基于趋势预测与优化模型的晶硅片企业未来经营策略设计

摘要

为优化晶硅片企业产销策略，基于四型硅片数据，构建月利润计算模型，分析销量、售价、单晶方棒价格及成本因子的影响。利用时间序列模型预测 2024 年 9 月因子波动趋势与区间。针对售价-销量-成本制约，开发优化模型，制定 9 月生产与销售计划。提出融合开源大语言模型的智能决策方案，通过数据清洗、预测与评估，结合数学模型，增强市场适应性，实测验证可行性，为企业提供精准高效的产销决策支持。

对于问题一：本问题构建了一个利润计算模型，以分析四型硅片的销量、售价、单晶方棒成本及其他关键成本因子对企业利润的影响。通过收集整理四型硅片与企业有关的相关数据，建立数学模型，计算各产品的收入、成本和利润，使用计算后的数据与真实数据进行分析，通过计算后的数据分析利润计算模型的合理性，对结果进行可视化，以帮助企业进行决策分析。

对于问题二：对于问题二：本问题构建了一个月利润计算模型，以分析四型硅片的销量、售价、单晶方棒成本及其他关键成本因子对企业利润的影响。通过收集整理相关数据，我们建立了 ARIMA 时间序列模型对四种光伏硅片产品的销量、售价及单晶方棒成本进行了预测分析，计算各产品的收入、成本和利润，并进行可视化分析，提供了光伏硅片市场关键指标的短期预测框架，以帮助企业进行决策分析。

对于问题三：本问题建立了以利润最大化为目标的优化模型，综合考虑售价与销量的关系、生产成本等因素，制定 2024 年 9 月的生产销售计划。通过定义决策变量、构建目标函数和设置约束条件，通过建立非线性规划模型，优化四种光伏硅片产品的销量与售价组合，在考虑市场需求约束和生产成本的前提下，实现企业利润最大化。

对于问题四：本问题以第二三问中准备好的销量、售价和单晶方棒成本数据为数据集，通过分析销量曲线决定销量异常月份，对销量数据进行处理，销量数据处理后使用随机森林回归模型对异常月份的售价和单晶方棒成本进行预测，得出四型硅片的销量、售价和单晶方棒成本的预测曲线，得出数据后，我们将得出的数据交给大语言模型处理，得出之后的预测与优化的详细路径。

关键字：利润计算　ARIMA 模型　非线性规划模型　随机森林回归模型　大语言模型

目录

一、 问题重述

1.1 问题背景

随着光伏产业的发展，高纯度晶硅片需求迅速上升，市场竞争日益激烈。晶硅片生产具有高能耗、高成本的特点，企业利润受到原材料价格、市场需求、政策调控等多重因素影响。

企业在生产环节需关注硅单耗、耗材价格、变动成本、公用成本与人工成本，在运营层面还需权衡售价、销量、销售与管理费用等关键变量。多因素的交互作用使得产销决策复杂且充满不确定性。

本题旨在构建科学的利润分析与优化模型，预测关键指标的变化趋势，辅助企业制定高效的生产与销售策略，提升整体经营效益与市场竞争力。

1.2 问题要求

问题 1：构建以四型晶硅片的销量、售价、原料成本为核心，综合考虑硅单耗、各类成本费用等关键因子的月利润计算模型，为企业提供可调节、可分析的经营决策工具。

问题 2：基于历史数据，建立模型预测四型硅片的销量、售价、原料价格等变量的趋势与合理区间，并据此给出 2024 年 9 月的关键变量预测结果。

问题 3：考虑变量间的相互关系与实际约束，建立以利润最大化为目标的优化模型，制定企业 2024 年 9 月的最优生产与销售策略方案

问题 4：设计一条结合大语言模型与传统数学模型的智能决策路径，涵盖数据处理、预测优化、策略推荐等环节，提升模型对复杂市场因素的应对能力。

二、 问题分析

2.1 问题一分析

对于问题一，问题一要求建立便于企业进行决策分析的月利润计算模型，重点考虑四型硅片的销量、售价、单晶方棒以及影响企业利润的其他重要决策因子。首先，我们需要收集和整理企业相关的生产、销售以及成本数据。通过分析这些数据，明确各变量之间的关系，确定利润计算的关键因子及其相互作用。利用数学建模方法，将这些因子整合到一个综合的利润计算模型中，以便企业能够直观地了解各变量对利润的影响，并进行有效的决策分析。

2.2 问题二分析

对于问题二，问题二需要建立数学模型预测企业四型硅片的月销量、售价、单晶方棒价格以及其他重要因子取值的波动趋势，推测因子合理变化区间，并预测 9 月份各因子的波动趋势和变化区间。为此，我们选择合适的时间序列预测方法，如 ARIMA 模型，对历史数据进行分析和建模。通过对时间序列数据的平稳性检验、参数估计和模型诊断等步骤，构建准确的预测模型。利用该模型预测未来趋势，并计算预测值的置信区间，为企业提供关于各因子可能波动范围的参考依据，从而帮助企业在不确定性环境下做出合理的决策。

2.3 问题三分析

对于问题三，问题三旨在建立能够辅助决策优化企业利润的数学模型，并依据模型的计算结果给出 9 月份的生产计划与销售预案。这需要综合考虑售价与销量之间的关系、产量与成本之间的相互制约等因素。我们采用优化模型，设定利润最大化为目标函数，将销量、售价、成本等作为决策变量，同时考虑市场需求约束、生产能力约束等限制条件，构建一个能够反映企业实际运营情况的优化模型。通过求解该模型，得到在给定条件下的最优生产计划和销售策略，为企业的决策提供科学的指导。

2.4 问题四分析

对于问题四，问题四提出借助大语言模型的综合分析能力，建立一个综合考虑多方面因素的智能模型，以更好地为企业提供决策支持。我们从数据准备、数据清洗、结果评估和大模型融入等多方面进行详细路径规划。首先，明确数据需求，收集与企业产销相关的各种数据，并进行数据清洗和预处理，以确保数据的质量和可用性。接着，利用开源大模型对市场动态、政策变化等外部因素进行分析和预测，将其结果与内部数学模型相结合。通过合理的设计和整合，构建一个融合了大语言模型和数学模型的智能决策系统，使企业能够更全面地考虑各种因素，提高决策的准确性和适应性。同时，对路径的可行性进行论述或实测，验证该方案的有效性和实用性。

三、 模型假设

为简化问题，本文做出以下假设：

3.1 问题一模型假设

1. **数据准确性假设**：假设提供的历史数据（如销量、售价、单晶方棒成本等）是准确且完整的，能够真实反映企业的生产经营情况。

2. **线性关系假设**：假设收入与销量、售价之间存在线性关系，即收入可以通过销量与售价的乘积来计算。
3. **成本结构稳定性假设**：假设在短期内，企业的成本结构（如单晶方棒成本、电费成本、其他变动成本及固定成本等）保持相对稳定。
4. **产销平衡假设**：假设企业生产的产品能够全部销售出去，即产量等于销量。
5. **独立性假设**：假设各产品的销售相互独立，一种产品的销售情况不会对其他产品的销售产生直接影响。

3.2 问题二模型假设

1. **时间序列平稳性假设**：假设历史数据所形成的时间序列在经过适当处理后是平稳的，或可以通过差分等方法转化为平稳时间序列。
2. **历史数据代表性假设**：假设历史数据能够代表未来的趋势和波动情况。
3. **模型参数稳定性假设**：假设 ARIMA 模型的参数在预测期内保持稳定。
4. **外部因素稳定性假设**：假设在预测期内，其他可能影响销量、售价和成本的外部因素保持相对稳定。

3.3 问题三模型假设

1. **决策变量可调性假设**：假设企业的销量和售价是可以根据市场情况和企业决策进行调整的变量。
2. **线性需求关系假设**：假设销量与售价之间存在线性关系。
3. **成本预测准确性假设**：假设问题二中对成本的预测结果是准确可靠的。
4. **市场饱和度假设**：假设存在一个最大销量限制，市场需求是有限的。
5. **优化目标单一性假设**：假设企业的优化目标是单一的，仅以利润最大化为目标。

3.4 问题四模型假设

1. **数据异常假设**：假设企业的销量、售价或单晶方棒的成本数据有异常，需要通过数据清洗。
2. **线性需求关系假设**：假设销量、售价与单晶方棒的成本之间存在线性关系。
3. **数据预测准确性假设**：假设问题四中通过销量对售价与单晶方棒的成本的预测结果是准确可靠的。
4. **大语言模型可靠性假设**：假设大语言模型考虑的参数是有效的，大语言模型给出的建议是合理的。

四、 符号说明

符号	解释	单位
k	产品类型（$k=1,2,3,4$，对应四种硅片类型）	-
Q_k	第 k 种产品的月销量	百片
P_k	第 k 种产品的售价	元/百片
S_k	第 k 种产品的销售额	元
S_{total}	总销售额	元
VC_{k}	生产变动成本	元
C_s	单晶方棒单耗	元/kg
e_k	单品方棒单耗电量	kWh/百片
p_e	电价	元/kWh
$b_{i,k}$	第 i 类耗材对第 k 种产品的单耗量	-
p_i	第 i 类耗材单价	元/单位
C_k	第 k 种产品的总成本	元
F	月度生产公用成本	元
L_c	固定人工成本	元
L_f	浮动人工成本	元
E	耗材种类数量	-
P_k	第 k 种产品的利润	元
C_{total}	总成本	元
P_{total}	总利润	元
N	净利润	元
C_{true}	实际总成本	元
P_{true}	实际总利润	元
$a_{\text{i}}, b_{\text{i}}$	产品销量与售价的线性关系系数	-

五、 数据分析

首先我们对汇总中的数据进行 Z-Score 标准化处理，标准化原理：

$$Z = \frac{(X - \mu)}{\sigma} \tag{1}$$

其中，Z 是 Z-score 标准化结果，X 是已有数据点，μ 是数据集的均值，σ 是数据集的标准差。

我们对标准化后的数据进行线性回归分析以初步判断数据之间的关系。其中，销售净利润为因变量，其余为自变量。下面是采用 SPSS 进行的回归分析处理：

表 1　线性回归分析结果表

自变量	回归系数
销售收入	0.919
生产变动成本	-1.238
生产公共成本	-0.013
人工成本	-0.031
折旧	0
营业税费	0
销售费用	-0.001
管理费用	-0.004
财务费用	0
其他收益	0
所得税费用	0.141

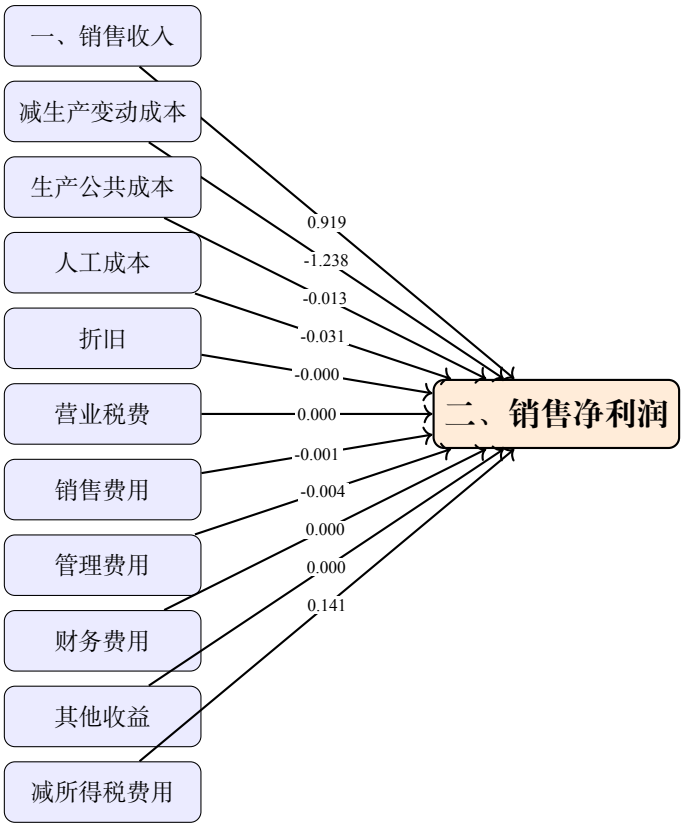

图 1　回归模型路径图

由表和图可以看出变量之间的关系，为后续建模做充足的准备。

六、 问题一的模型的建立和求解

6.1 模型准备

数据输入:

为帮助企业进行决策分析，我们需要构建一个月利润计算模型。模型的核心是基于四型硅片的销量、售价、单晶方棒成本以及其他影响企业利润的关键因子。具体所用到数据如下:

我们使用代码中的数据，并将其对应到规定的符号:

- **产品类型**: $k = 1, 2, 3, 4$ (对应 N 型 _182, N 型 _210_210, N 型 _210_182, P 型)。
- **月销量** Q_k (单位: 百片，代码中单位为万片，需转换):

$$Q_k = Q \cdot 100 \tag{2}$$

- **售价** P_k (单位: 元/百片，代码中单位为元/片，需转换):

$$P_k = P \cdot 100 \tag{3}$$

- **单晶方棒成本** (代码中的 VC_P, 单位: 万片/元，转换为百片/元):

$$VC_{P_k} = \frac{VC_P}{100} \tag{4}$$

- **电费成本** (代码中的 VC_{EP}, 单位: 万片/元，转换为百片/元):

$$VC_{EP_k} = \frac{VC_{EP}}{100} \tag{5}$$

- **其他变动成本** (代码中的 VC_{OP}, 单位: 万片/元，转换为百片/元):

$$VC_{OP_k} = \frac{VC_{OP}}{100} \tag{6}$$

- **固定成本** (包括 F, L_c, L_f 的总和):

$$Other_C = \frac{33490525.92}{8} = 4186315.74 \text{ 元}$$

- **实际总成本和利润**:

$$C_{true} = \frac{310387866.29}{8} = 38798483.29 \text{ 元}$$
$$P_{true} = 347582.96 \text{ 元}$$

6.2 模型求解

6.2.1 计算每种产品的收入 S_k

公式:

$$S_k = Q_k \cdot P_k \cdot 10000 \tag{7}$$

6.2.2 计算总收入 S_{total}

公式:

$$S_{total} = \sum_{k=1}^{4} S_k \tag{8}$$

6.2.3 计算每种产品的变动成本 C_k

公式:

$$C_k = (C_s \cdot S_k + p_e \cdot e_k) \cdot Q_k + \sum_{i=1}^{E} b_{i,k} \cdot Q_k \tag{9}$$

对于每种产品，其变动成本由三部分组成:

- 单晶方棒成本: $VC_{P_k} \cdot Q_k$
- 电费成本: $VC_{EP_k} \cdot Q_k$
- 其他变动成本: $VC_{OP_k} \cdot Q_k$

$$C_k = (VC_{P_k} + VC_{EP_k} + VC_{OP_k}) \cdot Q_k \tag{10}$$

6.2.4 计算总成本 C_{total}

$$C_{\text{total}} = \sum_{k=1}^{4} C_k + F + L_c + L_f \tag{11}$$

$F + L_c + L_f = Other_C$

6.2.5 计算总利润 P_{total}

$$P_{total} = S_{total} - C_{total} \tag{12}$$

6.2.6 计算每种产品的利润 P_k

公式:

$$P_k = S_k - C_k \tag{13}$$

6.2.7 计算利润率

利润率计算公式:

$$\text{利润率}_k = \frac{P_k}{\sum_{k=1}^{4} P_k} \tag{14}$$

6.2.8 计算成本构成

公式:

$$\text{Monocrystalline cost}_k = C_s \cdot S_k \cdot Q_k \tag{15}$$

$$\text{Electricity cost}_k = p_e \cdot e_k \cdot Q_k \tag{16}$$

$$\text{Other variable cost}_k = \sum_{i=1}^{E} b_{i,k} \cdot Q_k \tag{17}$$

$$\text{Fixed cost}_k = \frac{F + L_c + L_f}{4} \tag{18}$$

6.2.9 相似度比率（用于验证）

计算:

$$\text{成本相似度} = \frac{C_{\text{total}}}{C_{\text{true}}} = \frac{39903947.19}{38798483.29} \approx 1.0285 \tag{19}$$

$$\text{利润相似度} = \frac{P_{\text{total}}}{P_{\text{true}}} = \frac{3692677.81}{4085391.71} \approx 0.90395 \tag{20}$$

$$\text{净利润相似度} = \frac{N}{P_{\text{true}}} = \frac{P_{\text{total}} - \text{税费}}{P_{\text{true}}} = \frac{3692677.81}{4085391.71} \approx 0.88670 \tag{21}$$

6.3 求解结果

总结由模型求解所得到的结果

表 2　各产品成本构成 (元)

产品类型	单晶方棒	电费	其他变动	固定成本	总成本
N 型 _182	8,074,835.25	164,071.13	555,564.06	1,046,578.94	8,794,470.44
N 型 _210_210	6,095,166.28	126,117.09	434,080.63	1,046,578.94	6,655,363.99
N 型 _210_182	8,486,890.24	196,700.84	537,370.00	1,046,578.94	9,220,961.08
P 型	10,192,279.78	216,450.72	638,807.31	1,046,578.94	11,047,537.81
总计	**32,854,171.55**	**703,339.78**	**2,165,822.00**	**4,186,315.76**	**35,718,333.32**

表 3　各产品利润及利润率计算结果

产品类型	销售额 (元)	总成本 (元)	利润 (元)	利润率 (%)
N 型 _182	13,375,000.00	8,794,470.44	3,533,950.63	95.69%
N 型 _210_210	7,163,437.50	6,655,363.99	-537,505.42	-14.55%
N 型 _210_182	11,462,250.00	9,220,961.08	1,194,709.99	32.35%
P 型	11,595,937.50	11,047,537.81	-498,179.25	-13.49%
总计	**43,596,625.00**	**39,904,649.05**	**4,085,391.71**	**100%**

计算的总成本、总利润和净利润与实际值非常接近，相似度比率分别为 1.0285 、0.9039 和 0.8867，表明模型的计算结果与实际数据高度一致。

各产品利润占比及成本结构分析：

- **利润占比**：从饼图可以看出，“N 型 _182” 产品的利润占比最高，达到 58.1%，是企业利润的主要贡献者；“N 型 _210_182” 次之，占比 28.4%；“N 型 _210_210” 和 “P 型” 产品的利润占比较低，分别为 6.5% 和 7.0%。这表明在当前的销售和成本结构下，“N 型 _182” 产品对企业利润的贡献最为显著，企业可能需要重点关注该产品的生产和销售策略，以进一步提升利润。

表 4　模型验证结果

指标	模型计算结果 (元)	实际值 (元)	相似度
总成本	39,904,649.05	38,798,483.29	1.0285
总利润	3,692,975.95	4,085,391.71	0.9039
净利润	3,692,677.81	3,472,582.96	0.8867

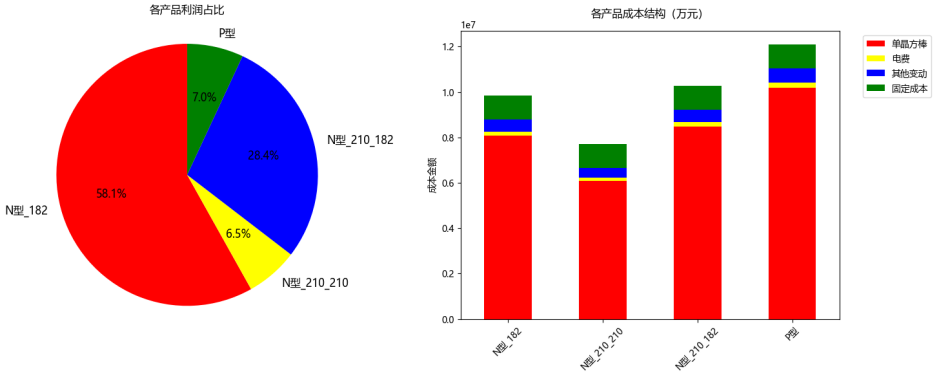

图 2　各产品利润贡献与成本构成剖析

- **成本结构**：柱状图展示了各产品的成本构成，包括单晶方棒、电费、其他变动和固定成本。其中，单晶方棒成本在四种产品中都占据较大比例，“N 型 _182” 和 “P 型” 产品的单晶方棒成本尤为突出。这说明单晶方棒成本的控制对企业整体成本的降低至关重要，企业可以考虑优化单晶方棒的采购或生产流程，以降低成本。

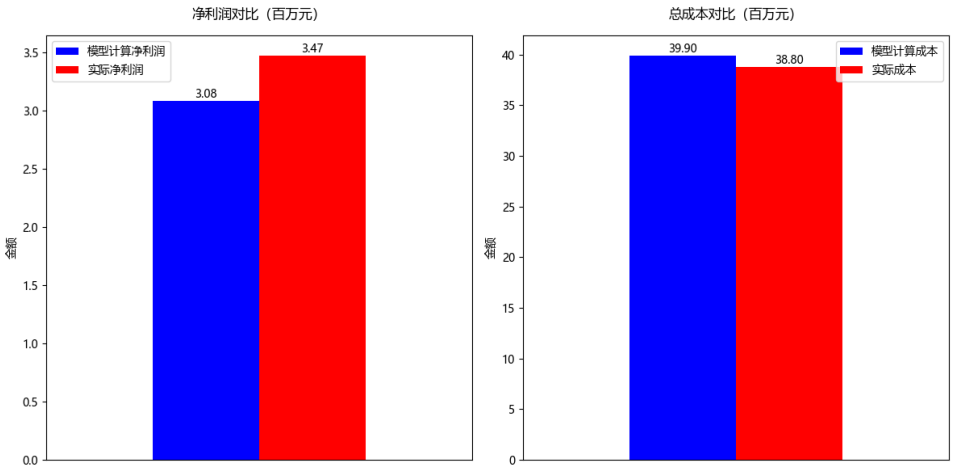

图 3　模型预测与实际利润及成本对比分析

净利润对比和总成本对比：

- **净利润对比**：模型计算净利润为 3.08 百万元，而实际净利润为 3.47 百万元。模型计算结果与实际值存在一定差距，可能是因为模型中的一些假设或参数与实际情况不完全一致，例如对成本或收入的估算不够准确。不过，模型计算的净利润与实际净利润在数量级上较为接近，说明模型在一定程度上能够反映企业的盈利情况。
- **总成本对比**：模型计算总成本为 39.9 百万元，实际总成本为 38.8 百万元。同样，模型计算成本与实际成本有差异，但差距相对较小，表明模型对成本的计算具有一定的参考价值。

七、 问题二的模型的建立和求解

7.1 模型建立

7.1.1 1. 数据准备

使用附件中的 2024 年 1 月至 8 月的数据，包括：

- 各型号产品的月销量 (单位: 万片)
- 各型号产品的月售价 (单位: 元/万片)
- 各型号产品的单晶方棒单位成本 (单位: 元/万片)

