基于历史数据的晶硅片产销策略优化

摘要

本文基于企业历史数据，构建以利润最大化为目标的硅晶片产销策略优化模型。通过边际影响识别对利润影响显著的关键因子，并据此简化利润计算模型；其次，结合灰色预测模型等多种方法，基于历史数据对关键因子的波动趋势及合理变化区间进行预测；在利润模型与因子变化区间基础上，引入改进的粒子群优化算法，寻找最优产销方案；最后，针对市场与政策的外部非结构化数据，耦合大语言模型，实现产销策略优化。

针对问题一，本文采用弹性分析方法，衡量各影响因子对利润的边际贡献，并选取对利润边际影响占比前 50%的因子作为关键变量。最终纳入模型的核心因子包括四型硅片的销量与售价、单晶方棒的单价与单耗、水价与水耗、电价与电耗。在此基础上，构建基于收入与成本核算框架的利润计算模型，为企业科学制定经营策略奠定基础。

针对问题二，考虑到企业历史数据样本较少，本文选用 ARIMA 时间序列预测、灰色预测模型以及 Holt-Winters 指数平滑模型对关键因子进行预测。基于 1 至 7 月数据对 8 月情况进行拟合预测，分析和评估模型适应性，结果表明 ARIMA 模型在本数据集上表现不佳，适用性较弱。因此，本文最终采用灰色预测模型与 Holt-Winters 指数平滑模型，分别针对具有波动趋势和相对平滑变化的因子，输出各因子 9 月预测值及其变化区间。

针对问题三，基于问题一和问题二的分析结果，本文以利润计算模型为目标函数，以关键因子的变化区间作为约束条件，基于改进的粒子群算法构建了利润优化模型。通过网格搜索法确定粒子群算法的最优超参数配置。在原有数学模型基础上，求解出使利润达到最优时各关键因子（如产量、售价等）的最优组合，为企业制定科学合理的生产计划与销售策略提供量化依据。

针对问题四，在原有数学模型基础上，耦合非结构化数据，构建市场和政策因素的利润优化模型。通过网络爬虫采集政府及第三方网站相关政策与市场信息，利用 BERT 模型对非结构化文本进行情感分类并量化，并将情感得分耦合进约束函数中，结合问题三的 PSO 算法输出综合因素下的最优产销方案。

关键词：灰色预测；Holt-Winters 指数平滑；边际影响；粒子群算法；Bert 模型

目录

1 问题重述

1.1 问题背景

近年来，随着光伏产业和半导体产业的快速发展和其他下游应用场景的不断拓展，高纯度晶硅片作为光伏产业链中的核心基础材料，其市场需求持续增长，随之引发了单晶硅市场激烈的市场竞争。单晶硅企业面临原材料成本波动、市场价格变化频繁、产能布局复杂等多重挑战。在此背景下，如何在成本控制、利润优化与供需平衡之间实现动态调节，成为企业提升经营效率和维持市场竞争力的关键问题。特别是在市场环境高度不确定、供需关系趋于紧张的情况下，晶硅片企业亟需建立科学有效的决策机制，以优化产销策略、提升整体利润水平。具体而言，企业需要考虑硅单耗（每单位产品所消耗的硅棒量）、耗材价格（如化学试剂、电力、水等消耗品成本）、生产变动成本（如设备维护、原材料采购成本）、生产公用成本（公用药剂、供热等公用设施费用）和人工成本等的影响。此外，企业的利润不仅受到生产成本的影响，还与销售量、销售费用、管理费用和财务费用等因素密切相关。直接关系到原料利用效率和单位成本。此外，各类耗材费用，如化学试剂、电力、水资源等消耗性物资成本，也对企业的生产成本产生显著影响。与此同时，生产过程中还会产生一定的变动成本，例如设备运行与维护支出、原材料采购费用等；以及较为固定的生产公用成本，包括公用药剂消耗、供热系统费用等基础设施支出。人工成本同样构成企业日常运营的重要支出部分，不容忽视。然而，企业利润并非仅由生产成本决定，还受到多项运营性费用的共同影响。销售量直接关系到收入规模，而销售费用、管理费用和财务费用等也会在不同程度上缩减企业盈利空间。

因此，从数据驱动的角度出发，构建基于多变量分析的产销策略决策体系，为企业提供科学的策略支持，实现从经验驱动向智能决策的转变。同时，借助先进的建模方法，企业可以更准确地预测市场变化趋势，制定灵活且具有前瞻性的产销计划，从而在激烈的市场竞争中占据有利位置。因此，考虑晶硅片行业的发展趋势以及企业经营的实际需求，建立基于历史数据的产销策略研究具有重要意义。

1.2 问题回顾

本文具体要解决的问题如下：

问题 1：请建立便于企业进行决策分析的月利润计算模型，该模型需要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。

问题 2：请建立数学模型，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；再利用所建立的模型预测 9 月份各因子的波动趋势和变化区间。

问题 3：一般来说，售价越低，销量就越大；产量越大，企业固定成本平均到单位产品上的份额就越低，变动成本的波动也会影响企业利润。众多因子之间的相互制约与冲突让企业很难决策下个月的生产计划与销售策略。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。

问题 4：由于销量、售价以及原材料、重要耗材的价格会受到政策与市场等综合因素的影响，单独根据企业内部数据的建模难以得出上佳的产销策略。若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。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。

2 问题分析

2.1 问题一的分析

第一问需要我们建立月利润计算模型以供企业进行决策分析。并重点关注四型硅片的销量、售价、单晶方棒以及其他影响企业利润的重要决策因子。根据本题附件 1 中“企业利润计算说明文档”以及附件 2 企业 1-8 月生产系统数据，我们需要综合计算生产变动成本、生产公共成本、人工成本、折旧成本、营业税费、销售费用、管理费用、财务费用等。按照**总成本=固定成本+变动成本、企业利润**

