataset.xlsx", index=False)  
19 # 加载数据  
20 merged_data = pd.read_excel(r"C:\Users\86136\Desktop\merged_dataset.xlsx")  
21 # 按品类和商品分类销售量  
22 c_sale = merged_data.groupby('分类名称')[['销量千  
克()']].sum().sort_values(ascending=False)  
23 product_sales = merged_data.groupby('单品名称')[['销量千  
克()']].sum().sort_values(ascending=False)
```

```

24 attachment1 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\1.xlsx")
25 attachment2 = pd.read_csv(r"C:\Users\86136\Desktop\题C(1)附件\2.csv")
26 attachment3 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\3.xlsx")
27 attachment4 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\4.xlsx")
28
29 # 将附件 1 与附件 2 合并以获取每个销售的类别名称
30 rb = attachment2.merge(attachment1, on='单品编码', how='left')
31
32 # 按“分类名称”(类别名称)对合并的数据进行分组，以获得每个类别的总销售额
33 c_sale_distribution = rb.groupby('分类名称')[['销量千
34 克()']].sum().sort_values(ascending=False)
35
36 # 绘制每个类别的销售分布
37 plt.figure(figsize=(10, 6))
38 c_sale_distribution.plot(kind='bar', color='blue')
39 plt.title("销售量分布 - 蔬菜品类")
40 plt.ylabel("销售量 千克()")
41 plt.xlabel("蔬菜品类")
42 plt.xticks(rotation=45)
43 plt.grid(axis='y')
44 plt.show()
45
46 # 按“单品名称”(产品名称)对合并的数据进行分组，以获得每个产品的总销量
47 psd = rb.groupby('单品名称')[['销量千克()']].sum().sort_values(ascending=False)
48
49 # 绘制前 10 种产品的销售分布图
50 plt.figure(figsize=(12, 7))
51 psd.head(10).plot(kind='bar', color='lightgreen')
52 plt.title("销售量分布 - 单品(Top 10)")
53 plt.ylabel("销售量 千克()")
54 plt.xlabel("单品名称")
55 plt.xticks(rotation=45)
56 plt.grid(axis='y')
57 plt.show()
58
# Group by 分类名称， and 单品名称， and sum the sales volume

```

```

59     c_sale = merged_data.groupby('分类名称')[['销量千克']] .sum().sort_values(ascending=False)
60     product_sales = merged_data.groupby('单品名称')[['销量千克']] .sum().sort_values(ascending=False)
61     fig, ax = plt.subplots(2, 1, figsize=(12, 12))
62
63     # 类别销售图
64     c_sale.plot(kind='bar', ax=ax[0], color='teal')
65     ax[0].set_title('Total Sales Volume by Category')
66     ax[0].set_ylabel('Sales Volume (kg)')
67     ax[0].set_xlabel('Category Name')
68
69     # 产品销售图（前 10 名产品）
70     product_sales.head(10).plot(kind='bar', ax=ax[1], color='coral')
71     ax[1].set_title('按时间分类的销售量趋势')
72     ax[1].set_ylabel('销售额 (kg)')
73     ax[1].set_xlabel('产品名称')
74     plt.tight_layout()
75     plt.show()
76
77     # Group by 销售日期, and 分类名称, and sum the sales volume
78     c_d_sales = merged_data.groupby(['销售日期', '分类名称'])[['销量千克']].sum().reset_index()
79
80     plt.figure(figsize=(16, 8))
81     for category in c_d_sales['分类名称'].unique():
82         sub = c_d_sales[c_d_sales['分类名称'] == category]
83         plt.plot(sub['销售日期'], sub['销量千克'], label=category)
84
85     plt.title('按时间分类的销售量趋势')
86     plt.xlabel('时间')
87     plt.ylabel('销量 (kg)')
88     plt.legend()
89     plt.grid(True)
90     plt.tight_layout()
91     plt.show()
92

```

```

93 # 数据使用透视表，获取宽格式的销售数据以进行相关性分析
94 cor_data = c_d_sales.pivot(index='销售日期', columns='分类名称', values='销量千
95 克())
96
97 # 绘制相关矩阵的热图
98 plt.figure(figsize=(10, 6))
99 plt.title('Correlation Between Sales Volumes of Different Categories')
100 plt.xticks(rotation=45)
101 plt.yticks(rotation=45)
102 cax = plt.matshow(cor_matrix, cmap='coolwarm', vmin=-1, vmax=1)
103 plt.colorbar(cax)
104 plt.xticks(range(len(cor_matrix.columns)), cor_matrix.columns)
105 plt.yticks(range(len(cor_matrix.columns)), cor_matrix.columns)
106 plt.show()
107
108 # 以下代码时问题三的第二小问
109 # 加载提取文件
110 attachment_3 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\3.xlsx")
111 attachment_4 = pd.read_excel(r"C:\Users\86136\Desktop\题C(1)附件\4.xlsx")
112 attachment_3.head(), attachment_4.head()
113
114 daily_sales = merged_data.groupby(['销售日期', '分类名称'])['销量千
115 克()'].sum().reset_index()
116 # 按日期和类别分组并进行汇总
117 def forecast_sales(category):
118     category_data = daily_sales[daily_sales['分类名
119 称'] == category].reset_index(drop=True)
120     category_data['day_num'] = np.arange(len(category_data))
121
122 X = category_data[['day_num']]
123 y = category_data['销量千克()']
124
125 # 将数据拆分为训练集和测试集80%20%
126 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
127 random_state=42, shuffle=False)

```

```

126 # 训练线性回归模型
127 model = LinearRegression()
128 model.fit(X_train, y_train)
129
130 # 预测测试集的销售情况
131 y_pred = model.predict(X_test)
132
133 # 计算预测的均方误差
134 mse = mean_squared_error(y_test, y_pred)
135
136 # 预测未来 7 天的销售额
137 next_7_days = np.array([max(category_data['day_num']) + i for i in range(1,
138 8)]).reshape(-1, 1)
139 future_forecast = model.predict(next_7_days)
140
141 return future_forecast, mse
142
143 forecasts = {}
144 errors = {}
145 for category in daily_sales['分类名称'].unique():
146     forecasts[category], errors[category] = forecast_sales(category)
147
148 # 为整个数据帧重新生成列daily_salesday_num
149 daily_sales['day_num'] = daily_sales.groupby('分类名称').cumcount()
150
151 # 基于预测和误差的可视化
152 plt.figure(figsize=(16, 8))
153
154 for category in daily_sales['分类名称'].unique():
155     category_data = daily_sales[daily_sales['分类名
156 称'] == category].reset_index(drop=True)
157     X = category_data[['day_num']]
158     y = category_data['销量千克()']
159     _, X_test, _, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
160                                             shuffle=False)

```

```

159 # Plotting the actual test data
160 plt.scatter(X_test.day_num, y_test, label=f"Actual {category}", marker='x')
161
162 # Plotting the forecasted data
163 next_7_days = np.array([max(category_data['day_num']) + i for i in range(1, 8)])
164 plt.plot(next_7_days, forecasts[category], label=f"Forecast {category}")
165
166 plt.title('销售预测与实际销售')
167 plt.xlabel('时间')
168 plt.ylabel('销量 (kg)')
169 plt.legend(loc='upper left')
170 plt.grid(True, which='both', linestyle='--', linewidth=0.5)
171 plt.tight_layout()
172 plt.show()

173
174 #绘制不同的类别的均方误差
175 plt.figure(figsize=(10, 6))
176 plt.bar(errors.keys(), errors.values(), color='skyblue')
177 plt.title('每个类别的MSE')
178 plt.xlabel('类别')
179 plt.ylabel('MSE')
180 plt.grid(axis='y', linestyle='--', linewidth=0.5)
181 plt.tight_layout()
182 plt.show()

183
184 pricing_data = pd.merge(attachment_3, attachment_1, on="单品编码", how="left")
185 avg_wholesale_price = pricing_data.groupby('分类名称')[['批发价格元千
186 克(/)']].mean()
187 cost_data = pd.merge(avg_wholesale_price, attachment_4, left_on='分类名
188 称', right_on='小分类名称', how="left")
189 cost_data['cost_per_kg'] = cost_data[['批发价格元千克(/)']] * (1 + cost_data['平均
190 损耗率(%)小分类编码不同值_']] / 100)
191 cost_data = cost_data[['小分类名称', 'cost_per_kg']]# 提取相关列

192
193 # 定义一个函数以根据预测的销量设置加价率
194 def determine_markup_rate(sales_forecast):
195     if sales_forecast < 20:

```

```

193     return 1.5 # 150% markup for low volume items
194
195     elif sales_forecast < 50:
196
197         return 1.3 # 130% markup for medium volume items
198
199     else:
200
201         return 1.2 # 120% markup for high volume items
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223

```

# 确定下周的价格策略

```

price_strategy = {}

for category, forecast in forecasts.items():
    cost = cost_data[cost_data['小分类名称'] == category]['cost_per_kg'].values[0]
    markup_rates = [determine_markup_rate(f) for f in forecast]
    prices = [cost * rate for rate in markup_rates]
    price_strategy[category] = prices

```

# 按“单品名称”分组，获取整个数据集的平均销售额

```

# 所有单品名称平均销量值#
avg_sales_overall = merged_data.groupby('单品名称')[['销量千
克()']].mean().reset_index()
# 将此用作 7 月 1 日的预测销售额
forecasted_sales_july1_overall = avg_sales_overall.copy()
forecasted_sales_july1_overall.rename(columns={'销量千克()': '预测销量月
日_71'}, inplace=True)
forecasted_sales_july1_overall.head()
# 按照以前的销量从而得到月日的预测销售量71
# 将预测销售额与定价和损失率数据合并
product_cost_data = pd.merge(forecasted_sales_july1_overall, pricing_data,
    on='单品名
称', how="left")
# 按照单品名称标准输出排列
product_cost_data = pd.merge(product_cost_data, attachment_4, left_on='分类名
称', right_on='小分类名称', how="left")
# 分类名称排在左边，小分类名称排在右边
product_cost_data['cost_per_kg'] = product_cost_data[['批发价格元千
克(/)'] * (1 + product_cost_data['平均损耗率(%)小分类编码不同值__'] / 100)
# 计算cost_per_kg
product_cost_data[['单品名称', '预测销量月日_71', '批发价格元千克(/)', '平均损耗
率(%)小分类编码不同值__', 'cost_per_kg']].head()

```

```

224
225 # 定义一个函数以根据单个产品的预测销量设置加价率
226 def determine_product_markup_rate(sales_forecast):
227     if sales_forecast < 0.5:
228         return 1.5 # 150% markup for low volume items
229     elif sales_forecast < 1:
230         return 1.3 # 130% markup for medium volume items
231     else:
232         return 1.2 # 120% markup for high volume items
233
234
235 # 对于函数下定义去设置市场率基于预测的销量对于单品
236 # 计算每个产品的预期利润
237 product_cost_data['markup_rate'] = product_cost_data['预测销量月'
    日_71'].apply(determine_product_markup_rate)
238 # 月日预测销量决定相应产品市场率71
239 product_cost_data['expected_price'] = product_cost_data['cost_per_kg'] *
    product_cost_data['markup_rate']
240 # 期望未来价格
241 product_cost_data['expected_profit_per_kg'] =
    product_cost_data['expected_price'] - product_cost_data['cost_per_kg']
242 # 期望未来每千克利润
243 product_cost_data['total_expected_profit'] =
    product_cost_data['expected_profit_per_kg'] * product_cost_data[
        '预测销量月日_71']
244 # 总期望利润
245 # 根据预期利润对产品进行排序
246 sorted_products = product_cost_data.sort_values(by='total_expected_profit',
    ascending=False).drop_duplicates(
247     sub='单品名称')
248
249 sorted_products[['单品名称', '预测销量月'
    日_71', 'cost_per_kg', 'expected_price', 'total_expected_profit']].head()
250
251 # 根据预期利润选择前 27-33 名产品
252 selected_products = sorted_products.head(33)
253 # 计算这些选定产品的总预期利润
254 total_expected_profit = selected_products['total_expected_profit'].sum()

```