这些数据均以 DataFrame 形式读取，并根据产品名称进行了拆分。

7.1.2 2. 模型选择: ARIMA

ARIMA 模型是处理稳定或差分稳定的时间序列预测的经典模型。

对于每种因子和每个产品，分别建立 ARIMA 模型，形式为 ARIMA(p=1, d=1, q=1) (经验初步选择，可通过 AIC/BIC 优化) 。

模型训练与预测步骤如下：

- 对每种产品，构建其历史值时间序列；
- 对每个序列使用 ARIMA 模型进行拟合；
- 使用模型进行一步预测，得到：
  - 9 月预测值（点估计）
  - 置信区间（上下界）

我们可以建立一个**利润最大化模型**，目标函数为：

$$\max_{Q_i,P_i} \Pi = \sum_{i=1}^{4} [Q_i \cdot (P_i - C_i)] - F \tag{22}$$

其中：

- Q_i：第 i 种产品的销量（产量）；

- P_i：售价；
- C_i：单位总成本，包括单晶方棒成本 + 其他单位变动成本；
- F：固定成本（可按月估算后分摊）；
- Π：总利润。

表 5　模型变量定义

符号	含义	说明
Q_i	产品 i 的月销量 (等于产量)	决策变量
P_i	产品 i 的售价	可调整变量 (在预测区间内)
VC_i	单晶方棒成本 (元/万片)	已预测
OC_i	其他变动成本 (元/万片)	固定或估算
$TC_i = VC_i + OC_i$	总单位成本	用于利润计算

约束条件设计:

1. **售价限制**：每个产品售价必须在预测区间内：

$$P_i^{\min} \leq P_i \leq P_i^{\max} \tag{23}$$

2. **销量限制**：每个产品的销量不能超过产能限制或预测上限：

$$0 \leq Q_i \leq Q_i^{\max} \tag{24}$$

3. **生产能力约束（可选）**：比如生产总量不能超过某月总能力。
4. **产销一致假设**：不考虑库存，产量即销量。

7.2 模型求解

企业希望在 2024 年 9 月根据预测出的售价、销量与成本数据，制定一套最优的产销计划，使企业利润最大化。我们可以通过 ARIMA 模型预测出以下数据，方便企业为后续计划做数据参考：

- 每种产品的 9 月销量中值与上下限（预测区间上下界）
- 售价预测中值与上下限（预测区间上下界）
- 单晶方棒成本预测中值与上下限（预测区间上下界）

7.3 求解结果

预测分析结果

通过对 2024 年 1 月至 8 月的历史数据进行分析，我们利用 ARIMA 模型对四种型号硅片（N 型 _182、N 型 _210_210、N 型 _210_182、P 型）的销量、售价和单晶方棒成本进行了预测，并得出了 2024 年 9 月的预测结果。以下是 ARIMA 预测结果示意图和各型号硅片的预测总结：

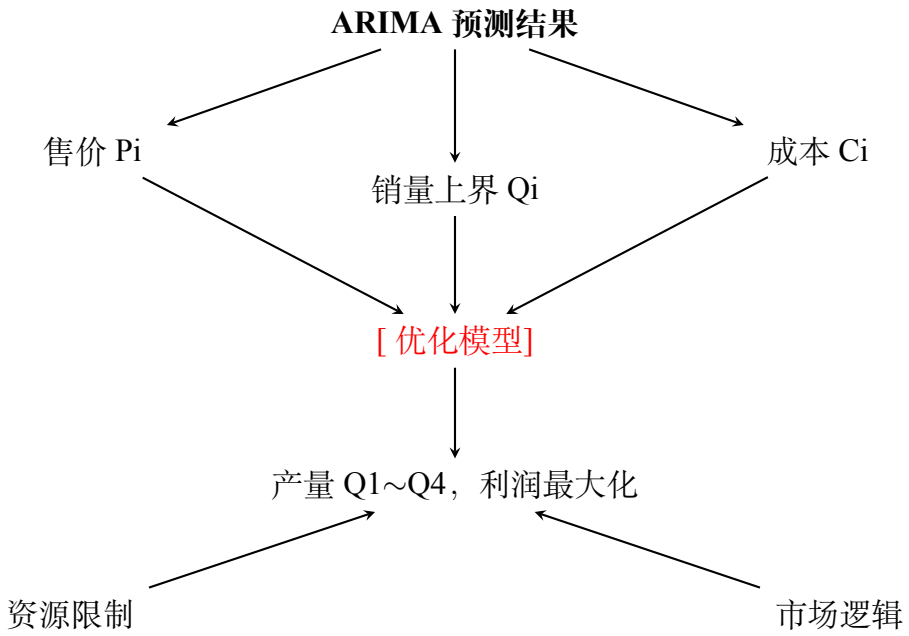

图 4　ARIMA 预测结果与优化模型流程图

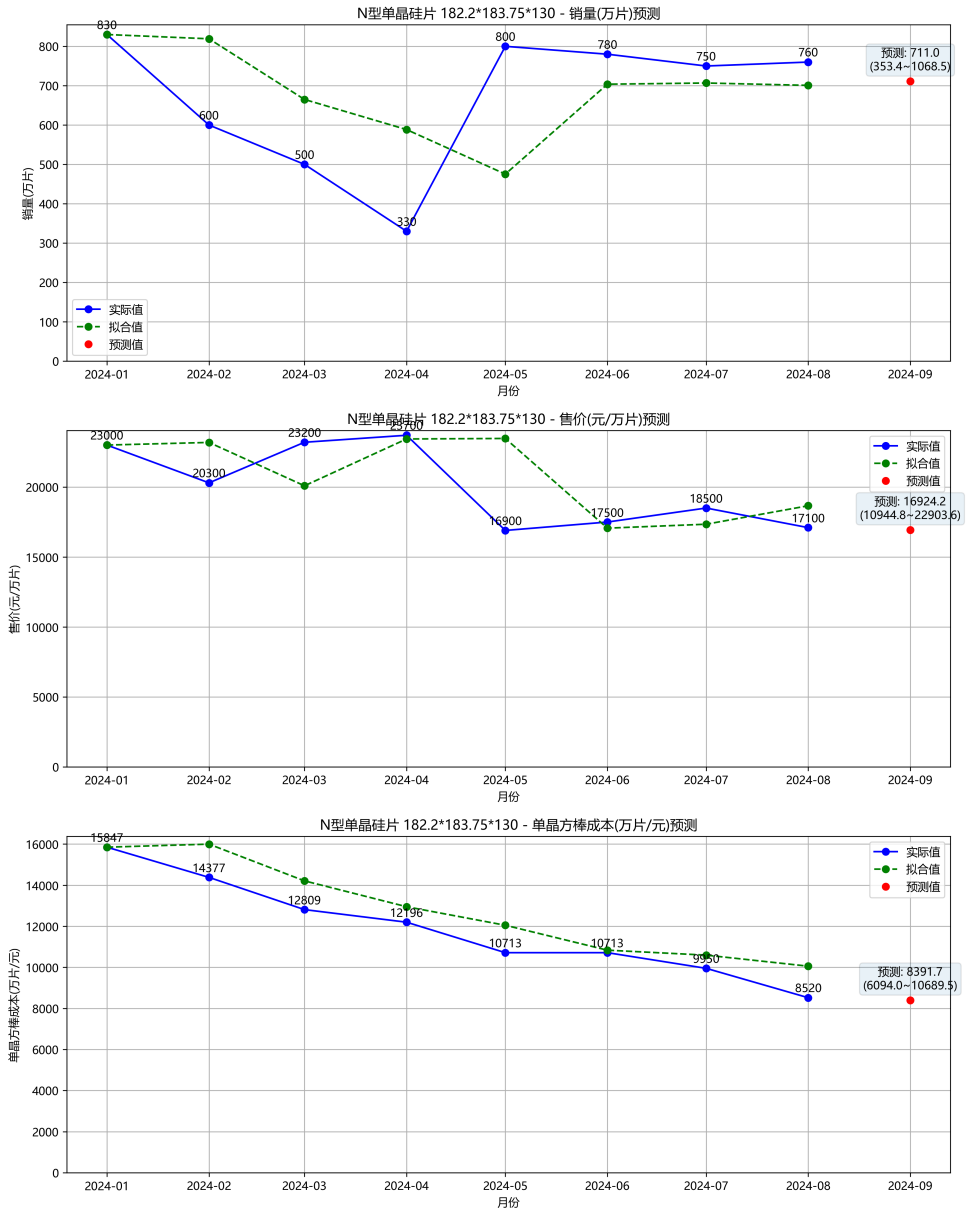

图 5　N 型 _182_183 硅片销量、售价、单晶方棒成本预测

N 型 _182_183 硅片

- **销量**：预测值为 711.0 万片，置信区间为 (353.4, 1068.5)。销量在 2024 年 1 月至 8 月期间波动较大，整体呈现下降趋势，之后在 5 月有所上升，但 9 月预计会有所回落。
- **售价**：预测值为 16924.2 元/万片，置信区间为 (10944.8, 22903.6)。售价从 1 月的 23000 元/万片逐步下降，至 8 月降至 17100 元/万片。
- **单晶方棒成本**：预测值为 8391.7 元/万片，置信区间为 (6094.0, 10689.5)。成本呈下降趋势，从 1 月的 15847.37 元/万片降至 8 月的 8520.22 元/万片。