=收入–成本的核算框架建立月利润计算模型。利用弹性分析衡量影响因子对利润的边际影响，按照敏感程度大小确定重要的影响因素，并纳入最终的利润计算模型。

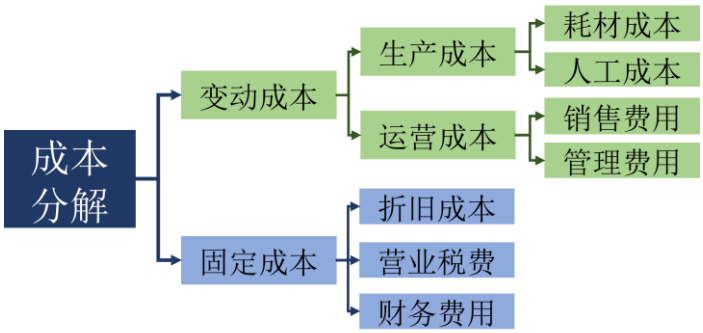


图 1 成本分解

2.2 问题二的分析

问题二需要建立四型硅片的月销量、售价、单晶方棒价格和其他重要因子的波动趋势预测模型，识别其随时间的变化规律，并预测下一月份的合理变化区间，进一步为销售与生产决策提供依据。受数据样本量的限制，本研究拟采用**ARIMA 时间序列预测模型、灰色预测模型和 Holt-Winters 指数平滑预测模型**三种预测模型，本题首先以 1 月至 7 月的数据为训练集，分别使用 ARIMA 与灰色预测模型、Holt-Winters 指数平滑预测模型对 8 月的数据进行预测，评估三类模型在不同因子上的适应性与预测精度。根据模型评估结果，为每一类影响因子选定最优预测模型，预测 9 月各因子变化区间。

2.3 问题三的分析

问题三需要在综合考虑影响企业利润的关键因子及其相互制约关系的基础上，实现利润的最大化。在建模过程中，可基于问题一中提出的利润计算模型，并结合企业生产运营中的重要影响因子，同时利用问题二中通过预测模型得到的九月份各因子的波动区间，明确因子的可行取值范围。综合上述因素，将问题转化为非线性优化问题。采用**改进的粒子群优化算法**（PSO）寻找变量最优组合，同时引入网格搜索算法对粒子群的超参数进行调优，以提升算法收敛效率和解的精度。以利润最大化为目标函数，确定各因子的最优组合，可作为企业下个月的

生产计划与销售策略参考。

2.4 问题四的分析

问题四需要我们综合考虑政策、市场以及企业内部对晶硅片销量、售价、原材料、耗材的影响，需要在前述的数学模型基础上，耦合大语言模型，实现考虑结构化数据、非结构化数据的多方面因素的混合模型，以便更好地为企业提供决策。该问题采用基于 Transform 架构的 Bert 模型实现对非结构化数据的量化。首先，借助网络爬虫技术从相关政府官方网站、第三方媒体网站（例如 www.chinapv.org.cn；www.miit.gov.cn；www.alibaba.com 等）爬取晶硅片相关信息，以爬取到的数据作为训练集输入预训练好的 Bert 模型中进行参数微调；然后，对 Bert 模型输入自定义非结构化数据实现文本情感分类与情感分数量化；最终，将量化后的情感分数耦合到原利润函数中，再使用 PSO 算法输出最终的预测结果。