```
255 # 设置每个选定产品的补货量  
256 min_display_volume = 2.5  
257 selected_products['replenishment_volume'] = selected_products['预测销量月  
日_71'].apply(lambda x: max(x, min_display_volume))  
258 #显示出所选产品、预期价格和补货量  
259 selected_products_final = selected_products[['单品名  
称', 'expected_price', 'replenishment_volume']]  
260 selected_products_final.to_excel(r"C:\Users\86136\Desktop\selected_products_final.xlsx",  
index=False)
```

---

## 附录4：PSO-BP模型的相关预测图

训练集数据的R2为： 0.58303  
测试集数据的R2为： 0.28928  
训练集数据的MAE为： 36.1413  
测试集数据的MAE为： 30.3915  
训练集数据的MBE为： 0.53034  
测试集数据的MBE为： 5.1263

图 23

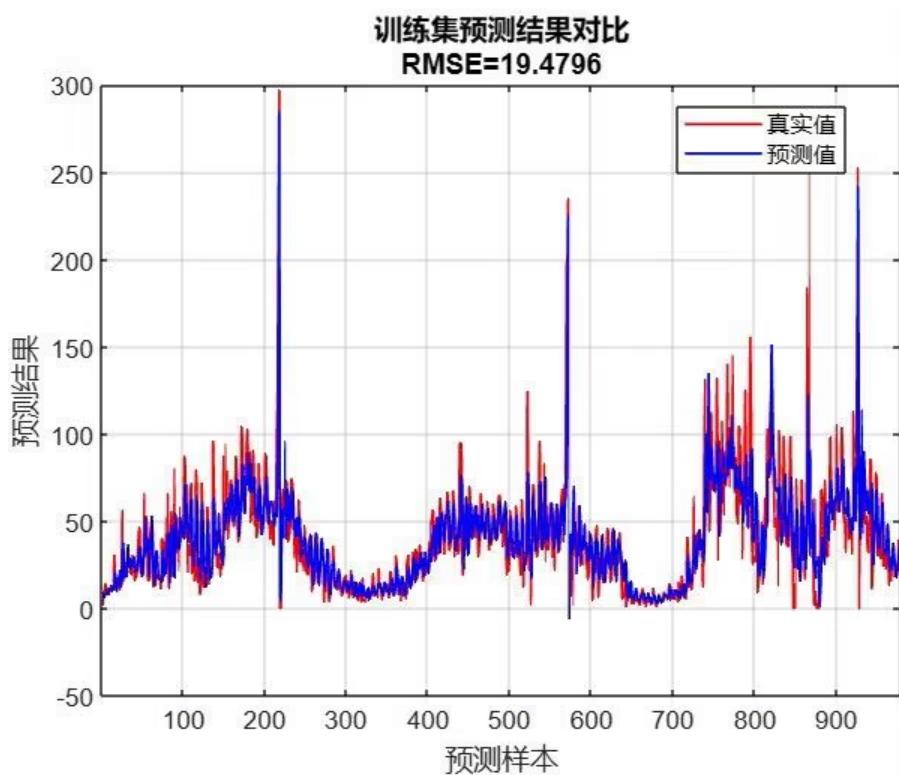


图 24

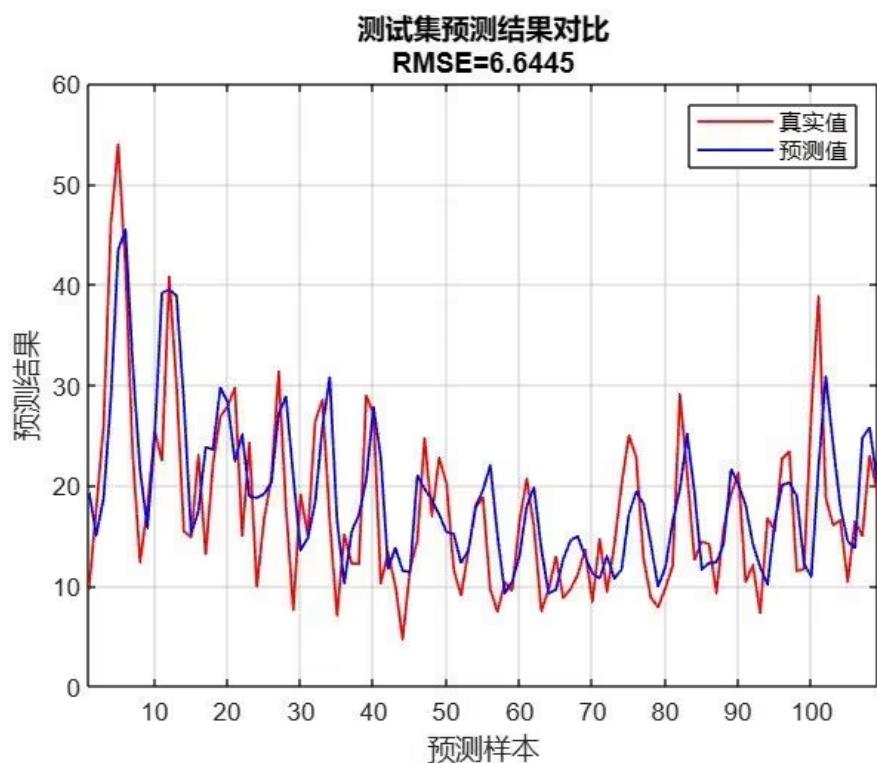


图 25

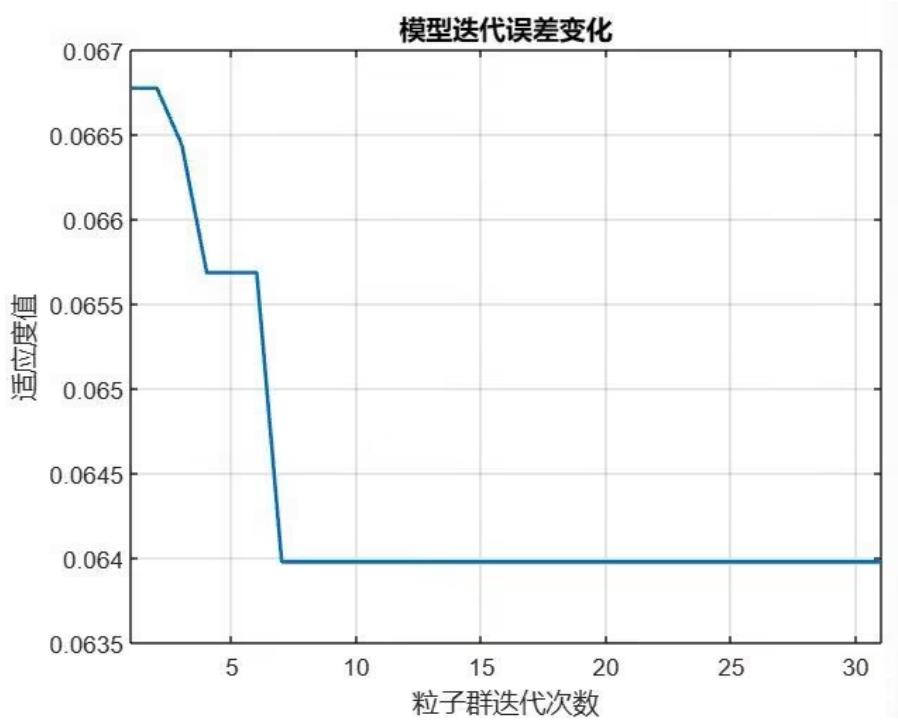


图 26

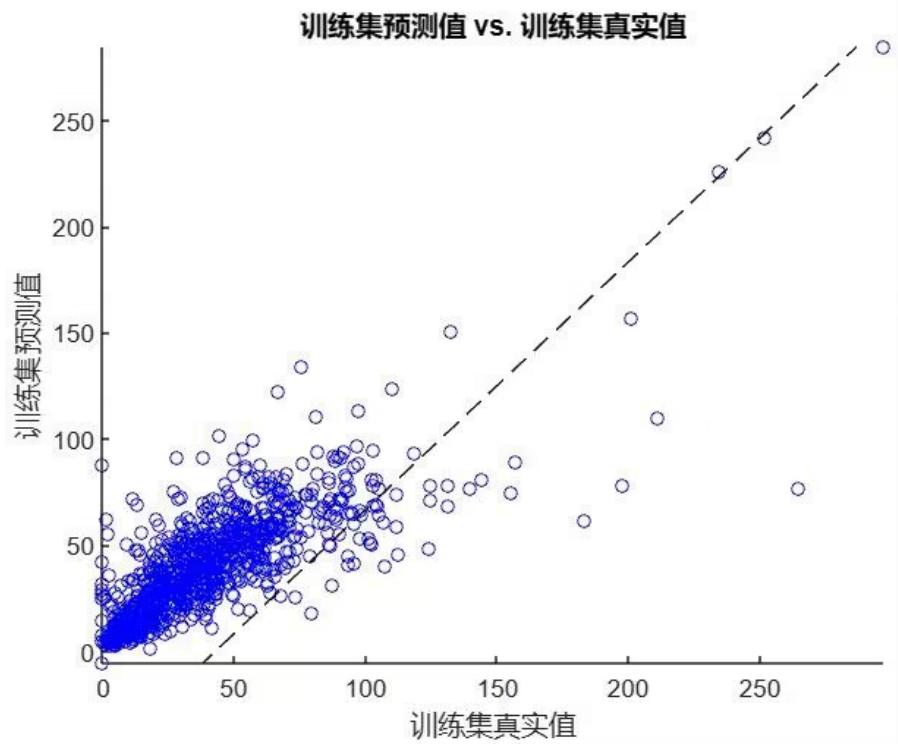


图 27

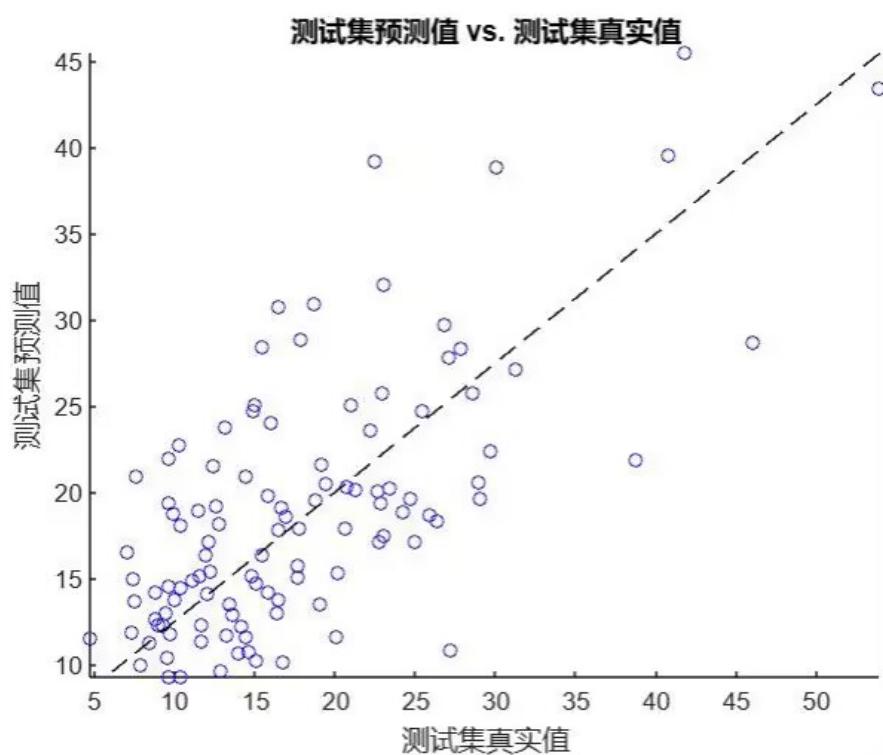


图 28

训练集数据的R2为： 0.63526  
测试集数据的R2为： 0.40388  
训练集数据的MAE为： 12.32  
测试集数据的MAE为： 5.1942  
训练集数据的MBE为： 0.15098  
测试集数据的MBE为： 1.2449

图 29

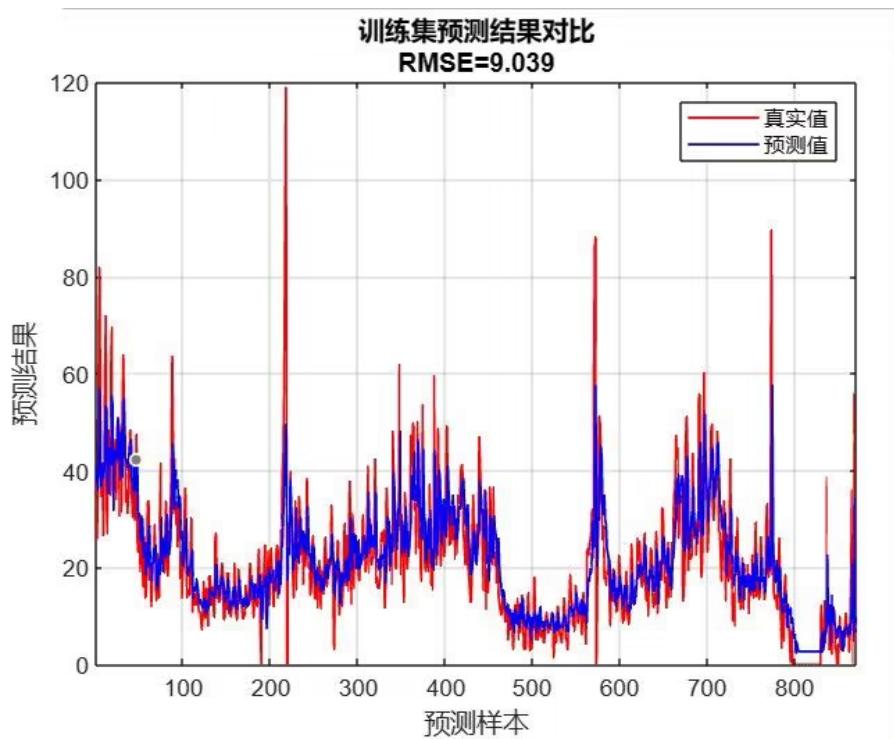


图 30

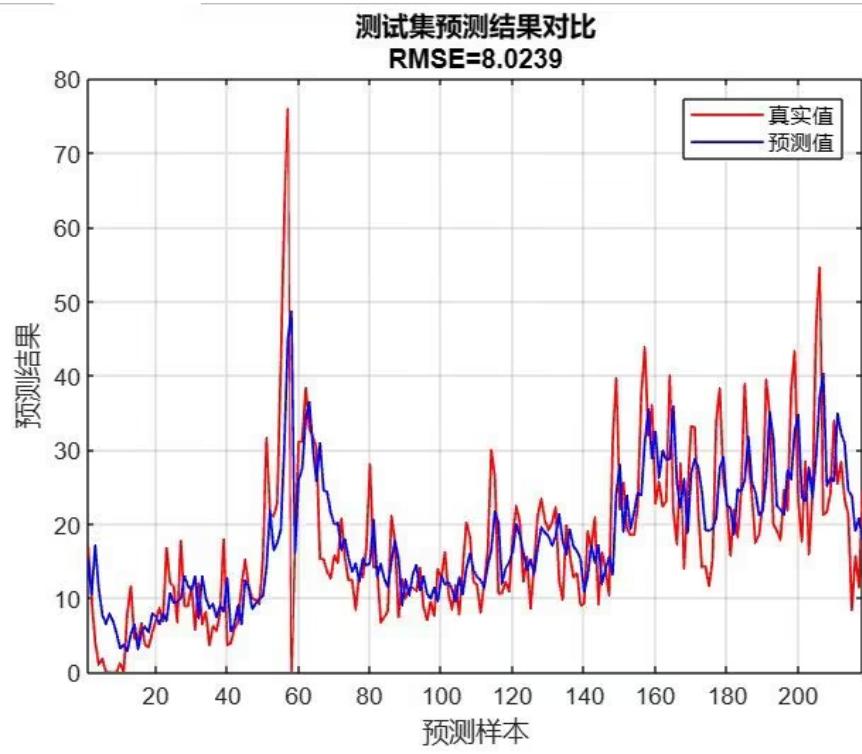


图 31

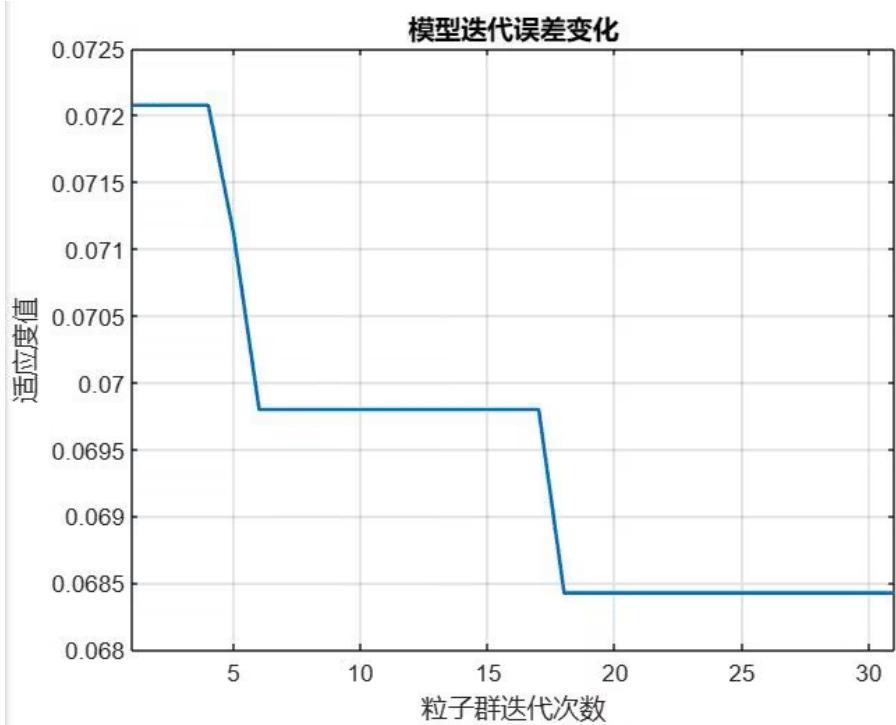


图 32

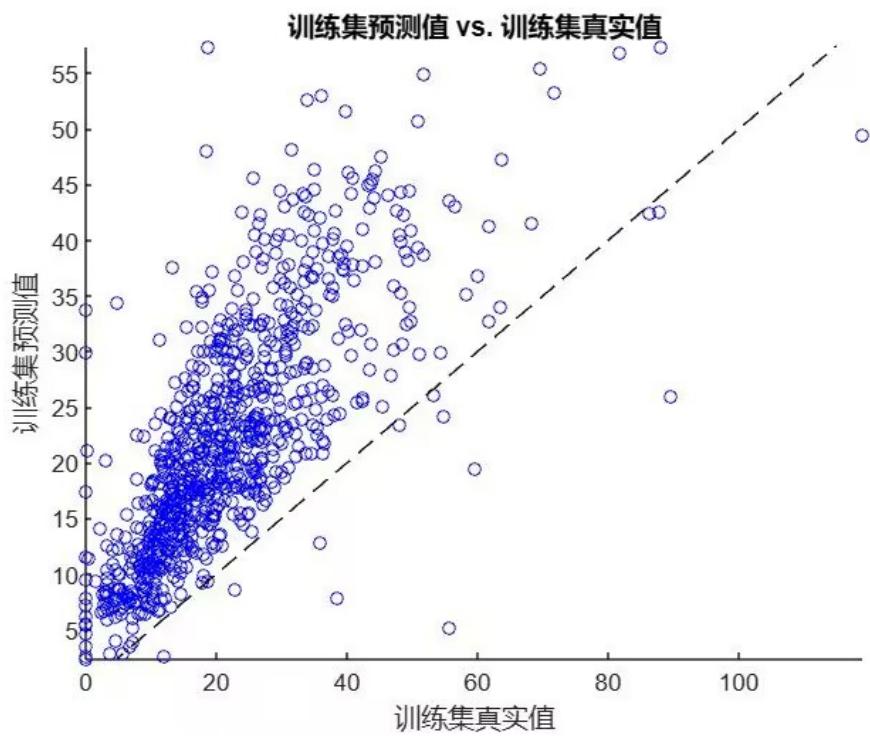


图 33

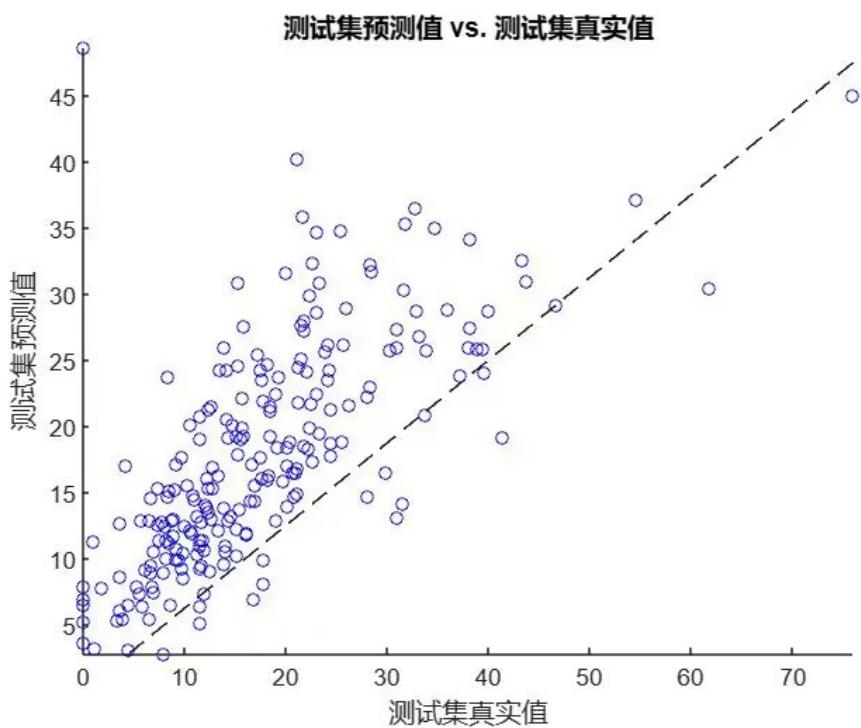


图 34

训练集数据的R2为：0.57704  
测试集数据的R2为：0.48846  
训练集数据的MAE为：6.0269  
测试集数据的MAE为：5.649  
训练集数据的MBE为：0.43148  
测试集数据的MBE为：0.39799

图 35

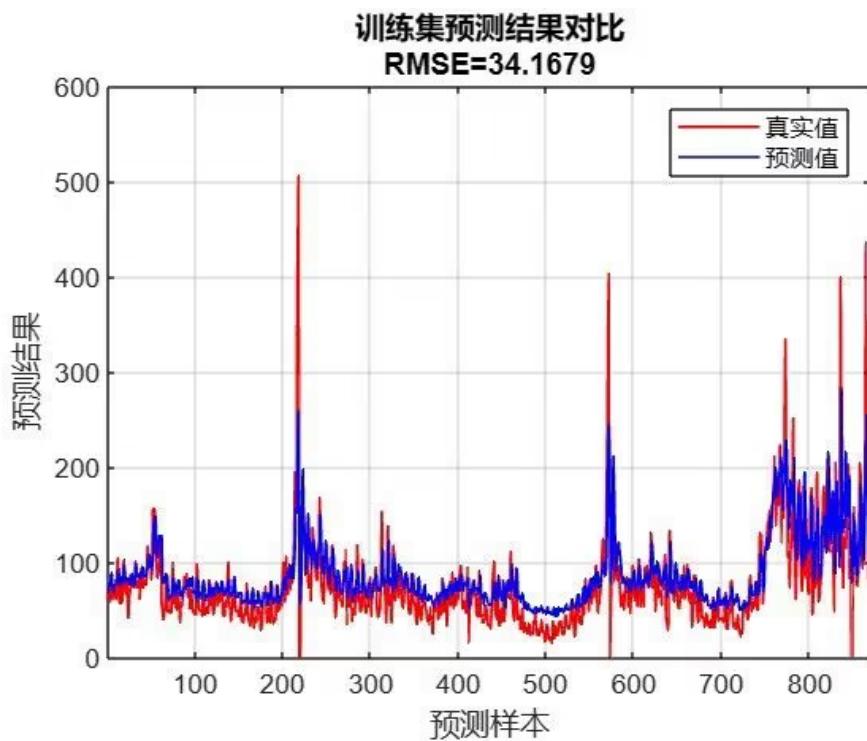


图 36

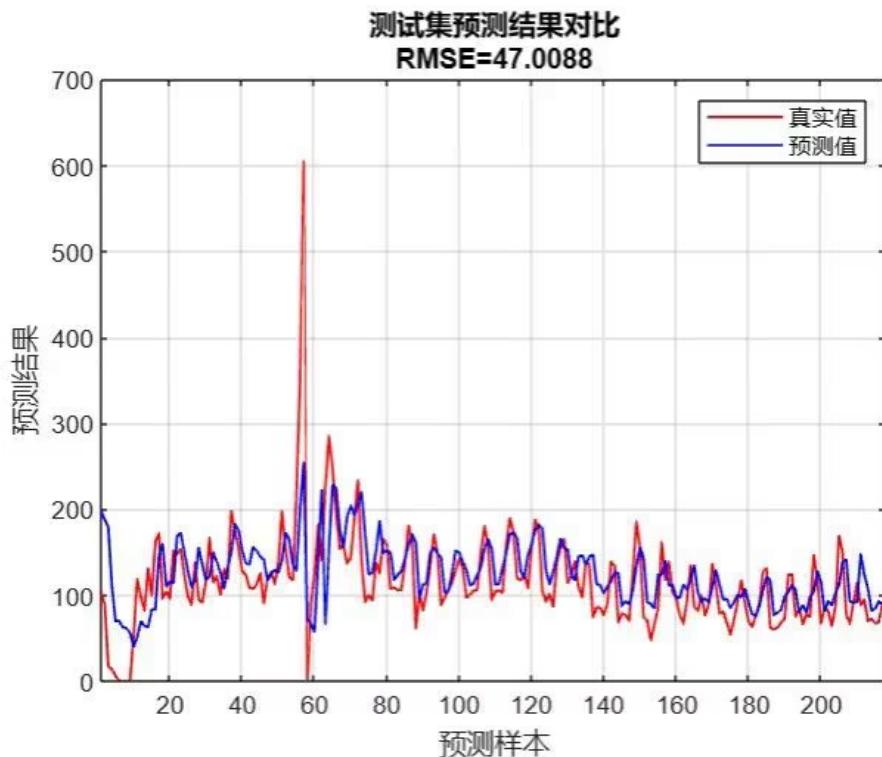


图 37

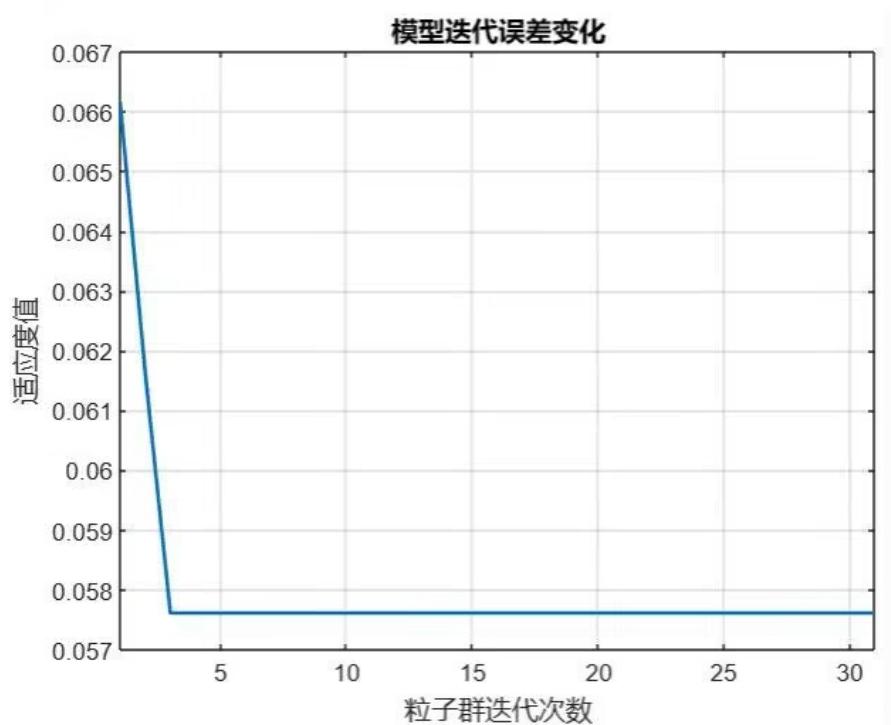


图 38

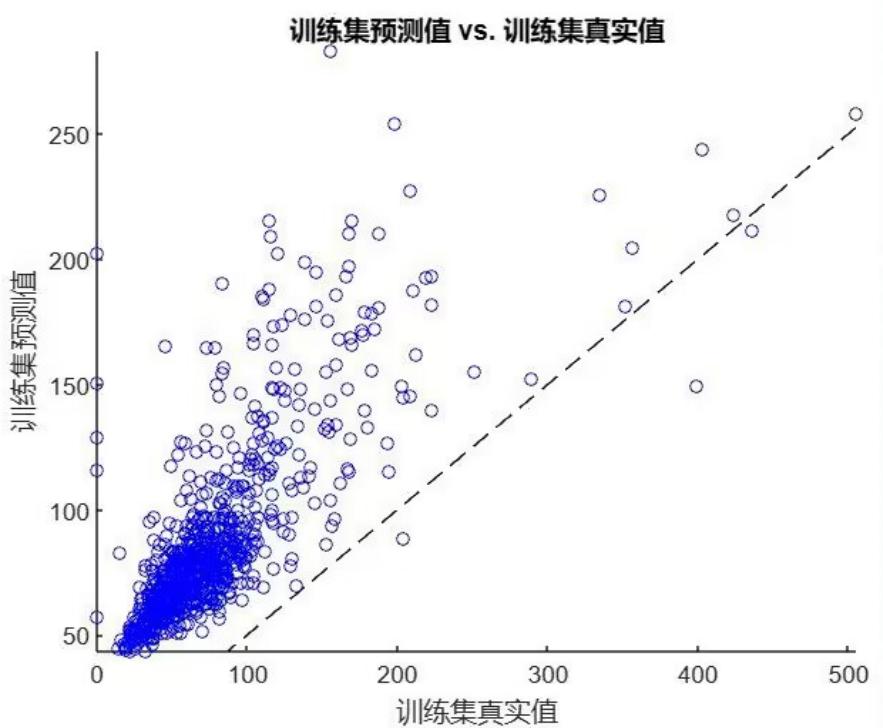


图 39

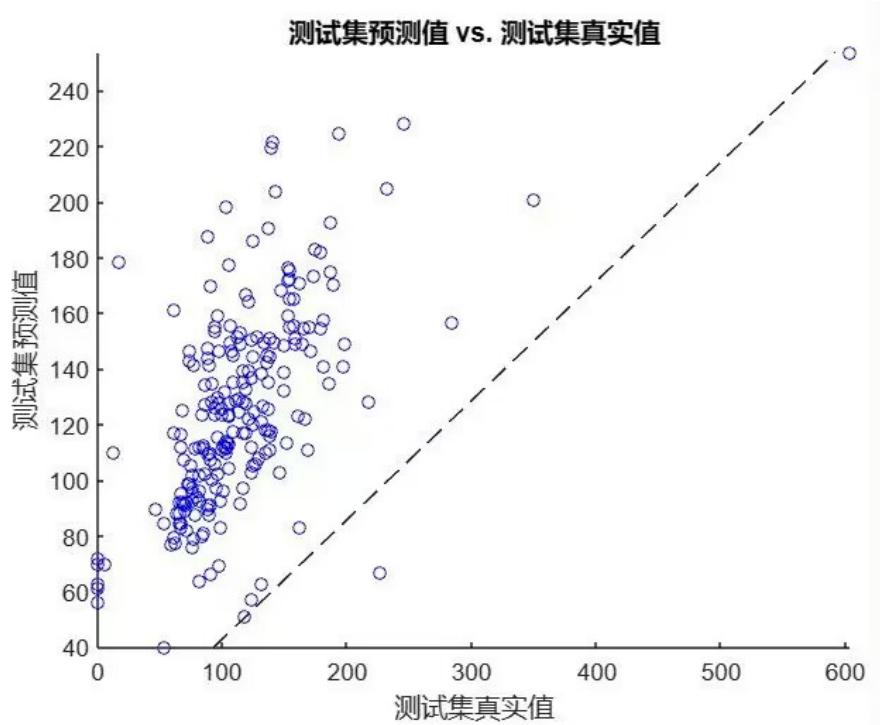


图 40

---

训练集数据的R2为： 0.53401  
测试集数据的R2为： 0.3297  
训练集数据的MAE为： 22.642  
测试集数据的MAE为： 31.2966  
训练集数据的MBE为： 10.4843  
测试集数据的MBE为： 10.5201

图 41

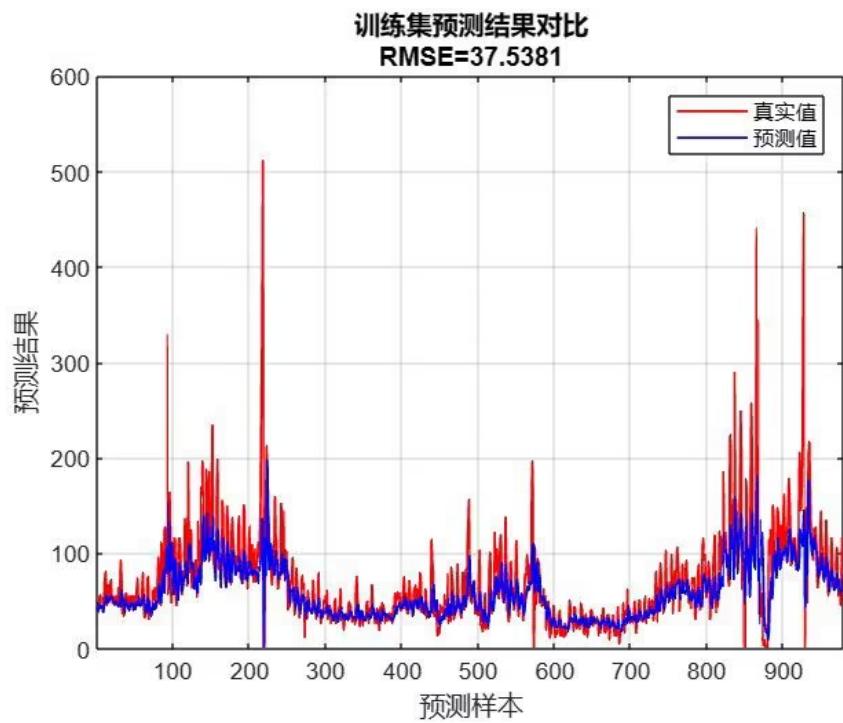


图 42

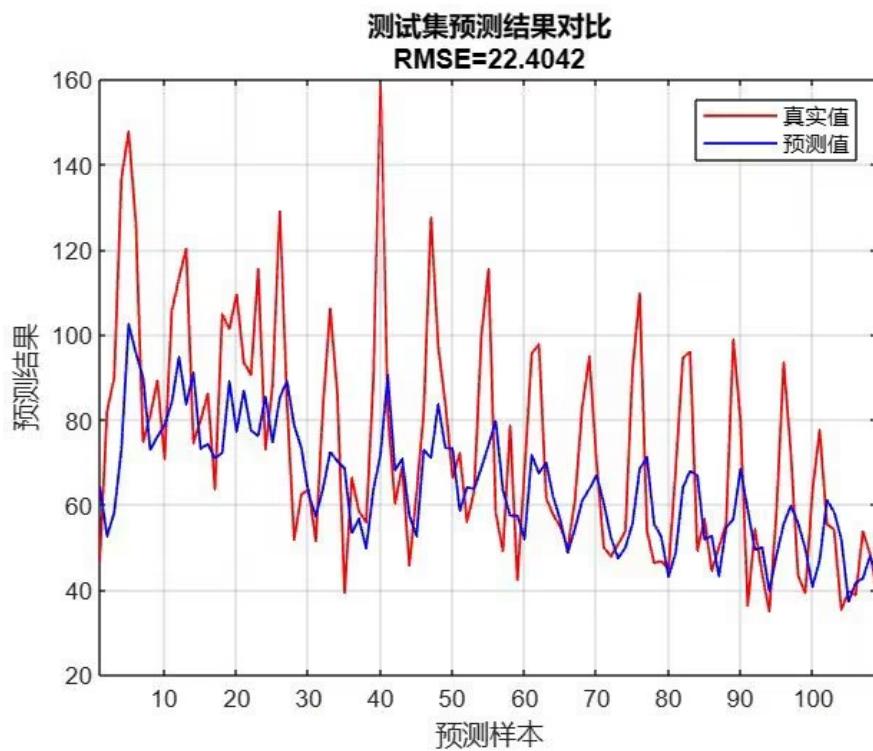


图 43

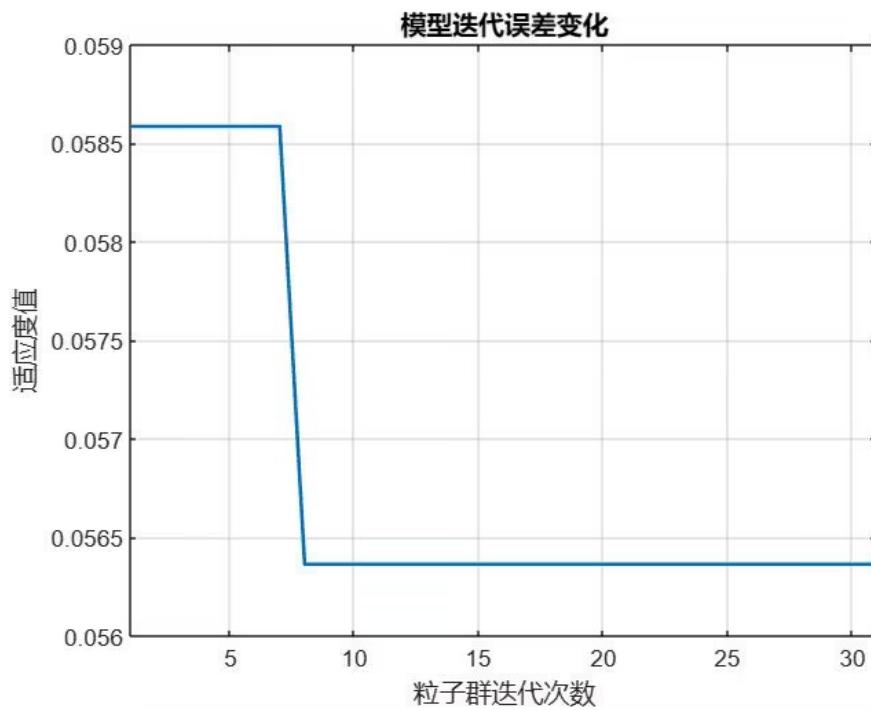


图 44

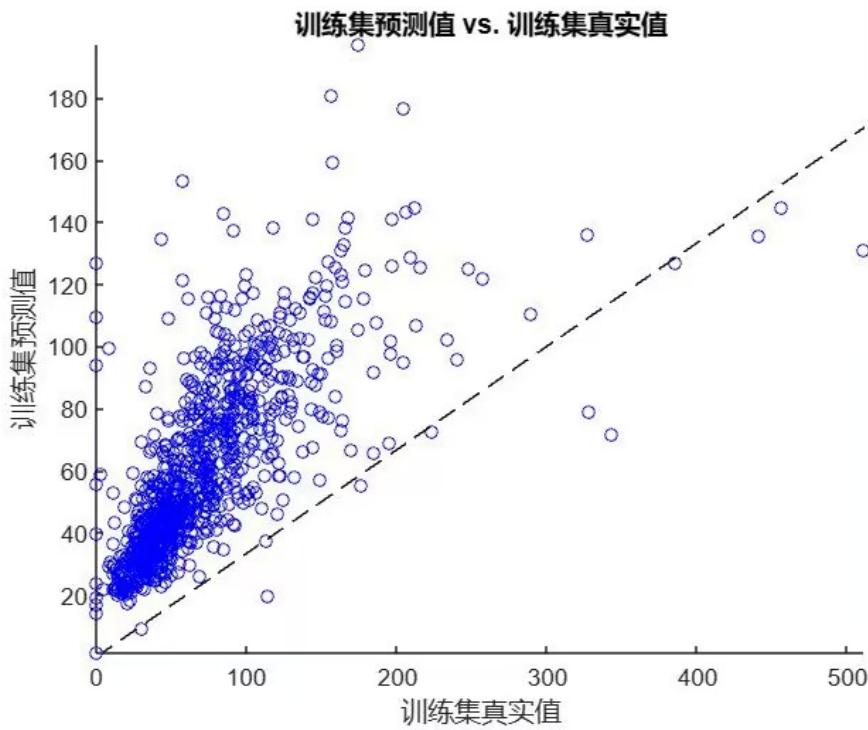


图 45

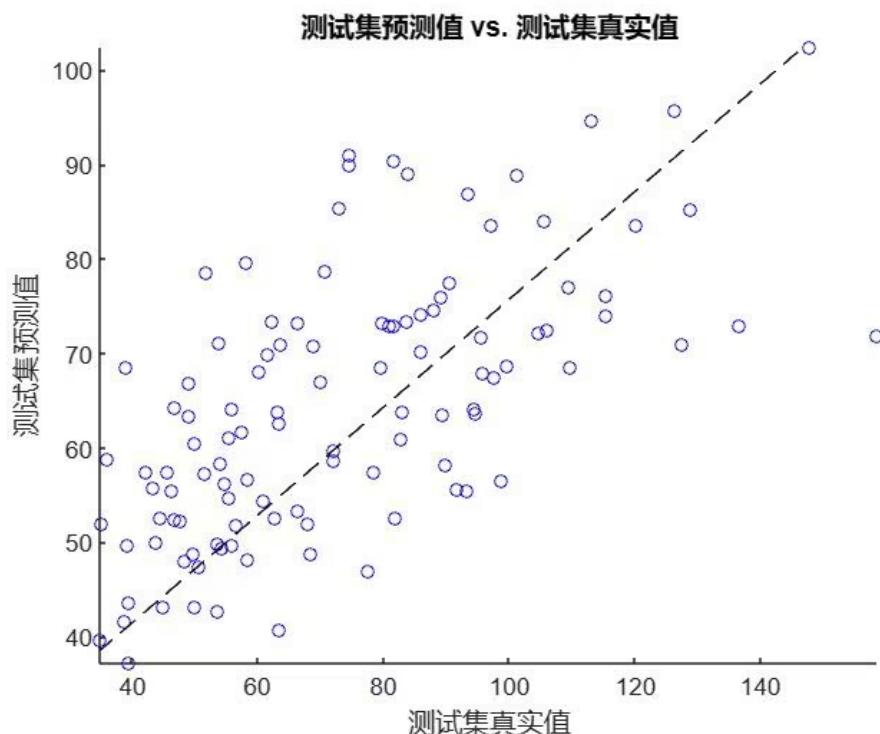


图 46

训练集数据的R2为：0.45227  
 测试集数据的R2为：0.28373  
 训练集数据的MAE为：20.323  
 测试集数据的MAE为：16.9284  
 训练集数据的MBE为：-10.2418  
 测试集数据的MBE为：-9.0729

图 47

## 附录5：关联规则结果图

support itemsets 0 0.001314 (七彩椒(1)) 1 0.000142 (七彩椒(2)) 2 0.027229 (上海青) 3 0.001275 (上海青(份)) 4 0.007654 (东门口小白菜) 5 0.041915 (云南油麦菜) 6 0.001848 (云南油麦菜(份)) 7 0.071153 (云南生菜) 8 0.002088 (云南生菜(份)) 9 0.000224 (余干椒) 10 0.001203 (保康高山大白菜) 11 0.054222 (净藕(1)) 12 0.004570 (双孢菇) 13 0.001382 (双孢菇(盒)) 14 0.000196 (和丰阳光海鲜菇(包)) 15 0.002053 (四川红香椿) 16 0.002833 (圆茄子(2)) 17 0.002582 (外地茼蒿) 18 0.026119 (大白菜) 19 0.004448 (大龙茄子) 20 0.019934 (奶白菜) 21 0.000292 (奶白菜(份)) 22 0.000303 (姜蒜小米椒组合装(小份)) 23 0.003371 (姬菇(1)) 24 0.000139 (姬