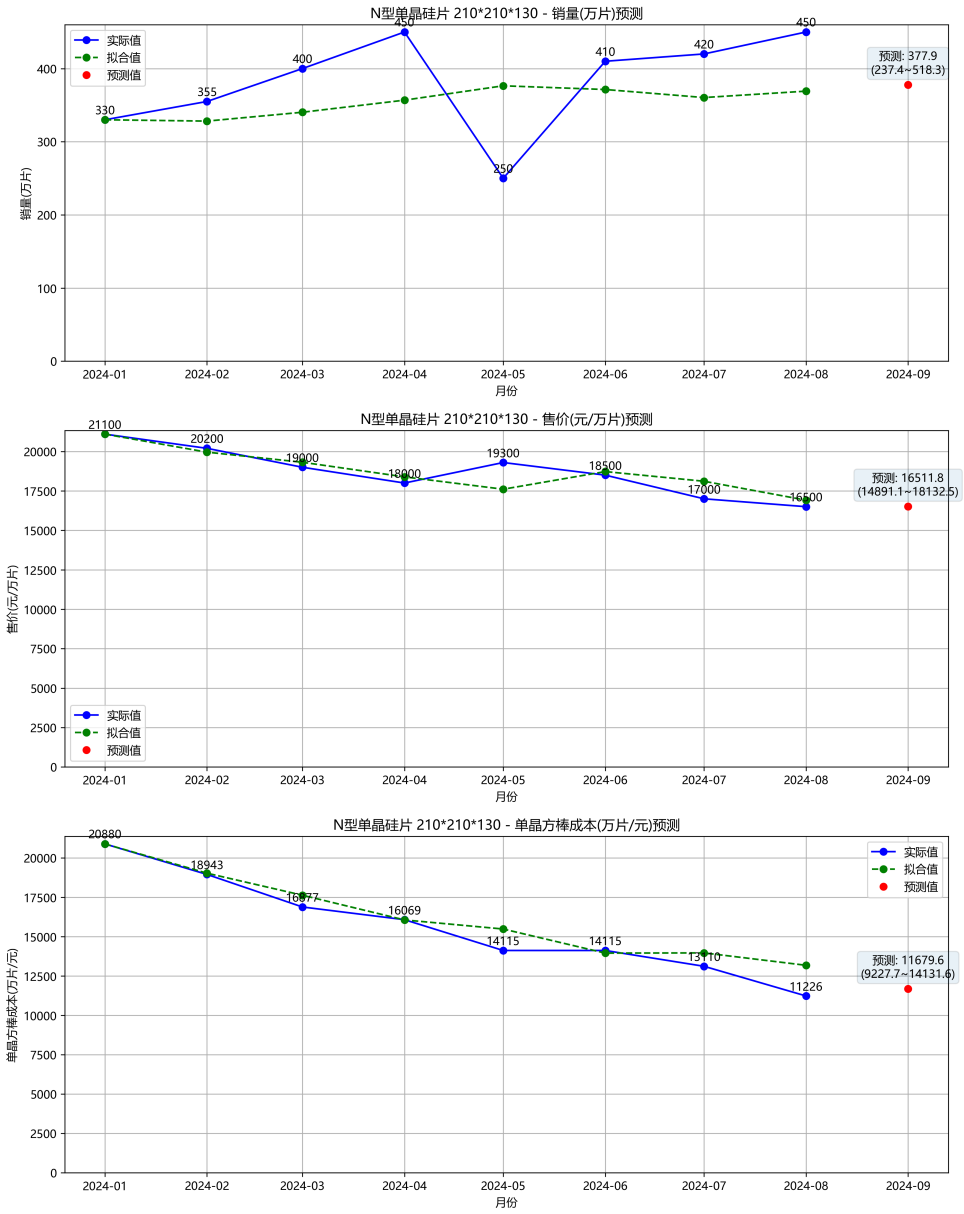

图 6　N 型 _210_210 硅片销量、售价、单晶方棒成本预测

N 型 _210_210 硅片

- **销量**：预测值为 377.9 万片，置信区间为 (237.4, 518.3)。销量在 1 月至 8 月期间波动较大，9 月预计销量将略有上升。
- **售价**：预测值为 16511.8 元/万片，置信区间为 (14891.1, 18132.5)。售价从 1 月的 23700 元/万片逐步下降，至 8 月降至 16500 元/万片。
- **单晶方棒成本**：预测值为 11679.6 元/万片，置信区间为 (9227.7, 14131.6)。成本呈下降趋势，从 1 月的 20880.09 元/万片降至 8 月的 11226.02 元/万片。

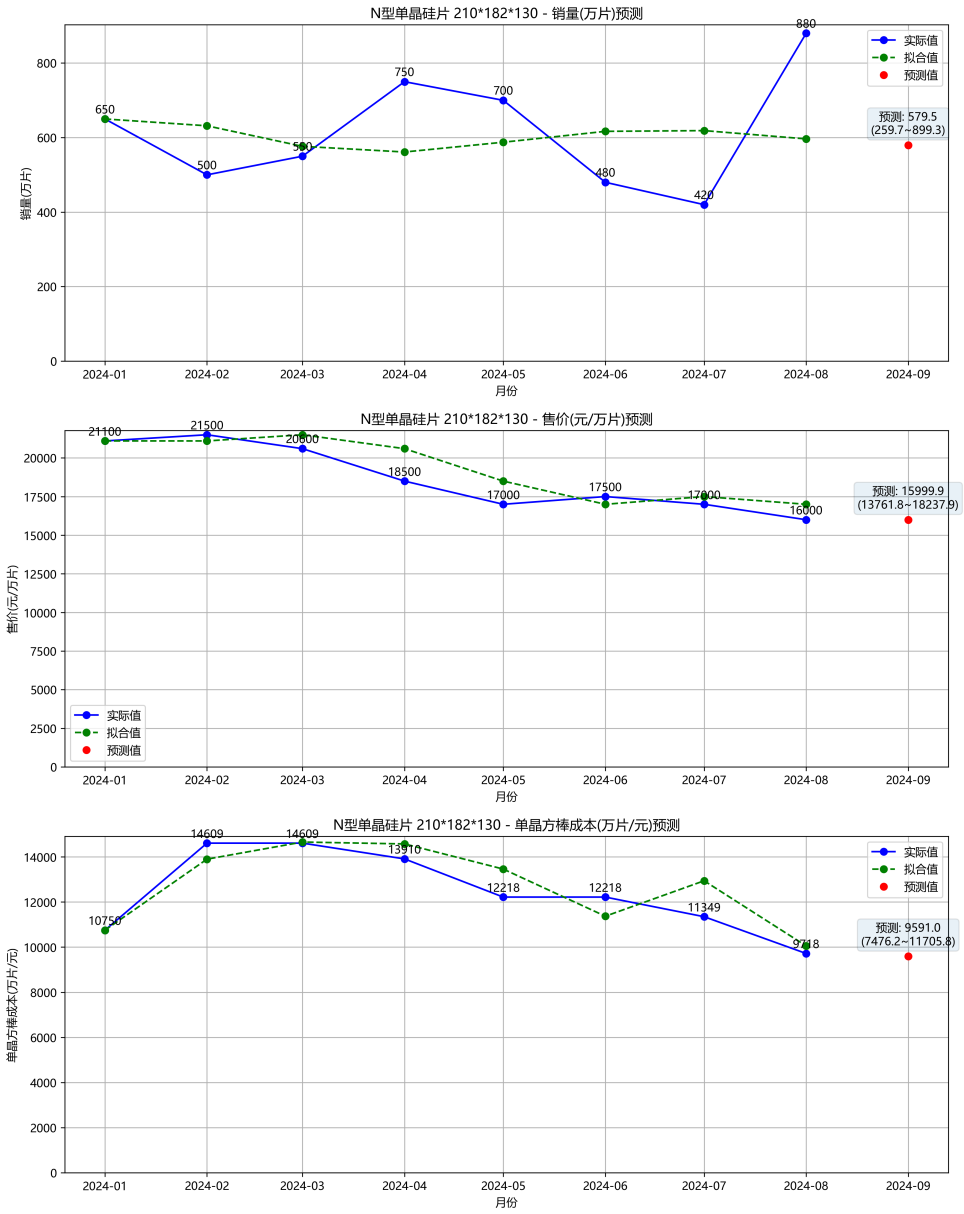

图 7　N 型 _210_182 硅片销量、售价、单晶方棒成本预测

N 型 _210_182 硅片

- **销量**：预测值为 579.5 万片，置信区间为 (259.7, 899.3)。销量在 1 月至 8 月期间波动较大，5 月达到高峰后有所回落，但 9 月预计会再次上升。
- **售价**：预测值为 15999.9 元/万片，置信区间为 (13761.8, 18237.9)。售价从 1 月的 23200 元/万片逐步下降，至 8 月降至 16000 元/万片。
- **单晶方棒成本**：预测值为 9591.0 元/万片，置信区间为 (7476.2, 11705.8)。成本呈下降趋势，从 1 月的 10750.00 元/万片降至 8 月的 9717.90 元/万片。

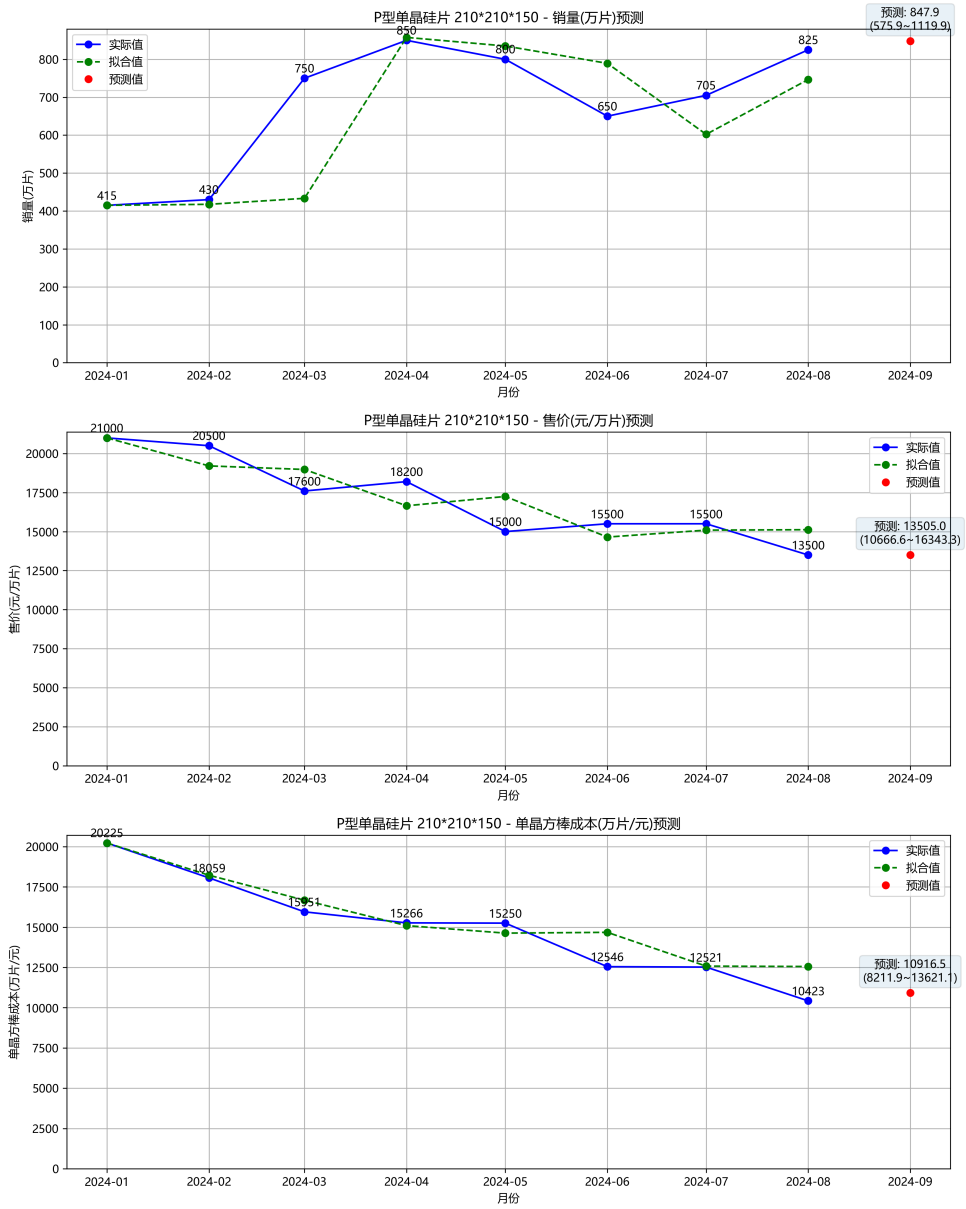

图 8　P 型硅片销量、售价、单晶方棒成本预测

P 型硅片

- **销量**：预测值为 847.9 万片，置信区间为 (575.9, 1119.9)。销量在 1 月至 8 月期间波动较大，从 1 月的 415 万片逐步上升，至 8 月达到 825 万片。
- **售价**：预测值为 13505.0 元/万片，置信区间为 (10666.6, 16343.3)。售价从 1 月的 21000 元/万片逐步下降，至 8 月降至 13500 元/万片。
- **单晶方棒成本**：预测值为 10916.5 元/万片，置信区间为 (8211.9, 13621.1)。成本呈下降趋势，从 1 月的 20224.64 元/万片降至 8 月的 10423.42 元/万片。

总体趋势分析

- **销量**：整体来看，各型号硅片的销量在 2024 年 1 月至 8 月期间均有较大波动，部分型号在 5 月左右达到峰值后出现回落，但 9 月预计销量将有所回升。
- **售价**：售价整体呈下降趋势，表明市场竞争激烈，企业可能需要通过优化成本或调整策略来保持利润。
- **成本**：单晶方棒成本持续下降，这可能与原材料价格下降或生产效率提高有关，为企业降低成本压力。