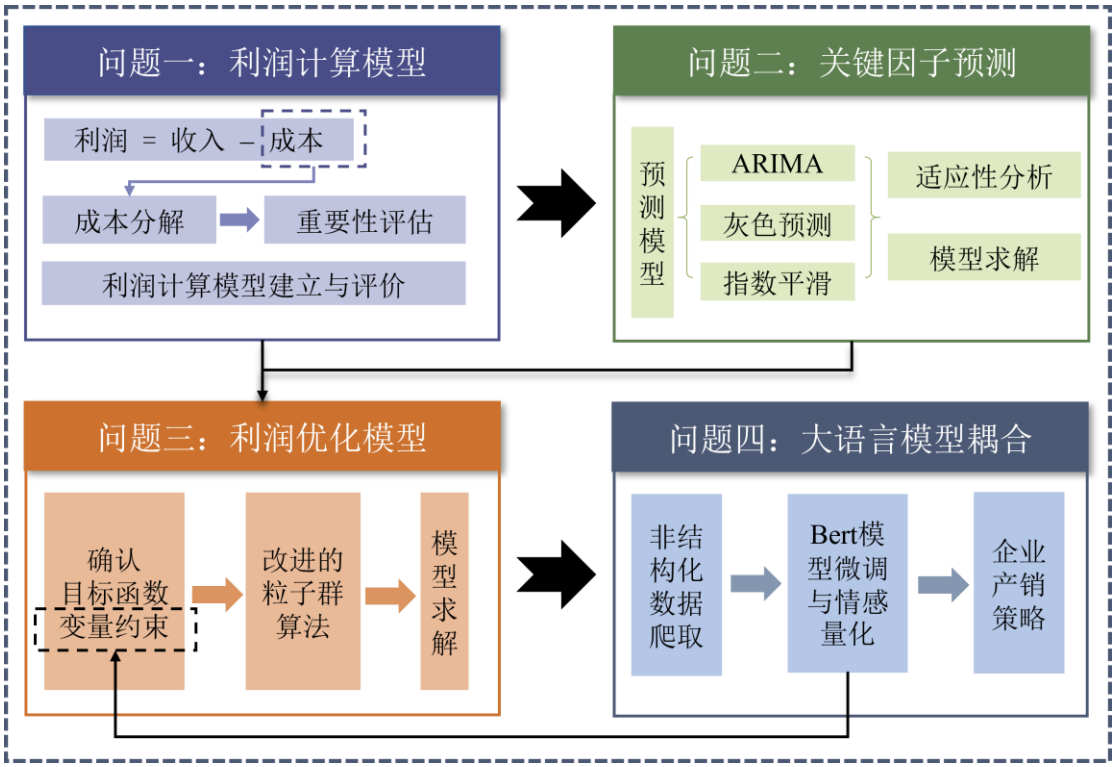


图 2 本文总流程图

3 模型假设与符号说明

3.1 模型假设

为了方便模型的建立和模型的可行性，我们这里首先对模型提出一些假设，使得模型更加完备，求出的结果更加合理。

（1）附件给出的所有数据均真实、有效；

（2）假设企业 0 库存，即产销平衡；

（3）本文计算利润均为税前利润；

（4）前三题分析中不考虑宏观社会经济水平、政策变动、行业外部冲击等外部环境因素对各因变量（如利润、售价、销量等）的影响，仅基于附件所给企业内部数据进行建模与预测分析；

（5）对于部分耗材，若其在 2024 年 1 至 8 月期间耗材成本保持不变，则认为该类耗材单价在预测期内维持定值。

3.2 符号说明

为了方便我们模型的建立与求解过程，我们这里对使用到的关键符号进行一下说明：

表 1 符号说明

符号	符号说明
π	销售利润
R	销售收入
C	总成本
π_n	销售净利润
Q_k	k 型硅单片销量
P_k	k 型硅单片售价
FC	固定成本
VC	变动成本
DC	折旧成本
TC	营业税费
FC	财务费用
LC	人工成本
MC	耗材成本
SC	销售费用
GC	管理费用
N_1	生产人员（计件人员+非计件人员）
N_2	安环物控人员人数

符号	符号说明
N_3	管理人员人数
L_{base}	基本工资总额
L_{fund}	社保公积金
L_{cont}	管理人员浮动工资
L_{reward}	年终奖计提
L_{fix}	固定人工成本
L_{fluc}	浮动人工成本
S_k	k 型硅片的硅棒单耗
m_k	k 型硅片的重量
α_k	第 k 型硅片的槽距
β_k	真实折算系数
θ_k	成品率
$e_{1,k}$	第 k 型硅片单位产量耗电量
e	总耗电量
p_e	电价
$B_{i,k}$	第 i 类耗材在第 k 型晶硅片生产中每万片消耗量
$P_{i,k}$	第 i 类耗材单价
w	用水量
p_w	水价
G_k	第 k 类硅棒单价
$S_{硅泥}$	硅泥销售收益

4 模型建立与求解

4.1 问题一模型建立与求解

4.1.1 销售收入—成本—利润模型

为便于企业在经营管理中科学地进行决策分析，本文构建了一个基于“销售收入—销售成本—销售利润”线性关系的月利润计算模型。该模型综合考虑了影响企业利润的多种关键因素，涵盖四型硅片销量、售价、单晶方棒产出量等主要经营变量，同时也纳入了生产和运营过程中产生的各类成本因子，以全面、系统反映企业盈利能力。模型的核心计算公式为：销售利润=销售收入–销售成本。其中，销售收入部分主要由四型硅片的销售数量与单位售价共同决定，即为各类型硅片的销量与相应售价的乘积之和。销售成本则细分为固定成本和变动成本两大类。固定成本包括折旧、营业税费和财务费用等在短期内相对稳定、不随产量变

动的支出；而变动成本则与生产规模密切相关，主要包括生产成本和运营成本两部分。生产成本进一步细化为耗材成本与人工成本，反映了硅片制造过程中直接投入的原材料、电力等资源以及人力资源；运营成本则包括销售费用和管理费用，用于支持企业日常运营和市场拓展活动。通过该模型的构建，可助力企业明晰销售收入与各类成本项之间的关系，帮助企业识别影响利润的关键变量，便于企业根据销售情况及成本花费确定利润，为企业制定新一轮生产计划和销售策略提供支持。具体各项成本与收入的计算方法将于 4.1.2 节中进一步阐述。

模型核心表达为：

$$\pi = R - C$$

销售收入为四型销售硅片的总收入构成，计算公式为：

$$R_k = \sum_{i=1}^{4} Q_k \times P_k$$

总成本包括固定成本和变动成本，包括折旧费用、营业税费、财务费用、人工成本、耗材成本、销售费用、管理费用。

$$C = FC + VC = (DC + TC + PC) + (LC + MC + SC + GC)$$

企业总利润需上交 15%所得税费用，实际净利润为 85%总利润，计算方法如下：

$$\pi_n = 0.85\pi$$

按照上述计算方法构建基于“销售收入—销售成本—销售利润”线性关系的月利润计算模型，便于企业根据该月的盈利情况及时调整销售和生产计划。

4.1.2 成本组成与计算

在企业利润核算中，成本是影响利润水平的核心因素之一，提高销售水平的同时需要控制成本。为全面准确地反映成本结构，本文将总成本划分为固定成本和变动成本两大类，并明确了各组成部分的计算方法。

固定成本（FC）是指在一定生产规模下相对不随产量变化而变动的成本，根据附件 2 中提供成本汇总情况，该企业不变动成本主要包括折旧费用（DC）、营业税费（TC）和财务费用（PC）三项。变动成本（VC）则随生产与销售规模变化而变动，主要包括人工成本（LC）、耗材成本（MC）、销售费用（SC）和管理费用（GC）。下面将对企业生产的各项成本详细说明。