菇(2)) 25 0.000155 (姬菇(份)) 26 0.001955 (姬菇(包)) 27 0.006712 (娃娃菜) 28 0.007105 (小白菜) 29 0.000102 (小白菜(份)) 30 0.000843 (小皱皮) 31 0.001338 (小皱皮(份)) 32 0.023853 (小米椒) 33 0.004210 (小米椒(份)) 34 0.008696 (小青菜(1)) 35 0.000519 (小青菜(2)) 36 0.000357 (小青菜(份)) 37 0.012660 (平菇) 38 0.000466 (快菜) 39 0.004853 (木耳菜) 40 0.002286 (本地上海青) 41 0.013974 (杏鲍菇(1)) 42 0.000560 (杏鲍菇(2)) 43 0.000185 (杏鲍菇(份)) 44 0.002332 (杏鲍菇(袋)) 45 0.010749 (枝江红菜苔) 46 0.000554 (枝江红菜苔(份)) 47 0.001443 (枝江青梗散花) 48 0.000146 (水果辣椒) 49 0.000517 (水果辣椒(份)) 50 0.000170 (油菜苔) 51 0.046662 (泡泡椒(精品)) 52 0.004073 (洪湖莲藕(粉藕)) 53 0.002378 (洪湖藕带) 54 0.004071 (海鲜菇(1)) 55 0.000358 (海鲜菇(2)) 56 0.000211 (海鲜菇(份)) 57 0.000634 (海鲜菇(包)) 58 0.002288 (海鲜菇(袋)(1)) 59 0.001903 (海鲜菇(袋)(3)) 60 0.003871 (海鲜菇(袋)(4)) 61 0.001051 (灯笼椒(1)) 62 0.000168 (灯笼椒(2)) 63 0.014793 (牛首油菜) 64 0.003494 (牛首生菜) 65 0.009458 (甜白菜) 66 0.000318 (田七) 67 0.000449 (白玉菇(2)) 68 0.000290 (白玉菇(盒)) 69 0.003725 (白玉菇(袋)) 70 0.002280 (白菜苔) 71 0.000115 (秀珍菇) 72 0.023615 (竹叶菜) 73 0.000674 (竹叶菜(份)) 74 0.000261 (紫苏) 75 0.037962 (紫茄子(2)) 76 0.000237 (紫贝菜) 77 0.003448 (红尖椒) 78 0.000427 (红尖椒(份)) 79 0.007931 (红杭椒) 80 0.000416 (红杭椒(份)) 81 0.021493 (红椒(1)) 82 0.000161 (红椒(份)) 83 0.001397 (红灯笼椒(1)) 84 0.000218 (红灯笼椒(2)) 85 0.000456 (红线椒) 86 0.000192 (红莲藕带) 87 0.021772 (红薯尖) 88 0.000275 (红薯尖(份)) 89 0.001161 (组合椒系列) 90 0.085622 (芜湖青椒(1)) 91 0.000181 (芥菜) 92 0.000357 (花茄子) 93 0.015377 (苋菜) 94 0.013617 (茼蒿) 95 0.001026 (茼蒿(份)) 96 0.003999 (葶苈) 97 0.000527 (莲蓬(个)) 98 0.017773 (菜心) 99 0.000996 (菜心(份)) 100 0.023406 (菠菜) 101 0.001985 (菠菜(份)) 102 0.000281 (菱角) 103 0.001449 (萝卜叶) 104 0.003526 (蔡甸藜蒿) 105 0.000497 (虫草花) 106 0.000157 (虫草花(份)) 107 0.001774 (虫草花(袋)) 108 0.037092 (螺丝椒) 109 0.000484 (螺丝椒(份)) 110 0.000166 (蟹味菇(1)) 111 0.000172 (蟹味菇(2)) 112 0.000148 (蟹味菇(盒)) 113 0.000717 (蟹味菇(袋)) 114 0.078485 (西兰花) 115 0.013647 (西峡花菇(1)) 116 0.071288 (西峡香菇(1)) 117 0.000475 (豌豆尖) 118 0.000837 (辣妹子) 119 0.000128 (野生粉藕) 120 0.019948 (金针菇(1)) 121 0.000556 (金针菇(2)) 122 0.003365 (金针菇(盒)) 123 0.003507 (金针菇(袋)(1)) 124 0.005383 (金针菇(袋)(2)) 125 0.004561 (金针菇(袋)(3)) 126 0.000118 (银耳(朵)) 127 0.009232 (长线茄) 128 0.001759 (随州泡泡青) 129 0.006525 (青尖椒) 130 0.000615 (青尖椒(份)) 131 0.001166 (青杭椒(2)) 132 0.000464 (青杭椒(份)) 133 0.027433 (青梗散花) 134 0.018934 (青线椒) 135 0.000591 (青线椒(份)) 136 0.009484 (青茄子(1)) 137 0.000113 (青菜苔) 138 0.000347 (马齿苋) 139 0.005006 (高瓜(1)) 140 0.000251 (高瓜(2)) 141 0.000194 (鱼腥草) 142 0.001552 (鲜木耳(1)) 143 0.000155 (鲜木耳(2)) 144 0.000129 (鲜木耳(份)) 145 0.008212 (黄心菜(1)) 146 0.001220 (黄心菜(2)) 147 0.022386 (黄白菜(2)) 148 0.000301 (云南油麦菜, 上海青) 149 0.000456 (上海青, 云南生菜) 150 0.000347 (上海青, 净藕(1)) 151 0.000142 (大白菜, 上海青)

152 0.000164 (小米椒, 上海青) 153 0.000109 (平菇, 上海青) 154 0.000164 (上海青, 杏鲍菇(1))  
155 0.000632 (上海青, 泡泡椒(精品)) 156 0.000102 (甜白菜, 上海青) 157 0.000113 (竹叶菜, 上海青)  