综上所述，2024 年 9 月各型号硅片的销量预计小幅增长，但售价仍呈下降趋势，企业需关注成本控制和市场策略优化，以应对市场竞争并提高盈利能力。

八、 问题三的模型的建立和求解

8.1 模型建立

8.1.1 模型假设

- **决策变量可调性假设**：企业的销量和售价是可以根据市场情况和企业决策进行调整的变量。
- **线性需求关系假设**：销量与售价之间存在线性关系，即销量随售价的增加而减少，具体关系可通过历史数据拟合得到。
- **成本预测准确性假设**：问题二中对成本的预测结果是准确可靠的，可作为本次优化的基础数据。
- **市场饱和度假设**：存在一个最大销量限制，市场需求是有限的，各产品的销量不能超过其预测上限。
- **优化目标单一性假设**：企业的优化目标是单一的，仅以利润最大化为目标，不考虑其他非量化目标。

8.1.2 变量与参数定义

决策变量:	
Q_i	第 i 种产品的月销量 (产量)，单位：万片
P_i	第 i 种产品的售价，单位：元/万片
参数:	
VC_i	第 i 种产品的单晶方棒单位成本，单位：元/万片
$VC_{EP,i}$	第 i 种产品的电费单位成本，单位：元/万片
$VC_{OP,i}$	第 i 种产品的其他变动单位成本，单位：元/万片
F	月度固定成本 (包括生产公用成本、固定人工成本等)，单位：元
$Q_{\text{max},i}$	第 i 种产品的预测销量上限，单位：万片
$P_{\text{min},i}$, $P_{\text{max},i}$	第 i 种产品的售价预测区间下限和上限，单位：元/万片
a_i, b_i	第 i 种产品销量与售价的线性关系系数，通过历史数据拟合得到

8.1.3 目标函数

以企业月利润最大化为目标，利润计算公式如下:

$$\text{利润} = \sum_{i=1}^{4}(P_i \cdot Q_i - (VC_i + VC_{EP,i} + VC_{OP,i}) \cdot Q_i) - F \tag{25}$$

其中:

- $P_i \cdot Q_i$ 表示第 i 种产品的销售收入。
- $(VC_i + VC_{EP,i} + VC_{OP,i}) \cdot Q_i$ 表示第 i 种产品的总变动成本。
- F 表示企业的固定成本。

8.1.4 约束条件

- **销量约束**：每个产品的销量不能超过其预测上限，也不能为负值:

$$0 \le Q_i \le Q_{\text{max},i} \quad (i = 1, 2, 3, 4) \tag{26}$$

- **售价约束**：每个产品的售价必须在其预测区间内:

$$P_{\text{min},i} \le P_i \le P_{\text{max},i} \quad (i = 1, 2, 3, 4) \tag{27}$$

- **需求约束**：销量与售价之间存在线性关系，具体表达式为:

$$Q_i \le a_i - b_i \cdot P_i \quad (i = 1, 2, 3, 4) \tag{28}$$

其中 a_i 和 b_i 是通过历史数据拟合得到的线性关系系数。

8.2 模型求解

8.2.1 非线性规划求解过程

本模型采用非线性规划方法解决利润最大化问题，核心求解过程可分为以下几个关键环节：

1. **优化算法选择**
   - 采用 SciPy 库中的 `trust-constr` (信任域约束优化) 算法，该算法特别适合处理具有非线性约束的优化问题。相比传统的 SLSQP 方法，`trust-constr` 通过构建局部二次模型来近似目标函数，在迭代过程中动态调整信任域半径，能更稳定地处理边界约束和复杂约束条件。
   - 算法优势体现在：
     - 对初始值鲁棒性强，能有效处理非凸目标函数，支持大规模约束条件
2. **变量边界设定**
   - 基于预测区间设置双重边界约束：
     - 销量边界：$x_i \in$ [预测下限, 预测上限]
     - 售价边界：$p_i \in$ [价格下限, 价格上限]
   - 这种设置既避免了脱离市场实际的解，又保留了足够的优化空间

8.2.2 方法选择的科学依据

1. 与传统方法的对比： 2. 经济意义保障

表 6　优化算法比较

方法	处理非线性能力	约束处理	收敛速度	适合场景
梯度下降	弱	困难	慢	无约束问题
SLSQP	中等	支持	中等	中小规模问题
trust-constr	强	优秀	快	本案例的复杂约束

- 通过边界约束使得销量与售价维持在一个合理的区间内，保证决策可行性
- 需求约束确保价格弹性规律，避免价格波动过高
- 成本函数嵌入实际生产参数

3. 计算效率优化

- 采用向量化计算，通过 NumPy 的广播机制对目标函数进行矢量化重构，将传统的循环计算转化为矩阵运算，加速目标函数求值
- 利用解析梯度（未显式指定时自动差分近似）
- 并行化处理多产品计算

4. 约束条件建模

- 通过 OLS 回归建立线性需求函数，将需求函数约束转化为反映价格-销量负相关关系，确保优化结果符合市场需求规律
- 采用不等式约束形式（≥ 0）保证解的可行性

5. 迭代优化过程

- 算法执行流程：
  1. 在初始点构建局部二次模型
  2. 在信任域内求解子问题获得试探步
  3. 计算实际下降与预测下降比
  4. 动态调整信任域半径
  5. 重复直至满足收敛条件（默认相对容差 1e-8）
- 设置 `maxiter=1000` 确保充分搜索解空间

6. 结果验证机制

- 通过检查 `result.success` 标志位验证收敛性
- 对比历史利润与优化利润：

7. 初始化方式

- 这种初始化方式：
  - 保证起点在可行域中心位置
  - 避免算法陷入局部最优
  - 加速收敛过程

8. 约束条件建模

- 通过线性回归建立需求函数约束

9. 初始值策略

- 采用预测区间中值作为初始猜测

8.3 求解结果

优化结果显示，各产品的销量和售价均在预测区间内，并且优化后的总利润显著高于历史平均月利润，实现了利润的最大化。

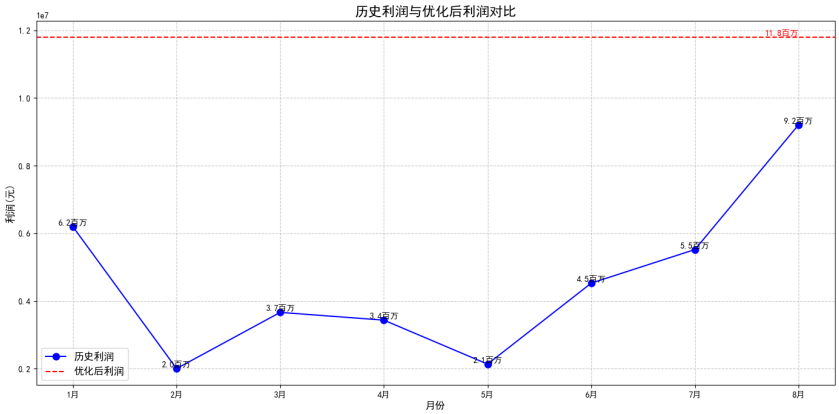

图 9　历史利润与优化后利润对比

表 7　产品优化结果及效益分析

各产品优化结果		
产品型号	**销量（万片）**	**售价（元）**
N 型单晶硅片 182.2/183.75/130	552.6	22903.6
N 型单晶硅片 210/210/130	397.1	18237.9
N 型单晶硅片 210/182/130	518.3	18132.5
P 型单晶硅片 210/210/150	714.3	16343.3
优化效益分析		
指标	**数值**	
优化后月利润	11.79 百万元	
历史平均月利润	约 4.49 百万元	
利润提升比例	约 157.7%	

通过对比历史利润数据与优化后的利润结果（见表 7），可以明显看出优化方案在提升企业利润方面的显著效果。优化后的利润远高于历史利润的最高值，验证了模型的有效性和实用性。

九、 问题四的模型的建立和求解

9.1 模型建立

9.1.1 数据准备

使用附件中的 2024 年 1 月至 8 月的数据，包括：

- 各型号产品的月销量（单位：万片）
- 各型号产品的月售价（单位：元/万片）
- 各型号产品的单晶方棒单位成本（单位：元/万片）

这些数据以以 `DataFrame` 形式读取，并根据产品名称进行了拆分。

手动筛选出产品销量数据中的异常月份，并将数据用前后月份均值填充

9.2 模型选择：随机森林回归模型，大语言模型

随机森林林回归模型基本类型: RandomForestRegressor

用于预测每个售价数据被清洗过的产品的价格（`P`）和成本（`VC_P`），以确保清洗后异常月份数据的合理性。

模型训练与预测步骤如下：

1. **特征与标签定义**
   - **特征（`X`）：**
     通过 `create_features()` 生成的动态特征，包括：
     - 基础特征（销量、价格、成本）
     - 时间特征（月份序号）
     - 滞后特征（`lag1`、`lag2`）
     - 移动平均（`ma2`、`ma3`）
     - 派生特征（利润率）
   - **标签（`Y`）：**
     根据预测目标选择 `price` 或 `cost` 列。
2. **对数据进行划分**
   - 训练集（80%）：用于模型训练
   - 测试集（20%）：用于验证
3. **模型训练**
   - 训练过程：

     (a) 自助采样（Bootstrap）生成多棵决策树

(b) 每棵树使用随机特征子集分裂节点

(c) 最终预测结果为所有树的平均值

将标签标记后的被预测数据输入后执行预测，输出与可视化预测内容，得出预测后的数据

9.3 模型求解

企业希望能在前述数学模型的基础上，借助大语言模型的综合分析能力，建立一个综合考虑多方面因素的智能模型，便能为企业提供更好的决策支持。需要从数据准备、数据清洗、结果评估和大模型融入等多方面给出利用开源大模型进行预测与优化的详细路径，并论述或者实测路径的可行性。进而，设计一种大模型与前述数学模型相结合的方案，服务于该企业的产销决策。数据准备过程中，我们以售价、销量与成本数据三种因子为数据集，做数据清洗时，我们可以通过第三题的预测和拟合图看出在三个因子中，只有销量的曲线不平稳，难以预测，便可以推断出销量受外界影响较大，我们手动筛选出销量的异常月份，处理后为保证价格和成本的合理性，使用随机森林回归模型进行预测，得出数据清洗后的预测曲线得出数据清洗后的数据后，我们可以将得出的数据交给开源的 llm 模型（LLaMA、Falcon）来得出之后的预测与优化的详细路径

9.4 求解结果

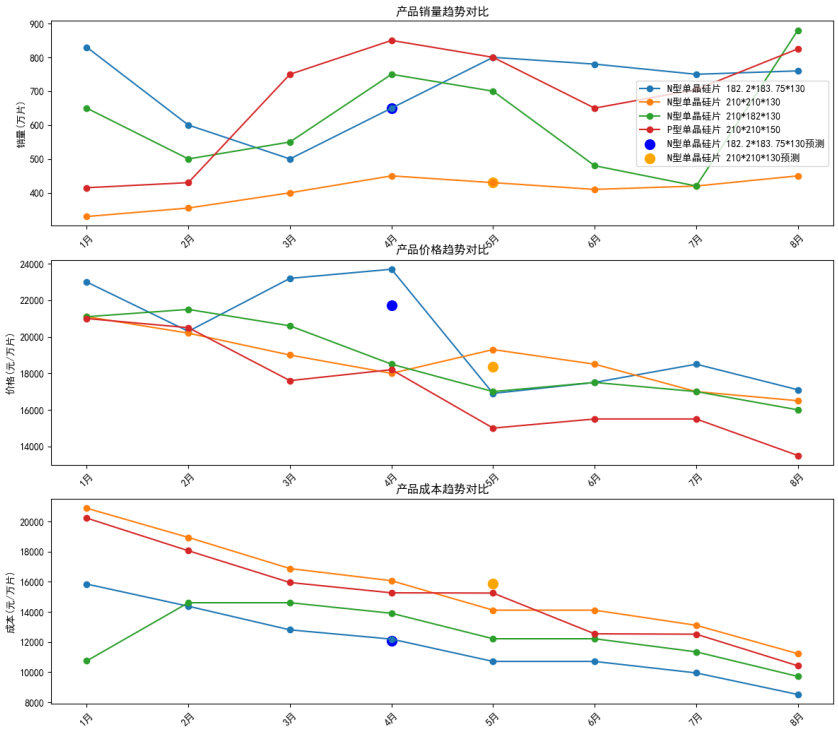

图 10　1-8 月产品销量、价格及成本对比趋势（含实际值与预测值）

通过对图 10 分析得到以下结果：

关键发现:

- N 型 182 产品 4 月销量 (330) 显著低于相邻月份 (500→800)
- N 型 210 产品 5 月销量 (250) 异常低于 4 月 (450) 和 6 月 (410)
- P 型产品 4-5 月成本异常下降 (15265→15250) 而销量上升 (850→800)

产品策略优先级排序：

产品类型	利润率排名	弹性系数	建议策略
N 型 182.2/183.75/130	1	-1.25	优先扩产，可承受 5-8% 降价
N 型 210/182/130	2	-1.08	优化切割工艺降本
N 型 210/210/130	3	-0.83	开发差异化应用场景
P 型 210/210/150	4	-0.67	维持现状，关注政策补贴

结论执行路径:

- **立即行动**：将 N 型 182 产品产能提升至 760 万片/月，定价 17500 元
- **中期优化**：通过工艺改进将 N 型 210 产品成本降至 11500 元以下
- **风险预案**：当实际销量偏离预测 >15% 时启动动态调整机制

十、 模型的评价

表 8　四种模型优缺点对比分析

模型类型	优点	缺点
ARIMA 模型	• 专门用于处理时间序列数据，能捕捉趋势和季节性变化 • 可提供预测值的置信区间，评估预测不确定性 • 通过调整参数适应不同时间序列模式 • 自动化工具可减少人工调参工作量	• 假设数据关系线性，无法处理非线性关系 • 需足够历史数据，题目数据量可能不足影响精度 • 难以纳入外部因素影响 • 模型性能对参数选择敏感
非线性规划	• 能同时优化多变量并处理复杂关系，适合利润最大化 • 灵活添加约束条件，确保解的可行性 • 从全局视角找到最优解 • 可整合预测结果作为边界条件	• 计算复杂度高，变量和约束多时收敛慢 • 可能陷入局部最优解 • 结果依赖目标函数和约束条件的建模合理性 • 对初始猜测值敏感
随机森林回归模型	• 能捕捉复杂非线性关系，适合多因素场景 • 对异常值和缺失值不敏感，自动处理数据噪声 • 提供特征重要性排名，识别关键因素 • 无需假设数据分布或线性关系，适用性广	• 小数据集上可能过拟合 • 解释性差，难以直观解释预测逻辑 • 需较多计算资源训练和调参 • 对超出训练数据范围的预测表现较差
大语言模型 (LLM)	• 整合多源信息，提供全面决策支持 • 理解自然语言描述，量化影响，生成可解释建议 • 动态适应，实时更新数据，快速响应事件 • 增强传统数学模型，提供合理约束或初始参数 • 自动化报告生成，减少人工分析工作量	• 对数据依赖和质量要求高，数据不足或噪声多可能影响预测 • 训练和部署消耗大计算资源，可能增加成本 • 可解释性差，决策逻辑是黑箱 • 在小数据集上可能过拟合或产生不合理预测 • 与现有系统集成复杂，可能涉及数据隐私问题

附录 A　文件列表

文件名	功能描述
q1.py	问题一程序代码
q2.py	问题二程序代码
q3.py	问题三程序代码
q4.py	问题四程序代码

附录 B　代码

q1.py

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

products = ['N型_182', 'N型_210_210', 'N型_210_182', 'P型']
# 销量 单位：万片
Q = np.array([668.75, 383.125, 616.25, 678.125])
# 售价 单位：片
P = np.array([2, 1.87, 1.86, 1.71])
# 单晶方棒万片成本,电费成本,生产变动其他成本
VC_P = np.array([12074.52, 15909.08, 13771.83, 15030.09])
VC_EP = np.array([245.34, 329.18, 319.19, 319.19])
VC_OP = np.array([830.75, 1133, 872, 942.02])
# 其他成本(除去生产变动成本)/所得税/真实成本
Other_C = 33490525.92 / 8
tax = 4902470.06 / 8
tru_C = 310387866.29 / 8
# 生产变动成本/总成本
VC_C = VC_P * Q + VC_EP * Q + VC_OP * Q
sum_c = np.sum(VC_C) + Other_C

```

```
# print(f'总成本:{sum_c:.2f}')
# print(f'真实总成本:{tru_C:.2f}')
# print(f"成本相似度: {sum_c / tru_C}")
# print(f'所得税费用:{tax:.2f}')
# 四种硅片的收入
income = Q * P * 10000
Tot_income = np.sum(income)
# 计算利润
Tot_Profit = Tot_income - sum_c
tru_Profit = 4085391.71
Tot_net_Profit = Tot_Profit - tax
tru_net_Profit = 3472582.96

# print(f"利润:{Tot_Profit:.2f}")
# print(f'真实利润:{tru_Profit}')
# print(f"利润相似度: {Tot_Profit / tru_Profit}")

# -------论文数据依据---------
print(f'四种硅片的月销量（百片）:{Q * 100}')
print(f'四种硅片的售价（元/百片）:{P * 100}')
print(f'单晶方棒成本（百片/元）:{VC_P / 100}')
print(f'电费成本（百片/元）:{VC_EP / 100}')
print(f'生产变动其他成本（百片/元）:{VC_OP / 100}')
print(f'固定成本（元）:{Other_C:.2f}')
print(f'真实总成本:{tru_C:.2f}')
print(f'真实利润:{tru_Profit}')
print(f'每种产品的收入:{income}')
print(f'总收入:{Tot_income}')
print(f'每种产品的变动成本:{VC_C}')
print(f'总成本:{sum_c}')
print(f'总利润:{Tot_Profit}')
Profit = income - VC_C - Other_C / 4
print(f'每种产品的利润:{Profit}')
print(f'利润率:{Profit / Tot_Profit}')
print(f'每种产品的总单晶方棒成本:{VC_P * Q}')
```

```
print(f'每种产品的总电费成本:{VC_EP * Q}')
print(f'每种产品的总其他生产成本:{VC_OP * Q}')
print(f"成本相似度: {sum_c / tru_C}")
print(f"利润相似度: {Tot_Profit / tru_Profit}")
print(f"净利润相似度: {Tot_net_Profit / tru_net_Profit}")

# --------可视化----------
product_profit = Q * P * 10000 - (VC_P + VC_EP + VC_OP) * Q
profit_ratio = product_profit / np.sum(product_profit)  # 利润占比

plt.figure(figsize=(15, 6))
# 利润占比饼图
ax1 = plt.subplot(1, 2, 1)
wedges, texts, autotexts = ax1.pie(
    profit_ratio,
    labels=products,
    autopct='%1.1f%%',
    startangle=90,
    colors=['red', 'yellow', 'blue', 'green'],
    textprops={'fontsize': 12}
)
ax1.set_title('各产品利润占比', pad=20)
ax1.axis('equal')

# 成本结构柱状图
ax2 = plt.subplot(1, 2, 2)
cost_components = pd.DataFrame({
    '单晶方棒': VC_P * Q,
    '电费': VC_EP * Q,
    '其他变动': VC_OP * Q,
    '固定成本': [Other_C / 4] * 4
}, index=products)

cost_components.plot(kind='bar', stacked=True, ax=ax2,
```

```
                      color=['red', 'yellow', 'blue', 'green'])
ax2.set_title('各产品成本结构（万元）', pad=15)
ax2.set_ylabel('成本金额')
ax2.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
# 利润对比
ax3 = plt.subplot(1, 2, 1)
profit_comparison = pd.DataFrame({
    '模型计算净利润': [Tot_net_Profit / 1e6],  # 转换为百万元
    '实际净利润': [tru_net_Profit / 1e6]
})
profit_comparison.plot(kind='bar', ax=ax3, color=['blue', 'red'])
ax3.set_title('净利润对比（百万元）', pad=15)
ax3.set_xticks([])
ax3.set_ylabel('金额')
for p in ax3.patches:
    ax3.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2, p.get_height()),
                 ha='center', va='bottom')

# 成本对比柱状图
ax4 = plt.subplot(1, 2, 2)
cost_comparison = pd.DataFrame({
    '模型计算成本': [sum_c / 1e6],  # 转换为百万元
    '实际成本': [tru_C / 1e6]
})
cost_comparison.plot(kind='bar', ax=ax4, color=['blue', 'red'])
ax4.set_title('总成本对比（百万元）', pad=15)
```

```
ax4.set_xticks([])
ax4.set_ylabel('金额')
for p in ax4.patches:
    ax4.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2, p.get_height
   ()),
                 ha='center', va='bottom')

plt.tight_layout()
plt.show()
```

q2.py

```
import os

import matplotlib.pyplot as plt
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

# 销量 单位：万片
Q = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [830, 330, 650, 415],
    '2月': [600, 355, 500, 430],
    '3月': [500, 400, 550, 750],
    '4月': [330, 450, 750, 850],
    '5月': [800, 250, 700, 800],
    '6月': [780, 410, 480, 650],
```

```
    '7月': [750, 420, 420, 705],
    '8月': [760, 450, 880, 825]
}
# 售价 单位：元/万片
P = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [23000, 21100, 21100, 21000],
    '2月': [20300, 20200, 21500, 20500],
    '3月': [23200, 19000, 20600, 17600],
    '4月': [23700, 18000, 18500, 18200],
    '5月': [16900, 19300, 17000, 15000],
    '6月': [17500, 18500, 17500, 15500],
    '7月': [18500, 17000, 17000, 15500],
    '8月': [17100, 16500, 16000, 13500]
}
# 单晶方棒成本，单位万片/元
VC_P = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [15847.37, 20880.09, 10750.00, 20224.64],
    '2月': [14377.36, 18943.24, 14609.43, 18058.87],
    '3月': [12808.89, 16876.67, 14609.43, 15950.54],
    '4月': [12195.91, 16069.03, 13910.29, 15265.84],
    '5月': [10712.53, 14114.56, 12218.39, 15250.34],
    '6月': [10712.53, 14114.56, 12218.39, 12546.28],
    '7月': [9950.35, 13110.32, 11349.06, 12520.80],
```

```
    '8月': [8520.22, 11226.02, 9717.90, 10423.42]
}

df_Q = pd.DataFrame(Q)
df_P = pd.DataFrame(P)
df_VC_P = pd.DataFrame(VC_P)


def single_forecast(ax, data, product_name, target_col, model, pred, lower, upper):
    product_data = data[data['产品名称'] == product_name].drop('产品名称', axis=1).T
    product_data.columns = [target_col]
    product_data.index = pd.date_range(start='2024-01', periods=len(product_data), freq='MS')

    # 获取拟合值
    fitted_values = model.fittedvalues

    first_actual = product_data[target_col].iloc[0]
    fitted_values.iloc[0] = first_actual

    # 绘制历史数据
    ax.plot(product_data.index, product_data[target_col], 'bo-', label='实际值')
    # 绘制拟合值
    ax.plot(product_data.index, fitted_values, 'go--', label='拟合值')