（1）耗材成本/生产成本

生产成本主要包括生产变动成本和生产公用成本，生产变动成本包括多种耗材成本，生产成本包为生产成本中不随产量变化、或价格低、消耗量大的耗材成本。

生产变动成本为：

$$\mathrm{MC}_1 = (\sum_{k=1}^{4}(\frac{m_k \times \alpha_k}{\beta_k \times \theta_k} \times P_k + e_{1,k} \times p_{e1,k} + \sum_{i=1}^{17} B_{i,k} \times P_{i,k}) \times Q_k) - S_{硅泥}$$

生产公用成本：

$$\mathrm{MC}_2 = \sum_{k=1}^{4}\sum_{i=1}^{7} A_{i,k} \times P_{i,k}$$

$$MC = \mathrm{MC}_1 + \mathrm{MC}_2$$

（2）生产成本——人工成本

根据附件信息，企业人工组成包括生产人员、安环物控人员以及管理人员，设定上述人工组成人数分别为 N_1、N_2、N_3。企业人工成本的基本组成部分为基本工资、社保公积金、安环物控人员工资、管理人员浮动工资、年终奖计提。除此之外，生产人员中一部分是计件人员，这部分人员除基本工资外，还将按产量发放计件工资，计件工资计入浮动人工成本，具体人工成本计算方法如下：

基本工资总额：

$$L_{base} = (N_1 + N_3) \times 3000$$

社保公积金：

$$L_{fund} = (N_1 + N_3) \times 1000$$

安环物控人员工资：

$$L_{cont} = N_2 \times 8000$$

管理人员浮动工资：

$$L_{manage} = N_3 \times 8000$$

年终奖计提：

$$L_{reward} = L_{base}/12$$

固定人工成本：

$$L_{fix} = L_{base} + L_{fund} + L_{cont} + L_{manage} + L_{reward}$$

浮动人工成本：

$$L_{flow} = Q_k \times 0.04$$

人工成本总和：

$$LC = L_{fix} + L_{flow}$$

（3）运营成本（销售费用+管理费用）和固定成本

运营费用中的销售费用和管理费用以及固定成本根据公司提供的实际数据计算。

4.1.3 重点影响因子特征选择

我们在上述分析中详细阐述了在企业运营中的全部成本组成。实际上，在基于月利润进行企业决策分析时，固定成本以及可变成本中单价稳定、耗用量小的耗材对企业生产与销售计划的影响相对较弱。为降低模型复杂度，可将此类成本项视为常数处理。因此，本研究在剔除固定成本及可变成本中不变的耗材成本后，结合重点影响因子提取方法，识别出对企业利润具有显著影响的关键决策变量，并据此构建利润分析模型。

在剔除固定成本及可变成本中单价固定的耗材成本后，纳入利润影响分析的主要因子包括：四型硅片的销量、售价，以及主要耗材成本（如单晶方棒、电费、用水费、电镀金刚线、水性切割液、美纹胶带、无水乙醇、主辊-涂布消耗量、主辊-开槽、自来水等）、销售成本、管理成本。此外，人工成本中仅浮动部分金额发生变化，且在各月人工构成保持一致的前提下，其变动主要由销量驱动，因此可将人工成本的影响归结为销量对利润的影响。

在将上述筛选后的影响因子采用弹性系数分析衡量影响因子变动对利润变动的边际影响[1]或敏感程度，我们根据各变量对利润变异的敏感程度进行排序，评估其相对贡献度。采用弹性分析的基本理论为当因子发生 1%时，利润π的变化。其数学模型为：

$$E_x^{\pi} = \frac{\mathrm{d}\pi}{\mathrm{d}x} \cdot \frac{x}{\pi}$$

为提高模型的解释力与简洁性，本研究引入弹性系数分析方法，评估各影响因子对企业利润的相对影响程度。具体而言，我们分别对前 8 个月的历史数据进行逐月分析，并以 1 月份数据为例，计算各因子对利润的弹性系数。弹性系数衡量的是某一因子变化 1%所引起的利润变化，能有效反映利润对不同因子的敏感