158 0.000257 (上海青, 紫茄子(2)) 159 0.000137 (上海青, 红椒(1)) 160 0.000129 (上海青, 红薯尖)  
161 0.000462 (芜湖青椒(1), 上海青) 162 0.000109 (苋菜, 上海青) 163 0.000207 (菜心, 上海青)  
164 0.000177 (菠菜, 上海青) 165 0.000312 (螺丝椒, 上海青) 166 0.000686 (西兰花, 上海青)  
167 0.000767 (西峡香菇(1), 上海青) 168 0.000188 (金针菇(1), 上海青) 169 0.000273 (上海青, 青梗散花)  
170 0.000163 (上海青, 青线椒) 171 0.000190 (上海青, 黄白菜(2)) 172 0.000155  
(东门口小白菜, 奶白菜) 173 0.000177 (东门口小白菜, 芜湖青椒(1)) 174 0.000124 (东门口小白菜, 西兰花)  
175 0.000124 (东门口小白菜, 西峡香菇(1)) 176 0.000793 (云南油麦菜, 云南生菜)  
177 0.000628 (云南油麦菜, 净藕(1)) 178 0.000301 (云南油麦菜, 大白菜) 179 0.000118 (云南油麦菜, 大龙茄子)  
180 0.000144 (云南油麦菜, 奶白菜) 181 0.000277 (云南油麦菜, 小米椒) 182  
0.000188 (云南油麦菜, 平菇) 183 0.000242 (云南油麦菜, 杏鲍菇(1)) 184 0.000157 (云南油麦菜, 枝江红菜苔)  
185 0.000992 (云南油麦菜, 泡泡椒(精品)) 186 0.000164 (牛首油菜, 云南油麦菜)  
187 0.000166 (云南油麦菜, 甜白菜) 188 0.000235 (云南油麦菜, 竹叶菜) 189 0.000519 (云南油麦菜, 紫茄子(2))  
190 0.000262 (云南油麦菜, 红椒(1)) 191 0.000216 (云南油麦菜, 红薯尖)  
192 0.000845 (芜湖青椒(1), 云南油麦菜) 193 0.000122 (云南油麦菜, 苋菜) 194 0.000207 (云南油麦菜, 茼蒿)  
195 0.000279 (菜心, 云南油麦菜) 196 0.000294 (云南油麦菜, 菠菜) 197 0.000475  
(云南油麦菜, 螺丝椒) 198 0.000955 (云南油麦菜, 西兰花) 199 0.000120 (云南油麦菜, 西峡花菇(1))  
200 0.000693 (西峡香菇(1), 云南油麦菜) 201 0.000275 (云南油麦菜, 金针菇(1)) 202  
0.000113 (云南油麦菜, 青尖椒) 203 0.000419 (云南油麦菜, 青梗散花) 204 0.000290 (云南油麦菜, 青线椒)  
205 0.000163 (云南油麦菜, 青茄子(1)) 206 0.000259 (云南油麦菜, 黄白菜(2)) 207  
0.000913 (云南生菜, 净藕(1)) 208 0.000115 (云南生菜, 双孢菇) 209 0.000344 (大白菜, 云南生菜)  
210 0.000285 (云南生菜, 奶白菜) 211 0.000120 (姬菇(1), 云南生菜) 212 0.000170 (云南生菜, 娃娃菜)  
213 0.000115 (小白菜, 云南生菜) 214 0.000429 (小米椒, 云南生菜) 215 0.000128  
(小青菜(1), 云南生菜) 216 0.000251 (平菇, 云南生菜) 217 0.000358 (云南生菜, 杏鲍菇(1)) 218  
0.000113 (枝江红菜苔, 云南生菜) 219 0.001181 (云南生菜, 泡泡椒(精品)) 220 0.000102 (海鲜菇(1), 云南生菜)  
221 0.000233 (牛首油菜, 云南生菜) 222 0.000224 (甜白菜, 云南生菜) 223 0.000377  
(竹叶菜, 云南生菜) 224 0.000711 (紫茄子(2), 云南生菜) 225 0.000111 (云南生菜, 红杭椒)  
226 0.000334 (红椒(1), 云南生菜) 227 0.000370 (云南生菜, 红薯尖) 228 0.001223 (芜湖青椒(1), 云南生菜)  
229 0.000224 (苋菜, 云南生菜) 230 0.000370 (茼蒿, 云南生菜) 231 0.000312  
(菜心, 云南生菜) 232 0.000538 (菠菜, 云南生菜) 233 0.000614 (螺丝椒, 云南生菜) 234 0.001767  
(西兰花, 云南生菜) 235 0.000181 (西峡花菇(1), 云南生菜) 236 0.001314 (西峡香菇(1), 云南生菜)  
237 0.000745 (金针菇(1), 云南生菜) 238 0.000124 (金针菇(袋)(1), 云南生菜) 239 0.000124  
(金针菇(袋)(3), 云南生菜) 240 0.000124 (长线茄, 云南生菜) 241 0.000128 (青尖椒, 云南生