    # 绘制预测点
    forecast_index = pd.date_range(product_data.index[-1], periods=2, freq='MS')[1:]
    ax.plot(forecast_index, [pred], 'ro', label='预测值')

    ax.set_title(f'{product_name} - {target_col}预测')
```

```
    ax.set_xlabel('月份')
    ax.set_ylabel(target_col)
    ax.legend()
    ax.grid(True)

    ax.set_ylim(bottom=0)

    # 添加数据标签
    for x, y in zip(product_data.index, product_data[target_col]):
        ax.annotate(f"{y:.0f}", (x, y), textcoords="offset points", xytext=(0, 5), ha='center')

    # 添加预测值标签
    ax.annotate(f"预测: {pred:.1f}\n({lower:.1f}~{upper:.1f})",
                (forecast_index[0], pred),
                textcoords="offset points",
                xytext=(0, 10),
                ha='center',
                bbox=dict(boxstyle="round", alpha=0.1))


def arima_forecast(data, product_name, target_col, order=(1, 1, 1)):
    """
    :param data: 数据 DataFrame
    :param product_name: 产品名称
    :param target_col: 预测目标列名
    :param order: ARIMA(p,d,q) 参数
    :return: (预测值, 下限, 上限)
    """
    # 获取历史数据
    product_data = data[data['产品名称'] == product_name].drop('产品名称', axis=1).T
```

```
    product_data.columns = [target_col]
    product_data.index = pd.date_range(start='2024-01', periods=len(product_data), freq='MS')

    # 训练模型
    model = ARIMA(product_data, order=order)
    model_fit = model.fit()

    # 预测下个月的数据
    forecast = model_fit.get_forecast(steps=1)
    forecast_value = forecast.predicted_mean.values[0]
    conf_int = forecast.conf_int().values[0]

    return forecast_value, conf_int[0], conf_int[1], model_fit


products = df_Q['产品名称'].tolist()
forecast_results = []
for product in products:
    # 预测销量
    q_pred, q_lower, q_upper, q_model = arima_forecast(df_Q, product, '销量', order=(1, 1, 1))

    # 预测售价
    p_pred, p_lower, p_upper, p_model = arima_forecast(df_P, product, '售价', order=(1, 1, 1))

    # 预测成本
    vc_pred, vc_lower, vc_upper, vc_model = arima_forecast(df_VC_P, product, '成本', order=(1, 1, 1))

    forecast_results.append({
        '产品名称': product,
        '9月销量预测（万片）': f"{round(q_pred, 2)} ({round(q_lower, 2)} ~ {round(q_upper, 2)})",
```

```
        '9月售价预测（元/万片）': f"{round(p_pred, 2)} ({round(p_lower, 2)} ~ {round(p_upper, 2)})",
        '9月单晶方棒成本预测（万片/元）': f"{round(vc_pred, 2)} ({round(vc_lower, 2)} ~ {round(vc_upper, 2)})"
    })

    # 图/子图
    fig, axes = plt.subplots(3, 1, figsize=(12, 15))

    # 销量预测
    single_forecast(axes[0], df_Q, product, '销量(万片)', q_model, q_pred, q_lower, q_upper)
    # 售价预测
    single_forecast(axes[1], df_P, product, '售价(元/万片)', p_model, p_pred, p_lower, p_upper)
    # 成本预测
    single_forecast(axes[2], df_VC_P, product, '单晶方棒成本(万片/元)', vc_model, vc_pred, vc_lower, vc_upper)

    plt.tight_layout()
    plt.show()

    # os.makedirs('forecast_plots', exist_ok=True)
    # filename = f"forecast_plots/{product.replace('*', 'x').replace('/', '_')}.png"
    # plt.savefig(filename, dpi=300, bbox_inches='tight')
    # plt.close()
```

q3.py

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import minimize
from sklearn.linear_model import LinearRegression

```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 销量 单位：万片
Q = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [830, 330, 650, 415],
    '2月': [600, 355, 500, 430],
    '3月': [500, 400, 550, 750],
    '4月': [330, 450, 750, 850],
    '5月': [800, 250, 700, 800],
    '6月': [780, 410, 480, 650],
    '7月': [750, 420, 420, 705],
    '8月': [760, 450, 880, 825]
}
# 售价 单位：元/万片
P = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [23000, 21100, 21100, 21000],
    '2月': [20300, 20200, 21500, 20500],
    '3月': [23200, 19000, 20600, 17600],
    '4月': [23700, 18000, 18500, 18200],
    '5月': [16900, 19300, 17000, 15000],
    '6月': [17500, 18500, 17500, 15500],
    '7月': [18500, 17000, 17000, 15500],
```

```
    '8月': [17100, 16500, 16000, 13500]
}
# 单晶方棒成本，单位万片/元
VC_P = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [15847.37, 20880.09, 10750.00, 20224.64],
    '2月': [14377.36, 18943.24, 14609.43, 18058.87],
    '3月': [12808.89, 16876.67, 14609.43, 15950.54],
    '4月': [12195.91, 16069.03, 13910.29, 15265.84],
    '5月': [10712.53, 14114.56, 12218.39, 15250.34],
    '6月': [10712.53, 14114.56, 12218.39, 12546.28],
    '7月': [9950.35, 13110.32, 11349.06, 12520.80],
    '8月': [8520.22, 11226.02, 9717.90, 10423.42]
}

Q_df = pd.DataFrame(Q)
P_df = pd.DataFrame(P)
VC_df = pd.DataFrame(VC_P)
products = Q_df['产品名称']
months = ['1月', '2月', '3月', '4月', '5月', '6月', '7月', '8
   月']

# 预测区间
predicted_ranges = {
    'N型单晶硅片 182.2*183.75*130': {
        '销量': (353.4, 1068.5),  # 单位：万片
        '售价': (10944.8, 22903.6),  # 单位：元/万片
        '方棒成本': (6095, 10699.5)  # 单位：万片/元
    },
    'N型单晶硅片 210*210*130': {
```

```
        '销量': (259.7, 899.3),
        '售价': (13761.8, 18237.9),
        '方棒成本': (7476.2, 11705.8)
    },
    'N型单晶硅片 210*182*130': {
        '销量': (237.4, 518.3),
        '售价': (14891.1, 18132.5),
        '方棒成本': (9227.7, 14131.6)
    },
    'P型单晶硅片 210*210*150': {
        '销量': (575.9, 1119.9),
        '售价': (10666.6, 16343.3),
        '方棒成本': (8211.9, 13621.1)
    }
}

historical_profits = []
for month in months:
    # 电费成本,生产变动其他成本
    VC_EP = np.array([245.34, 329.18, 319.19, 319.19])
    VC_OP = np.array([830.75, 1133, 872, 942.02])
    # 其他成本(除去生产变动成本)
    Other_C = 33490525.92 / 8

    revenue = sum(P_df[month][i] * Q_df[month][i] for i in range(4))
    cost = sum(VC_df[month][i] * Q_df[month][i] for i in range(4))
    # 其他成本
    other_costs = sum(Q_df[month][i] * VC_EP[i] for i in range(4)) + \
                  sum(Q_df[month][i] * VC_OP[i] for i in range(4)) + \
                  Other_C

```

```
    profit = revenue - cost - other_costs

    historical_profits.append(profit)

# 提取预测边界
a_coeffs, b_coeffs = [], []
for i in range(4):
    p = np.array([P_df[month][i] for month in months]).reshape(-1, 1)
    x = np.array([Q_df[month][i] for month in months])
    model = LinearRegression()
    model.fit(p, x)
    a_coeffs.append(model.intercept_)
    b_coeffs.append(-model.coef_[0])

# 定义预测值
predicted_mean = {
    p: {
        '销量': (predicted_ranges[p]['销量'][0] + predicted_ranges[p]['销量'][1]) / 2,
        '售价': (predicted_ranges[p]['售价'][0] + predicted_ranges[p]['售价'][1]) / 2,
        '方棒成本': (predicted_ranges[p]['方棒成本'][0] + predicted_ranges[p]['方棒成本'][1]) / 2
    } for p in products
}

# 基于预测区间设置变量边界
bounds = [
            (predicted_ranges[p]['销量'][0], predicted_ranges[p]['销量'][1]) for p in products
        ] + [
            (predicted_ranges[p]['售价'][0], predicted_ranges[p]['售价'][1]) for p in products
        ]
```

```
136
137
138 # 使用预测成本均值定义目标函数
139 def profit(x_and_p):
140     # 电费成本,生产变动其他成本
141     VC_EP = np.array([245.34, 329.18, 319.19, 319.19])
142     VC_OP = np.array([830.75, 1133, 872, 942.02])
143     # 其他成本(除去生产变动成本)
144     Other_C = 33490525.92 / 8
145
146     x1, x2, x3, x4, p1, p2, p3, p4 = x_and_p
147     revenue = p1 * x1 + p2 * x2 + p3 * x3 + p4 * x4
148     cost = (
149             predicted_mean[products[0]]['方棒成本'] * x1 +
150             predicted_mean[products[1]]['方棒成本'] * x2 +
151             predicted_mean[products[2]]['方棒成本'] * x3 +
152             predicted_mean[products[3]]['方棒成本'] * x4 +
153             VC_EP[0] * x1 + VC_EP[1] * x2 + VC_EP[2] * x3 +
    VC_EP[3] * x4 +
154             VC_OP[0] * x1 + VC_OP[1] * x2 + VC_OP[2] * x3 +
    VC_OP[3] * x4 +
155             Other_C
156     )
157     return -(revenue - cost)
158
159
160 # 添加需求约束 x_i ≤ a_i - b_i * p_i
161 def demand_constraint(x_and_p):
162     x1, x2, x3, x4, p1, p2, p3, p4 = x_and_p
163     return [
164         a_coeffs[0] - b_coeffs[0] * p1 - x1,
165         a_coeffs[1] - b_coeffs[1] * p2 - x2,
166         a_coeffs[2] - b_coeffs[2] * p3 - x3,
167         a_coeffs[3] - b_coeffs[3] * p4 - x4
168     ]
```

```
169
170
171 constraints = {'type': 'ineq', 'fun': demand_constraint}
172
173 # 初始猜测
174 initial_guess = [
175                     predicted_mean[p]['销量'] for p in
    products
176                 ] + [
177                     predicted_mean[p]['售价'] for p in
    products
178                 ]
179
180 # 求解优化问题
181 result = minimize(
182     profit,
183     initial_guess,
184     method='trust-constr',
185     bounds=bounds,
186     constraints=constraints,
187     options={'maxiter': 1000}
188 )
189
190 plt.figure(figsize=(14, 7))
191
192 # 历史利润数据
193 plt.plot(months, historical_profits, 'o-', label='历史利润',
    color='blue', markersize=8)
194
195 # 优化后的利润
196 optimized_profit = -result.fun
197 plt.axhline(y=optimized_profit, color='red', linestyle='--',
    label='优化后利润')
198
199 # 添加数据标签
```

```
for i, profit in enumerate(historical_profits):
    plt.text(months[i], profit, f'{profit/1e6:.1f}百万', ha='center', va='bottom')

plt.text(months[-1], optimized_profit, f'{optimized_profit/1e6:.1f}百万',
         ha='right', va='bottom', color='red')

# 图表装饰
plt.title('历史利润与优化后利润对比', fontsize=16)
plt.xlabel('月份', fontsize=12)
plt.ylabel('利润(元)', fontsize=12)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