程度。基于计算结果，我们将弹性系数排名前 50%的变量选定为主要决策因子，以减少模型复杂度，同时保留对利润具有实质性影响的核心变量。

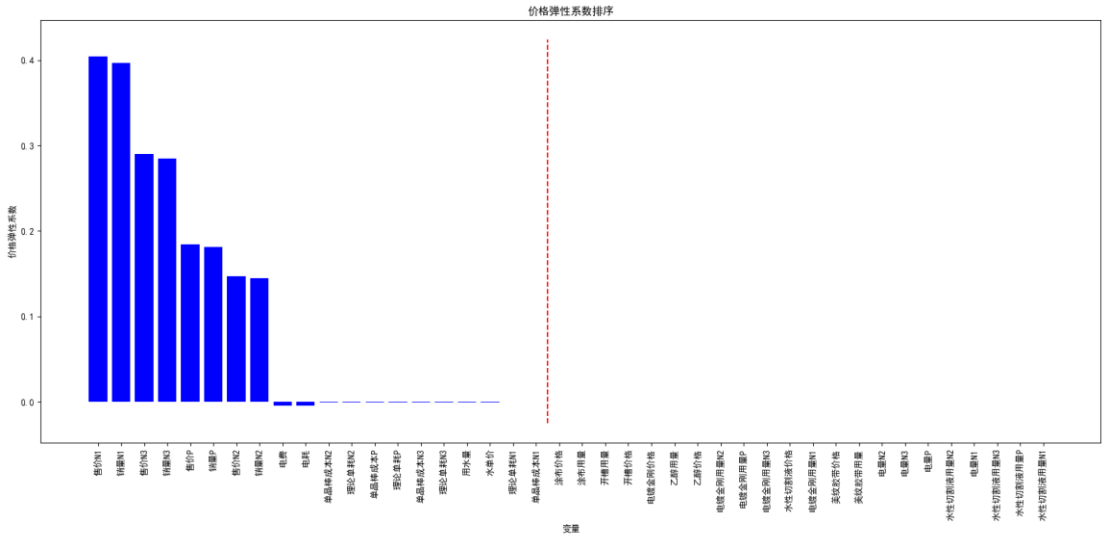


图 3 弹性系数分析结果

最终入选的重要因子包括：四型硅片的售价与销量、单晶方棒、水费，以及生产成本中的一般电耗成本。这些因子在多个时间点均表现出较高的利润弹性，其弹性系数结果见表 2。

表 2 重要影响因子及其弹性系数

影响因子	弹性系数	影响因子	弹性系数
N1 售价	0.40403	N2 单晶棒成本	-0.00044
N1 销量	0.39701	N2 理论单耗	-0.00044
N3 售价	0.29027	P 单晶棒成本	-0.00043
N3 销量	0.28477	P 理论单耗	-0.00043
P 售价	0.18445	N3 单晶棒成本	-0.00038
P 销量	0.18094	N3 理论单耗	-0.00038
N2 售价	0.14737	用水量	-0.00035
N2 销量	0.14458	水单价	-0.00035
电费	-0.00473	N1 理论单耗	-0.00034
电耗	-0.00470	N1 单晶棒成本	-0.00034

4.1.4 利润计算模型建立与求解

考虑上述重要因子：四型硅片销量、售价、单晶方棒、电费水费等，考虑到销量对其他售价的影响（例如硅泥等），直接与销量有影响的成本将被考虑，最终建立一下模型：

利润 = 四型硅片收入 – 单晶方棒成本 – 水电费 – 其他由销量影响的因子

$$\pi = \sum_{k=1}^{4} Q_k \times P_k + \sum_{k=1}^{4} 0.02Q_k - \sum_{k=1}^{4} Q_k \times G_k \times S_k \times 10000 - p_e \times e - p_w \times w - \sum_{k=1}^{4} 0.04Q_k$$
$$- (755.76Q_1 + 1093.14Q_2 + 871.71Q_3 + 941.73Q_4)/10000 - 2973012$$

公式简化后：

$$\pi = \sum_{k=1}^{4} Q_k \times (P_k - G_k \times S_k - 0.02) - p_e \times e - p_w \times w - (755.76Q_1 + 1093.14Q_2$$
$$+ 871.71Q_3 + 941.73Q_4)/10000 - 2973012$$

其中模型 755.76、1093.14、871.71、941.73 为变动成本中收售价变化较大因子均值，2973012 为所有固定成本（包括固定成本和变动成本中的不变成本）。通过销量、售价、单晶方棒价格、水电费用直接影响利润。可利用 python 实现利润计算，通过以上模型可比较实际利润（税前）和模型计算利润（税前）差值。

表 3 模型计算利润与实际利润

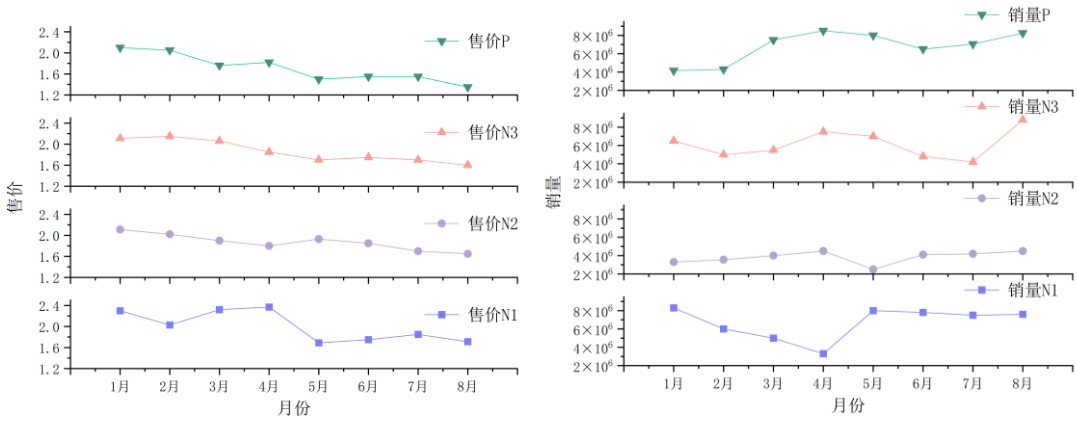
月份	税前实际利润	税前模型计算利润
1 月	1520788.426	1994587.660
2 月	1527642.283	1672538.079
3 月	3951004.689	4128399.208
4 月	4069707.790	3996856.159
5 月	-153334.546	-265350.4021
6 月	5291204.755	5102753.874
7 月	6339855.694	6106961.407
8 月	10207146.040	9730891.657

4.2 问题二模型建立与求解

4.2.1 重要因子变化趋势分析

本题旨在利用 1－8 月企业系统数据，预测影响企业利润的关键因子（如四型硅片月销量、售价、原材料价格等）的未来变化趋势及合理区间，并据此预测 9 月份各因子的波动趋势与变化范围。为实现该目标，需对现有数据进行统计分析，识别各关键因子的变化趋势特征，例如递增、递减或波动性变化等，从而为预测模型的选择提供依据。以售价和销量为例（见图 4），四型硅片的售价整体呈现出持续下降的趋势，因此更适合采用灰色预测模型或 ARIMA 时间序列预测等预测模型；而销量则呈现出较强的波动性，灰色预测等其他预测模型在此类特征