菜) 242 0.000497 (云南生菜, 青梗散花) 243 0.000373 (青线椒, 云南生菜) 244 0.000159 (青茄子(1), 云南生菜) 245 0.000177 (黄心菜(1), 云南生菜) 246 0.000408 (云南生菜, 黄白菜(2)) 247 0.000403 (大白菜, 净藕(1)) 248 0.000286 (净藕(1), 奶白菜) 249 0.000133 (娃娃菜, 净藕(1)) 250 0.000401 (小米椒, 净藕(1)) 251 0.000116 (小青菜(1), 净藕(1)) 252 0.000421 (平菇, 净藕(1)) 253 0.000225 (净藕(1), 杏鲍菇(1)) 254 0.000253 (枝江红菜苔, 净藕(1)) 255 0.001105 (泡泡椒(精品), 净藕(1)) 256 0.000456 (牛首油菜, 净藕(1)) 257 0.000161 (甜白菜, 净藕(1)) 258 0.000192 (竹叶菜, 净藕(1)) 259 0.000469 (紫茄子(2), 净藕(1)) 260 0.000103 (净藕(1), 红杭椒) 261 0.000371 (红椒(1), 净藕(1)) 262 0.000249 (净藕(1), 红薯尖) 263 0.000893 (芜湖青椒(1), 净藕(1)) 264 0.000142 (苋菜, 净藕(1)) 265 0.000379 (茼蒿, 净藕(1)) 266 0.000290 (菜心, 净藕(1)) 267 0.000477 (菠菜, 净藕(1)) 268 0.000395 (螺丝椒, 净藕(1)) 269 0.001271 (西兰花, 净藕(1)) 270 0.000227 (西峡花菇(1), 净藕(1)) 271 0.001611 (西峡香菇(1), 净藕(1)) 272 0.000510 (金针菇(1), 净藕(1)) 273 0.000523 (净藕(1), 青梗散花) 274 0.000329 (青线椒, 净藕(1)) 275 0.000111 (青茄子(1), 净藕(1)) 276 0.000126 (黄心菜(1), 净藕(1)) 277 0.000288 (净藕(1), 黄白菜(2)) 278 0.000102 (西兰花, 双孢菇) 279 0.000146 (大白菜, 小米椒) 280 0.000200 (大白菜, 平菇) 281 0.000133 (大白菜, 杏鲍菇(1)) 282 0.000135 (大白菜, 枝江红菜苔) 283 0.000769 (大白菜, 泡泡椒(精品)) 284 0.000218 (牛首油菜, 大白菜) 285 0.000249 (大白菜, 紫茄子(2)) 286 0.000203 (芜湖青椒(1), 大白菜) 287 0.000157 (大白菜, 茼蒿) 288 0.000207 (大白菜, 菠菜) 289 0.000174 (螺丝椒, 大白菜) 290 0.000405 (大白菜, 西兰花) 291 0.000525 (西峡香菇(1), 大白菜) 292 0.000298 (大白菜, 金针菇(1)) 293 0.000379 (大白菜, 青梗散花) 294 0.000128 (大白菜, 青线椒) 295 0.000270 (大龙茄子, 泡泡椒(精品)) 296 0.000113 (螺丝椒, 大龙茄子) 297 0.000105 (西兰花, 大龙茄子) 298 0.000105 (西峡香菇(1), 大龙茄子) 299 0.000227 (泡泡椒(精品), 奶白菜) 300 0.000103 (牛首油菜, 奶白菜) 301 0.000128 (竹叶菜, 奶白菜) 302 0.000188 (紫茄子(2), 奶白菜) 303 0.000395 (芜湖青椒(1), 奶白菜) 304 0.000102 (苋菜, 奶白菜) 305 0.000148 (菠菜, 奶白菜) 306 0.000144 (螺丝椒, 奶白菜) 307 0.000344 (西兰花, 奶白菜) 308 0.000296 (西峡香菇(1), 奶白菜) 309 0.000120 (奶白菜, 青梗散花) 310 0.000111 (西峡香菇(1), 姬菇(1)) 311 0.000166 (泡泡椒(精品), 娃娃菜) 312 0.000181 (西兰花, 娃娃菜) 313 0.000170 (西峡香菇(1), 娃娃菜) 314 0.000220 (金针菇(1), 娃娃菜) 315 0.000102 (娃娃菜, 青梗散花) 316 0.000159 (小白菜, 泡泡椒(精品)) 317 0.000126 (螺丝椒, 小白菜) 318 0.000172 (西兰花, 小白菜) 319 0.000120 (西峡香菇(1), 小白菜) 320 0.000172 (小米椒, 杏鲍菇(1)) 321 0.000575 (小米椒, 泡泡椒(精品)) 322 0.000155 (小米椒, 竹叶菜) 323 0.000249 (小米椒, 紫茄子(2)) 324 0.000207 (小米椒, 红杭椒) 325 0.000329 (小米椒, 红椒(1)) 326 0.000142 (小米椒, 红薯尖) 327 0.000373 (芜湖青椒(1), 小米椒) 328 0.000190 (菜心, 小米椒) 329 0.000107 (菠菜, 小米椒) 330 0.000455 (螺丝椒, 小米椒) 331 0.000399 (西兰花, 小米椒) 332 0.000506 (西峡香菇(1), 小米椒) 333 0.000273 (金针菇(1), 小米椒) 334 0.000181 (小米椒, 青梗散花) 335 0.000931 (小米椒, 青线椒) 336 0.000131 (小米

椒, 黄白菜(2)) 337 0.000231 (芜湖青椒(1), 小青菜(1)) 338 0.000185 (西兰花, 小青菜(1)) 339 0.000148 (西峡香菇(1), 小青菜(1)) 340 0.000122 (枝江红菜苔, 平菇) 341 0.000351 (平菇, 泡泡椒(精品)) 342 0.000251 (牛首油菜, 平菇) 343 0.000116 (平菇, 紫茄子(2)) 344 0.000164 (芜湖青椒(1), 平菇) 345 0.000349 (平菇, 茼蒿) 346 0.000283 (平菇, 菠菜) 347 0.000312 (西兰花, 平菇) 348 0.000360 (西峡香菇(1), 平菇) 349 0.000261 (金针菇(1), 平菇) 350 0.000168 (平菇, 青梗散花) 351 0.000129 (西兰花, 木耳菜) 352 0.000634 (泡泡椒(精品), 杏鲍菇(1)) 353 0.000115 (竹叶菜, 杏鲍菇(1)) 354 0.000222 (紫茄子(2), 杏鲍菇(1)) 355 0.000172 (红椒(1), 杏鲍菇(1)) 356 0.000172 (红薯尖, 杏鲍菇(1)) 357 0.000194 (芜湖青椒(1), 杏鲍菇(1)) 358 0.000133 (菜心, 杏鲍菇(1)) 359 0.000209 (螺丝椒, 杏鲍菇(1)) 360 0.000466 (西兰花, 杏鲍菇(1)) 361 0.000353 (西峡香菇(1), 杏鲍菇(1)) 362 0.000135 (金针菇(1), 杏鲍菇(1)) 363 0.000277 (杏鲍菇(1), 青梗散花) 364 0.000115 (青线椒, 杏鲍菇(1)) 365 0.000168 (杏鲍菇(1), 黄白菜(2)) 366 0.000273 (枝江红菜苔, 泡泡椒(精品)) 367 0.000163 (牛首油菜, 枝江红菜苔) 368 0.000102 (枝江红菜苔, 紫茄子(2)) 369 0.000164 (芜湖青椒(1), 枝江红菜苔) 370 0.000128 (枝江红菜苔, 茼蒿) 371 0.000128 (枝江红菜苔, 菠菜) 372 0.000261 (枝江红菜苔, 西兰花) 373 0.000187 (西峡香菇(1), 枝江红菜苔) 374 0.000508 (牛首油菜, 泡泡椒(精品)) 375 0.000314 (甜白菜, 泡泡椒(精品)) 376 0.000331 (竹叶菜, 泡泡椒(精品)) 377 0.000822 (紫茄子(2), 泡泡椒(精品)) 378 0.000224 (泡泡椒(精品), 红杭椒) 379 0.000804 (红椒(1), 泡泡椒(精品)) 380 0.000479 (泡泡椒(精品), 红薯尖) 381 0.000266 (茼蒿, 泡泡椒(精品)) 382 0.000495 (菜心, 泡泡椒(精品)) 383 0.000327 (菠菜, 泡泡椒(精品)) 384 0.000188 (螺丝椒, 泡泡椒(精品)) 385 0.001451 (西兰花, 泡泡椒(精品)) 386 0.000299 (西峡花菇(1), 泡泡椒(精品)) 387 0.001478 (西峡香菇(1), 泡泡椒(精品)) 388 0.000469 (金针菇(1), 泡泡椒(精品)) 389 0.000109 (金针菇(袋)(1), 泡泡椒(精品)) 390 0.001022 (泡泡椒(精品), 青梗散花) 391 0.000434 (青线椒, 泡泡椒(精品)) 392 0.000322 (青茄子(1), 泡泡椒(精品)) 393 0.000220 (黄心菜(1), 泡泡椒(精品)) 394 0.000602 (泡泡椒(精品), 黄白菜(2)) 395 0.000115 (西兰花, 海鲜菇(1)) 396 0.000124 (西峡香菇(1), 海鲜菇(1)) 397 0.000139 (牛首油菜, 芜湖青椒(1)) 398 0.000286 (牛首油菜, 茼蒿) 399 0.000270 (牛首油菜, 菠菜) 400 0.000440 (牛首油菜, 西兰花) 401 0.000508 (牛首油菜, 西峡香菇(1)) 402 0.000220 (牛首油菜, 金针菇(1)) 403 0.000150 (牛首油菜, 青梗散花) 404 0.000102 (牛首油菜, 青线椒) 405 0.000116 (甜白菜, 紫茄子(2)) 406 0.000135 (螺丝椒, 甜白菜) 407 0.000196 (西兰花, 甜白菜) 408 0.000238 (西峡香菇(1), 甜白菜) 409 0.000111 (甜白菜, 青梗散花) 410 0.000237 (竹叶菜, 紫茄子(2)) 411 0.000285 (竹叶菜, 红薯尖) 412 0.000516 (芜湖青椒(1), 竹叶菜) 413 0.000179 (竹叶菜, 莴苣) 414 0.000133 (菜心, 竹叶菜) 415 0.000353 (螺丝椒, 竹叶菜) 416 0.000323 (西兰花, 竹叶菜) 417 0.000244 (西峡香菇(1), 竹叶菜) 418 0.000190 (竹叶菜, 青梗散花) 419 0.000133 (竹叶菜, 青线椒) 420 0.000135 (竹叶菜, 黄白菜(2)) 421 0.000329 (紫茄子(2), 红椒(1)) 422 0.000231 (紫茄子(2), 红薯尖) 423 0.001098 (芜湖青椒(1), 紫茄子(2)) 424 0.000116 (莴苣, 紫茄子(2)) 425

0.000205 (菜心, 紫茄子(2)) 426 0.000218 (菠菜, 紫茄子(2)) 427 0.000519 (螺丝椒, 紫茄子(2)) 428 0.000824 (西兰花, 紫茄子(2)) 429 0.000159 (西峡花菇(1), 紫茄子(2)) 430 0.000582 (西峡香菇(1), 紫茄子(2)) 431 0.000205 (金针菇(1), 紫茄子(2)) 432 0.000349 (紫茄子(2), 青梗散花) 433 0.000200 (紫茄子(2), 青线椒) 434 0.000107 (黄心菜(1), 紫茄子(2)) 435 0.000220 (紫茄子(2), 黄白菜(2)) 436 0.000148 (芜湖青椒(1), 红尖椒) 437 0.000137 (螺丝椒, 红尖椒) 438 0.000124 (红椒(1), 红杭椒) 439 0.000168 (芜湖青椒(1), 红杭椒) 440 0.000176 (螺丝椒, 红杭椒) 441 0.000131 (西兰花, 红杭椒) 442 0.000177 (西峡香菇(1), 红杭椒) 443 0.000697 (青线椒, 红杭椒) 444 0.000118 (红椒(1), 红薯尖) 445 0.000963 (芜湖青椒(1), 红椒(1)) 446 0.000118 (菜心, 红椒(1)) 447 0.000137 (菠菜, 红椒(1)) 448 0.000436 (螺丝椒, 红椒(1)) 449 0.000527 (西兰花, 红椒(1)) 450 0.000139 (西峡花菇(1), 红椒(1)) 451 0.000466 (西峡香菇(1), 红椒(1)) 452 0.000155 (金针菇(1), 红椒(1)) 453 0.000397 (青尖椒, 红椒(1)) 454 0.000155 (红椒(1), 青梗散花) 455 0.000296 (红椒(1), 青线椒) 456 0.000118 (红椒(1), 黄白菜(2)) 457 0.000449 (芜湖青椒(1), 红薯尖) 458 0.000139 (苋菜, 红薯尖) 459 0.000122 (菜心, 红薯尖) 460 0.000275 (螺丝椒, 红薯尖) 461 0.000429 (西兰花, 红薯尖) 462 0.000281 (西峡香菇(1), 红薯尖) 463 0.000111 (金针菇(1), 红薯尖) 464 0.000218 (红薯尖, 青梗散花) 465 0.000146 (青线椒, 红薯尖) 466 0.000107 (青茄子(1), 红薯尖) 467 0.000135 (红薯尖, 黄白菜(2)) 468 0.000327 (芜湖青椒(1), 苋菜) 469 0.000177 (芜湖青椒(1), 茼蒿) 470 0.000200 (菜心, 芜湖青椒(1)) 471 0.000342 (芜湖青椒(1), 菠菜) 472 0.000255 (芜湖青椒(1), 螺丝椒) 473 0.001295 (芜湖青椒(1), 西兰花) 474 0.000392 (芜湖青椒(1), 西峡花菇(1)) 475 0.001221 (西峡香菇(1), 芜湖青椒(1)) 476 0.000159 (芜湖青椒(1), 金针菇(1)) 477 0.000281 (芜湖青椒(1), 长线茄) 478 0.000303 (芜湖青椒(1), 青梗散花) 479 0.000249 (芜湖青椒(1), 青线椒) 480 0.000170 (芜湖青椒(1), 青茄子(1)) 481 0.000299 (芜湖青椒(1), 黄白菜(2)) 482 0.000168 (螺丝椒, 苋菜) 483 0.000255 (西兰花, 苋菜) 484 0.000124 (西峡香菇(1), 苋菜) 485 0.000103 (苋菜, 青梗散花) 486 0.000453 (菠菜, 茼蒿) 487 0.000109 (螺丝椒, 茼蒿) 488 0.000336 (西兰花, 茼蒿) 489 0.000362 (西峡香菇(1), 茼蒿) 490 0.000309 (金针菇(1), 茼蒿) 491 0.000113 (金针菇(袋)(1), 茼蒿) 492 0.000137 (菜心, 菠菜) 493 0.000218 (菜心, 螺丝椒) 494 0.000453 (菜心, 西兰花) 495 0.000368 (菜心, 西峡香菇(1)) 496 0.000194 (菜心, 青梗散花) 497 0.000148 (菜心, 青线椒) 498 0.000155 (菜心, 黄白菜(2)) 499 0.000201 (螺丝椒, 菠菜) 500 0.000662 (西兰花, 菠菜) 501 0.000554 (西峡香菇(1), 菠菜) 502 0.000261 (金针菇(1), 菠菜) 503 0.000133 (菠菜, 青梗散花) 504 0.000118 (菠菜, 青线椒) 505 0.000109 (菠菜, 黄白菜(2)) 506 0.000625 (螺丝椒, 西兰花) 507 0.000133 (螺丝椒, 西峡花菇(1)) 508 0.000604 (西峡香菇(1), 螺丝椒) 509 0.000146 (螺丝椒, 金针菇(1)) 510 0.000118 (螺丝椒, 长线茄) 511 0.000253 (螺丝椒, 青梗散花) 512 0.000259 (螺丝椒, 青线椒) 513 0.000164 (螺丝椒, 青茄子(1)) 514 0.000237 (螺丝椒, 黄白菜(2)) 515 0.000338 (西兰花, 西峡花菇(1)) 516 0.001951 (西峡香菇(1), 西兰花) 517 0.000484 (西兰花, 金针菇(1)) 518 0.000103 (西兰花, 金针菇(袋)(1)) 519

0.000153 (西兰花, 长线茄) 520 0.000466 (西兰花, 青梗散花) 521 0.000340 (西兰花, 青线椒)  
522 0.000170 (西兰花, 青茄子(1)) 523 0.000179 (黄心菜(1), 西兰花) 524 0.000355 (西兰花, 黄  
白菜(2)) 525 0.000111 (西峡花菇(1), 金针菇(1)) 526 0.000116 (西峡花菇(1), 青梗散花) 527  
0.000704 (西峡香菇(1), 金针菇(1)) 528 0.000124 (西峡香菇(1), 金针菇(袋)(1)) 529 0.000150  
(西峡香菇(1), 金针菇(袋)(2)) 530 0.000124 (西峡香菇(1), 金针菇(袋)(3)) 531 0.000142 (西峡  
香菇(1), 青尖椒) 532 0.000582 (西峡香菇(1), 青梗散花) 533 0.000327 (西峡香菇(1), 青线椒)  
534 0.000157 (西峡香菇(1), 青茄子(1)) 535 0.000200 (黄心菜(1), 西峡香菇(1)) 536 0.000423  
(西峡香菇(1), 黄白菜(2)) 537 0.000329 (金针菇(1), 青梗散花) 538 0.000144 (金针菇(1), 青线  
椒) 539 0.000185 (黄心菜(1), 金针菇(1)) 540 0.000133 (金针菇(1), 黄白菜(2)) 541 0.000176  
(青线椒, 青梗散花) 542 0.000144 (青茄子(1), 青梗散花) 543 0.000155 (黄心菜(1), 青梗散花)  
544 0.000240 (青梗散花, 黄白菜(2)) 545 0.000128 (青线椒, 黄白菜(2)) 546 0.000113 (云南生  
菜, 西兰花, 泡泡椒(精品)) 547 0.000124 (西峡香菇(1), 西兰花, 云南生菜) 548 0.000116 (西  
峡香菇(1), 泡泡椒(精品), 净藕(1)) 549 0.000129 (西峡香菇(1), 西兰花, 净藕(1)) 550 0.000118  
(西峡香菇(1), 西兰花, 泡泡椒(精品)]) Traceback (most recent call last):