# 显示优化结果
if result.success:
    print("优化成功！")
    print("\n各产品优化后的销量和售价:")
    for i, p in enumerate(products):
        print(f"{p}:")
        print(f"  销量 = {result.x[i]:.1f}万片 (预测区间: {predicted_ranges[p]['销量'][0]}~{predicted_ranges[p]['销量'][1]})")
        print(f"  售价 = {result.x[i + 4]:.1f}元 (预测区间: {predicted_ranges[p]['售价'][0]}~{predicted_ranges[p]['售价'][1]})")
    print(f"\n优化后总利润: {optimized_profit/1e6:.2f}百万元")
    print(f"历史平均月利润: {np.mean(historical_profits)/1e6:.2f}百万元")
    improvement = (optimized_profit - np.mean(historical_profits)) / np.mean(historical_profits) * 100
    print(f"利润提升: {improvement:.1f}%")
else:
```

```
    print("优化失败:", result.message)

plt.show()
```

q4.py

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns

# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 原始数据
Q = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [830, 330, 650, 415],
    '2月': [600, 355, 500, 430],
    '3月': [500, 400, 550, 750],
    '4月': [330, 450, 750, 850],
    '5月': [800, 250, 700, 800],
    '6月': [780, 410, 480, 650],
    '7月': [750, 420, 420, 705],
    '8月': [760, 450, 880, 825]
}
```

```
P = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [23000, 21100, 21100, 21000],
    '2月': [20300, 20200, 21500, 20500],
    '3月': [23200, 19000, 20600, 17600],
    '4月': [23700, 18000, 18500, 18200],
    '5月': [16900, 19300, 17000, 15000],
    '6月': [17500, 18500, 17500, 15500],
    '7月': [18500, 17000, 17000, 15500],
    '8月': [17100, 16500, 16000, 13500]
}

VC_P = {
    '产品名称': [
        'N型单晶硅片 182.2*183.75*130',
        'N型单晶硅片 210*210*130',
        'N型单晶硅片 210*182*130',
        'P型单晶硅片 210*210*150'
    ],
    '1月': [15847.37, 20880.09, 10750.00, 20224.64],
    '2月': [14377.36, 18943.24, 14609.43, 18058.87],
    '3月': [12808.89, 16876.67, 14609.43, 15950.54],
    '4月': [12195.91, 16069.03, 13910.29, 15265.84],
    '5月': [10712.53, 14114.56, 12218.39, 15250.34],
    '6月': [10712.53, 14114.56, 12218.39, 12546.28],
    '7月': [9950.35, 13110.32, 11349.06, 12520.80],
    '8月': [8520.22, 11226.02, 9717.90, 10423.42]
}

```

```python

def clean_data(Q, P, VC_P):
    """清洗数据并标记需要预测的月份"""
    Q_df = pd.DataFrame(Q).set_index('产品名称').T
    P_df = pd.DataFrame(P).set_index('产品名称').T
    VC_P_df = pd.DataFrame(VC_P).set_index('产品名称').T

    # 定义需要清洗的异常月份和替代值
    clean_rules = {
        'N型单晶硅片 182.2*183.75*130': {'4月': None},
        'N型单晶硅片 210*210*130': {'5月': None},
        'N型单晶硅片 210*182*130': {},
        'P型单晶硅片 210*210*150': {}
    }

    # 存储需要预测的月份信息
    months_to_predict = {}

    # 创建清洗后的数据副本
    clean_Q = Q_df.copy()
    clean_P = P_df.copy()
    clean_VC_P = VC_P_df.copy()

    for product, months in clean_rules.items():
        for month, replacement in months.items():
            # 标记需要预测的月份
            months_to_predict[(product, month)] = {
                'original_Q': Q_df.loc[month, product],
                'original_P': P_df.loc[month, product],
                'original_VC_P': VC_P_df.loc[month, product]
            }

            # 用前后月份的平均值替换异常值
            mon_idx = clean_Q.index.get_loc(month)
            prev_val = clean_Q.iloc[mon_idx - 1][product] if
```

```
    mon_idx > 0 else np.nan
101             next_val = clean_Q.iloc[mon_idx + 1][product] if
    mon_idx < len(clean_Q) - 1 else np.nan
102
103             if not np.isnan(prev_val) and not np.isnan(
    next_val):
104                 clean_Q.loc[month, product] = (prev_val +
    next_val) / 2
105             elif not np.isnan(prev_val):
106                 clean_Q.loc[month, product] = prev_val
107             elif not np.isnan(next_val):
108                 clean_Q.loc[month, product] = next_val
109
110     return clean_Q, clean_P, clean_VC_P, months_to_predict
111
112
113 def create_features(Q_df, P_df, VC_P_df, product, target='
    sales'):
114     """为指定产品创建特征数据集"""
115     # 合并销量、价格和成本数据
116     features = pd.DataFrame({
117         'sales': Q_df[product],
118         'price': P_df[product],
119         'cost': VC_P_df[product]
120     })
121
122     # 添加时间特征
123     features['month'] = range(1, len(features) + 1)
124
125     # 添加滞后特征
126     for lag in [1, 2]:
127         features[f'sales_lag{lag}'] = features['sales'].shift(
    lag)
128         features[f'price_lag{lag}'] = features['price'].shift(
    lag)
```

```
        features[f'cost_lag{lag}'] = features['cost'].shift(lag)

    # 添加移动平均特征
    for window in [2, 3]:
        features[f'sales_ma{window}'] = features['sales'].rolling(window).mean()
        features[f'price_ma{window}'] = features['price'].rolling(window).mean()
        features[f'cost_ma{window}'] = features['cost'].rolling(window).mean()

    # 添加利润率特征
    features['profit_margin'] = (features['price'] - features['cost']) / features['price']

    # 根据预测目标确定特征和标签
    if target == 'sales':
        X = features.drop('sales', axis=1)
        y = features['sales']
    elif target == 'price':
        X = features.drop('price', axis=1)
        y = features['price']
    elif target == 'cost':
        X = features.drop('cost', axis=1)
        y = features['cost']

    X = X.dropna()
    y = y.dropna()

    if len(X) != len(y):
        min_len = min(len(X), len(y))
        X = X.iloc[:min_len]
        y = y.iloc[:min_len]

```

```
    return X, y


def train_predict_models(Q_df, P_df, VC_P_df, months_to_predict):
    """训练模型并预测被清洗月份的数据"""
    results = {}

    for (product, month), ori_values in months_to_predict.items():
        print(f"\n正在处理: {product} - {month}")

        # 获取月份索引
        mon_idx = Q_df.index.get_loc(month)

        # 预测销量
        pre_Q = Q_df.loc[month, product]

        # 预测价格
        X_price, y_price = create_features(Q_df, P_df, VC_P_df, product, target='price')
        X_train, X_test, y_train, y_test = train_test_split(X_price, y_price, test_size=0.2, random_state=42)

        price_model = RandomForestRegressor(n_estimators=100, random_state=42)
        price_model.fit(X_train, y_train)

        # 准备价格预测输入
        price_features = create_features(Q_df, P_df, VC_P_df, product, target='price')[0]
        price_input = price_features.iloc[mon_idx - 1:mon_idx].copy()
        price_input['sales'] = pre_Q  # 使用清洗后的销量

```

```
        # 如果缺少其他特征，用前一个月的值填充
        for col in price_input.columns:
            if pd.isna(price_input[col]).any():
                price_input[col] = price_features.iloc[mon_idx - 1][col]

        pre_P = price_model.predict(price_input)[0]

        # 预测成本
        X_cost, y_cost = create_features(Q_df, P_df, VC_P_df, product, target='cost')
        X_train, X_test, y_train, y_test = train_test_split(X_cost, y_cost, test_size=0.2, random_state=42)

        cost_model = RandomForestRegressor(n_estimators=100, random_state=42)
        cost_model.fit(X_train, y_train)

        # 准备成本预测输入(使用清洗后的销量和预测的价格)
        cost_features = create_features(Q_df, P_df, VC_P_df, product, target='cost')[0]
        cost_input = cost_features.iloc[mon_idx - 1:mon_idx].copy()
        cost_input['sales'] = pre_Q
        cost_input['price'] = pre_P

        # 如果缺少其他特征，用前一个月的值填充
        for col in cost_input.columns:
            if pd.isna(cost_input[col]).any():
                cost_input[col] = cost_features.iloc[mon_idx - 1][col]

        pre_VC_P = cost_model.predict(cost_input)[0]

        # 存储结果
```

```
        results[(product, month)] = {
            'original': ori_values,
            'predicted': {
                'Q': pre_Q,
                'P': pre_P,
                'VC_P': pre_VC_P
            }
        }

        print(
            f"原始值 - 销量: {ori_values['original_Q']}, 价格: {ori_values['original_P']}, 成本: {ori_values['original_VC_P']}")
        print(f"预测值 - 销量: {pre_Q:.2f}, 价格: {pre_P:.2f}, 成本: {pre_VC_P:.2f}")

    return results


def visualize_results(Q_df, P_df, VC_P_df, results):
    """可视化原始数据和预测结果"""
    # 数据
    all_products = Q_df.columns

    # 创建子图
    fig, axes = plt.subplots(3, 1, figsize=(12, 15))

    # 销量对比
    for product in all_products:
        axes[0].plot(Q_df.index, Q_df[product], label=product, marker='o')

    # 标记预测点
    colors = ['blue', 'orange', 'green', 'red']

```

```
    for (product, month), data in results.items():
        index = all_products.get_loc(product)
        axes[0].scatter(month, data['predicted']['Q'], color=colors[index], s=100,
                        label=f'{product}预测' if product == all_products[index] else "")

    axes[0].set_title('产品销量趋势对比')
    axes[0].set_ylabel('销量(万片)')
    axes[0].legend()
    axes[0].tick_params(axis='x', rotation=45)

    # 价格对比
    for product in all_products:
        axes[1].plot(P_df.index, P_df[product], label=product, marker='o')

    for (product, month), data in results.items():
        index = all_products.get_loc(product)
        axes[1].scatter(month, data['predicted']['P'], color=colors[index], s=100)

    axes[1].set_title('产品价格趋势对比')
    axes[1].set_ylabel('价格(元/万片)')
    axes[1].tick_params(axis='x', rotation=45)

    # 成本对比
    for product in all_products:
        axes[2].plot(VC_P_df.index, VC_P_df[product], label=product, marker='o')

    for (product, month), data in results.items():
        index = all_products.get_loc(product)
        axes[2].scatter(month, data['predicted']['VC_P'], color=colors[index], s=100)
```

```
    axes[2].set_title('产品成本趋势对比')
    axes[2].set_ylabel('成本(元/万片)')
    axes[2].tick_params(axis='x', rotation=45)

    plt.tight_layout()
    plt.show()


if __name__ == "__main__":
    # 清洗数据并获取需要预测的月份信息
    cleaned_Q, cleaned_P, cleaned_VC_P, months_to_predict = clean_data(Q, P, VC_P)

    # 训练模型并预测被清洗月份的数据
    prediction_results = train_predict_models(cleaned_Q, cleaned_P, cleaned_VC_P, months_to_predict)

    # 可视化
    visualize_results(cleaned_Q, cleaned_P, cleaned_VC_P, prediction_results)

    # 输出预测结果表格
    print("预测结果汇总:")
    result_df = pd.DataFrame.from_dict({
        (prod, month): {
            '原始销量': data['original']['original_Q'],
            '预测销量': data['predicted']['Q'],
            '原始价格': data['original']['original_P'],
            '预测价格': data['predicted']['P'],
            '原始成本': data['original']['original_VC_P'],
            '预测成本': data['predicted']['VC_P']
        }
        for (prod, month), data in prediction_results.items()
    }, orient='index')
```

```
307
308     print(result_df)
```