下的拟合效果较差，因此需考虑更适合波动性数据的模型进行处理。



（a）　　（b）

图 4 四型硅片销量及售价变化趋势分析（a）售价变化趋势；（b）销量变化趋势

考虑到数据样本量较少（仅 1–8 月数据），传统依赖大量历史数据的深度学习模型并不适用，本文选取两种更适应小样本、短期预测场景的方法进行建模：（1）ARIMA 时间序列预测模型：适用于具有稳定性或可通过差分平稳化的时间序列，能够捕捉趋势、周期与随机成分，具有较强的解释性和预测能力。（2）灰色预测模型：适合样本容量小、系统信息不完全的预测问题，建模简单、参数估计稳定，尤其适用于短期发展趋势预测。（3）Holt-Winters 指数平滑预测，适用于具有平稳性或弱趋势性的时间序列，尤其适合中短期预测，对数据波动敏感，相比灰度预测更擅长处理波动性的数据。下面我们分析各模型在本数据集的适应度，以提高预测精度。

4.2.2 模型适应性分析与求解

为实现对关键影响因子的精确预测，需对各预测模型的适应度进行评价。本研究采用 1-7 月的数据对第 8 月的因子值进行拟合预测，并以平均相对误差作为模型评价指标，误差越小表明模型预测精度越高。最终选取拟合效果最佳的模型用于预测 9 月份各关键因子的变化区间及具体预测值。

4.2.2.1 ARIMA 模型

ARIMA 模型是指通过对时间序列进行差分处理，使其平稳后利用自回归和移动平均项对数据的内在相关性进行建模[2]。我们首先通过探索性分析对第一类 N 型单晶硅片的销量进行了平稳性检验，第一类 N 型单晶硅片的销量原始时间

序列如图 1 所示。通过原始时序图可以发现序列呈现波动下降趋势。随后进行了 ACF 与 PACF 图检验，图 2 中展示了原始数据序列一阶拆分前后自相关图和偏自相关图。图中表明大部分编号位于置信区间内，说明序列基本平稳。原始序列进行一阶差分后 ACF、PACF 图均为 0 阶拖尾，因此一阶差分后的序列为平稳序列，p 和 q 均等于 0。

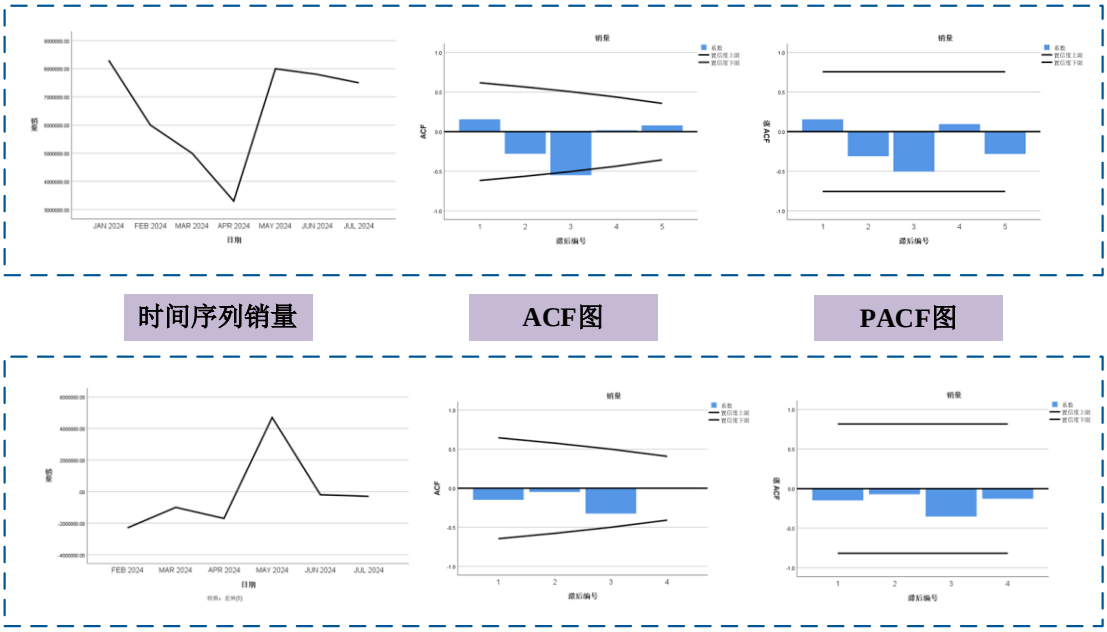


图 5 ARIMA 模型平稳性检验

根据之前所确定的p、d、q确定模型ARIMA(0,1,0),在SPSS系统中进行建模，拟合效果如图3所示。

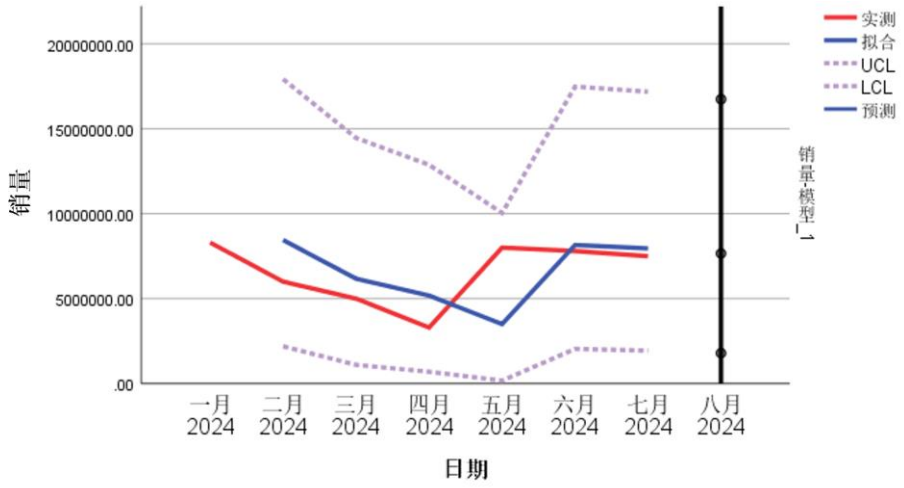


图6 ARIMA预测模型拟合效果

ARIMA预测模型拟合效果参数值见表4。

表4 ARIMA预测模型拟合效果

模型	预测变量数	模型拟合统计量			Ljung-BoxQ()		离群值数
ARIMA(0,1,0)	0	R方	正态化的BIC	统计量	DF	Sig	
		-0.814	29.77	.	0	.	0