0 (奶白菜) (东门口小白菜) ... 1.000134 0.017430 1 (东门口小白菜) (奶白菜) ... 1.000354  
0.017214 2 (大龙茄子) (泡泡椒(精品)) ... 1.014898 0.231746 3 (泡泡椒(精品)) (大龙茄子) ...  
1.001342 0.242007 4 (金针菇(1)) (娃娃菜) ... 1.004360 0.399116 5 (娃娃菜) (金针菇(1)) ...  
1.013250 0.393798 6 (小米椒) (红杭椒) ... 1.000752 0.088015 7 (红杭椒) (小米椒) ... 1.002302  
0.086603 8 (小米椒) (青线椒) ... 1.020929 0.527669 9 (青线椒) (小米椒) ... 1.026648 0.525023  
10 (平菇) (牛首油菜) ... 1.005161 0.258086 11 (牛首油菜) (平菇) ... 1.004404 0.258645 12 (茼  
蒿) (平菇) ... 1.013330 0.513378 13 (平菇) (茼蒿) ... 1.014366 0.512881 14 (金针菇(1)) (平菇) ...  
1.000407 0.031359 15 (平菇) (金针菇(1)) ... 1.000646 0.031128 16 (枝江红菜苔) (牛首油菜) ...  
1.000341 0.022411 17 (牛首油菜) (枝江红菜苔) ... 1.000246 0.022503 18 (茼蒿) (牛首油菜) ...  
1.006375 0.300824 19 (牛首油菜) (茼蒿) ... 1.005859 0.301183 20 (红尖椒) (螺丝椒) ... 1.002671  
0.064913 21 (螺丝椒) (红尖椒) ... 1.000239 0.067181 22 (青线椒) (红杭椒) ... 1.029966 0.799584  
23 (红杭椒) (青线椒) ... 1.075539 0.790716 24 (红椒(1)) (青尖椒) ... 1.012185 0.661222 25 (青  
尖椒) (红椒(1)) ... 1.041950 0.651260 26 (茼蒿) (菠菜) ... 1.010180 0.300095 27 (菠菜) (茼  
蒿) ... 1.005839 0.303103 28 (金针菇(1)) (茼蒿) ... 1.001882 0.122199 29 (茼蒿) (金针菇(1)) ...  
1.002777 0.121415 30 (金针菇(袋)(1)) (茼蒿) ... 1.019137 0.578326 31 (茼蒿) (金针菇(袋)(1)) ...  
1.004810 0.584254 32 (金针菇(1)) (黄心菜(1)) ... 1.001061 0.115803 33 (黄心菜(1)) (金  
针菇(1)) ... 1.002613 0.114433 34 (西兰花, 泡泡椒(精品)) (云南生菜) ... 1.007107 0.084469 35  
(西兰花, 云南生菜) (泡泡椒(精品)) ... 1.018314 0.269181 36 (泡泡椒(精品), 云南生菜) (西  
兰花) ... 1.018768 0.178045 37 (西兰花) (泡泡椒(精品), 云南生菜) ... 1.000256 0.192981 38 (泡  
泡椒(精品)) (西兰花, 云南生菜) ... 1.000651 0.281858 39 (云南生菜) (西兰花, 泡泡椒(精品))

... 1.000134 0.090808 40 (云南生菜, 西峡香菇(1)) (西兰花) ... 1.017387 0.167339 41 (西兰花)  
(云南生菜, 西峡香菇(1)) ... 1.000264 0.181352 42 (泡泡椒(精品), 西峡香菇(1)) (净藕(1)) ...  
1.026625 0.311928 43 (泡泡椒(精品), 净藕(1)) (西峡香菇(1)) ... 1.038075 0.323691 44 (西峡香  
菇(1), 净藕(1)) (泡泡椒(精品)) ... 1.027578 0.354709 45 (泡泡椒(精品)) (西峡香菇(1), 净藕(1))  
... 1.000886 0.371471 46 (西峡香菇(1)) (泡泡椒(精品), 净藕(1)) ... 1.000529 0.348153 47 (净  
藕(1)) (泡泡椒(精品), 西峡香菇(1)) ... 1.000670 0.329324 48 (西兰花, 西峡香菇(1)) (净藕(1))  
... 1.012923 0.182379 49 (西兰花, 净藕(1)) (西峡香菇(1)) ... 1.033906 0.299726 50 (西峡香  
菇(1), 净藕(1)) (西兰花) ... 1.001946 0.022334 51 (西兰花) (西峡香菇(1), 净藕(1)) ... 1.000037  
0.024197 52 (西峡香菇(1)) (西兰花, 净藕(1)) ... 1.000544 0.322322 53 (净藕(1)) (西兰花, 西峡  
香菇(1)) ... 1.000435 0.192458 54 (泡泡椒(精品), 西兰花) (西峡香菇(1)) ... 1.011150 0.125796  
55 (泡泡椒(精品), 西峡香菇(1)) (西兰花) ... 1.001646 0.018962 56 (西兰花, 西峡香菇(1)) (泡泡  
椒(精品)) ... 1.014844 0.230525 57 (泡泡椒(精品)) (西兰花, 西峡香菇(1)) ... 1.000585 0.241337  
58 (西兰花) (泡泡椒(精品), 西峡香菇(1)) ... 1.000029 0.020546 59 (西峡香菇(1)) (泡泡椒(精  
品), 西兰花) ... 1.000209 0.135255