由于本次拟合为手动创建 ARIMA 模型，没有选择“自动检测离群值”选项，从拟合图形上来看，拟合曲线滞后于原始时序曲线，并且 R 方数值太低，采用 ARIMA 模型效果交叉，模型在本数据集中适应度较低，因此不予考虑该模型。

4.2.2.2 灰色预测模型

灰色预测模型是一种基于原始数据累加生成，构建一阶微分方程来模拟和预测数据变化趋势的数学模型[3]。对于变化趋势较为平稳、未出现明显波动的影响因子，适宜采用灰色预测进行建模分析。灰色预测模型通过对原始时间序列进行累加生成一个新的“生成序列”，该序列在一定程度上削弱了数据的随机性、增强了其内在规律性。在此基础上，模型利用微分方程对生成序列进行拟合，从而实现对系统未来行为的有效预测。

首先对数据预处理并生成累计生产序列：

$$x^{(0)} = \left\{x^{(0)}(k) \mid_{k=1,2,\dots,N}\right\}$$

$$x^{(1)}(k) = \left\{\sum_{j=1}^{i} x^{(0)}(k) \;\middle|\; k = 1,2,\dots,N\right\}$$

灰色预测模型 GM(1,1)假设生成序列$x^{(1)}$满足如下一阶线性微分方程：

$$\frac{dx^{(1)}(t)}{dt} + ax^{(1)}(t) = b$$

其中，a为发展系数，b为灰色作用量。

使用离散近似表示微分方程，采用背景值$z^{(1)}(k)$：

$$x^{(1)}(k) = \frac{1}{2}\left(x^{(1)}(k) + x^{(1)}(k-1)\right), k = 2,3,\dots,n$$

$$x^{(0)}(k) + az^{(1)}(k) = b$$

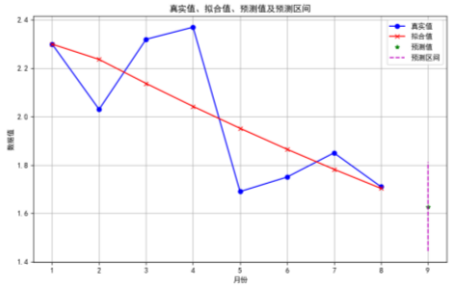
根据上述公式矩阵形式并进行参数估计，计算 a，b。

解微分方程，得到预测模型如下：

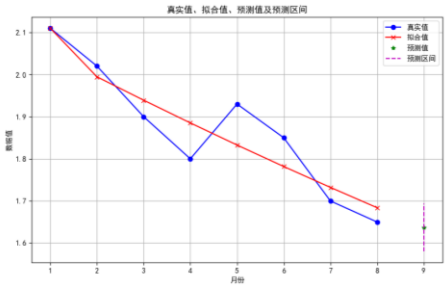
$$\hat{x}^{(1)}(k)=\left(x^{(0)}(1)-\frac{b}{a}\right)e^{-a(k-1)}+\frac{b}{a},k=1,2,\dots,n$$

$$\hat{x}^{(0)}(k)=\hat{x}^{(1)}(k)-\hat{x}^{(1)}(k-1),k=1,2,\dots,n$$

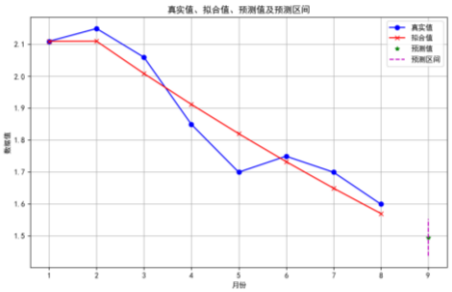
图 5 展示了利用灰色预测模型对四型硅片售价及单晶方棒的变化区间及具体预测值的结果。



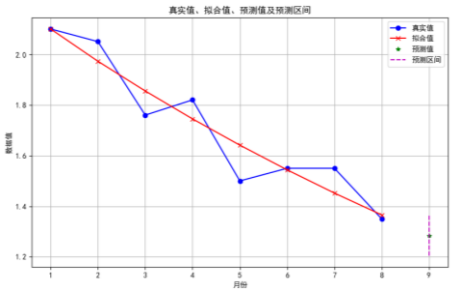
（a）N1 晶硅片售价



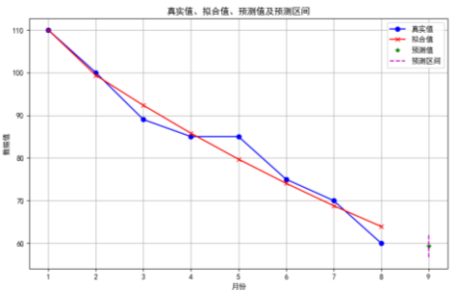
（b）N2 晶硅片售价



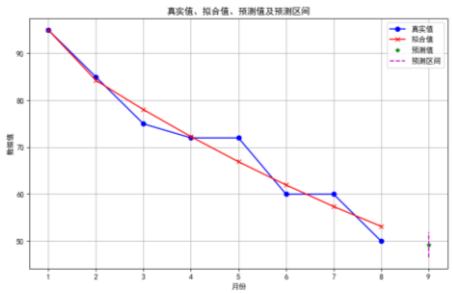
（c）N3 晶硅片售价



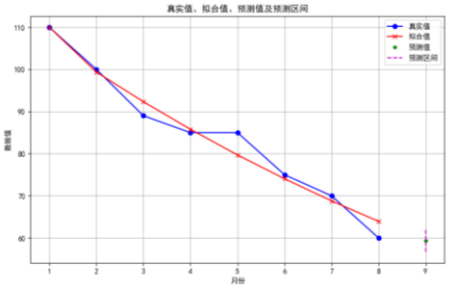
（d）P 晶硅片售价



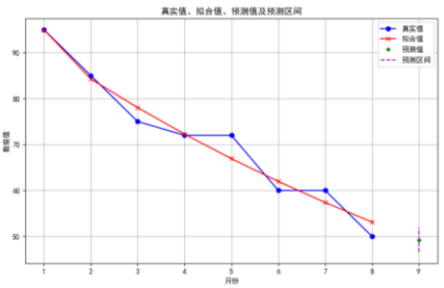
（e）N1、N2、N3 单晶方棒



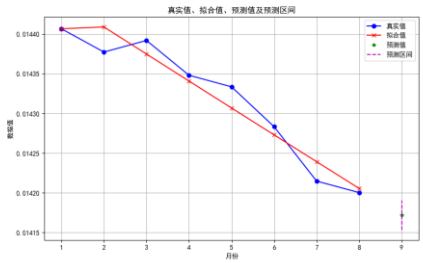
（f）P 单晶方棒



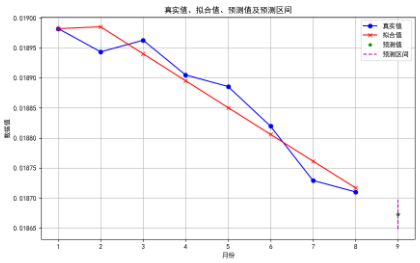
（g）N1、N2、N3 单晶方棒成本



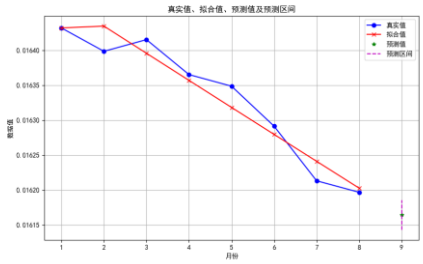
（h）P 单晶方棒成本



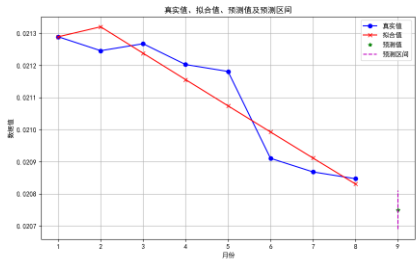
（i）N1 理论单耗



（j）N2 理论单耗



（k）N3 理论单耗



（l）P 理论单耗

图 7 灰度预测模型预测四型硅片售价因子结果

区间用残差估计计算，预测结果如下：

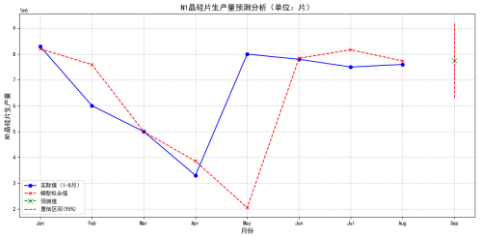
表 5 灰度预测结果

因子	平均预测误差	9 月预测值	九月变化区间
N1 晶硅片售价	7.25%	1.63	[1.48，1.75]
N2 晶硅片售价	2.59%	1.64	[1.58，1.69]
N3 晶硅片售价	2.57%	1.50	[1.45，1.55]
P 晶硅片售价	3.83%	1.28	[1.21，1.36]
N1、N2、N3 单晶方棒	2.63%	59.39	[56.70, 62.08]
P 单晶方棒	3.28%	49.20	[46.55，51.85]
N1 单晶方棒单价	2.63%	59.38577	[56.70, 62.08]
P 单晶方棒单价	3.28%	49.20084	[46.55, 51.85]
N1 理论单耗	0.11%	0.01417	[0.01, 0.01]
N2 理论单耗	0.11%	0.01867	[0.02, 0.02]
N3 理论单耗	0.11%	0.01616	[0.02, 0.02]
P 理论单耗	0.24%	0.02075	[0.02, 0.02]

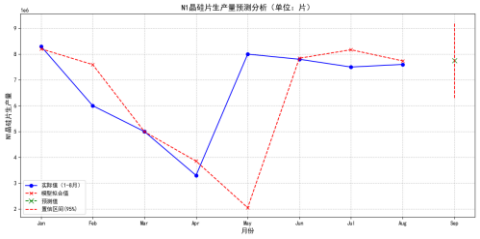
4.2.2.3 Holt-Winters 指数平滑预测模型

Holt-Winters 指数平滑通过分解时间序列的水平、趋势和季节性，提供了一种高效且解释性强的预测框架。其核心在于通过指数衰减权重平衡历史信息与近期变化，特别适合中短期、周期性明显的数据场景[4]。由于本题中数据较少，季节性数据难以判断，因此采用水平成分（l_t）和趋势部分（b_t）分别反应时间序列的基础水平和水平的线性增长或下降，以最小化误差为目标进行网格搜索选取最

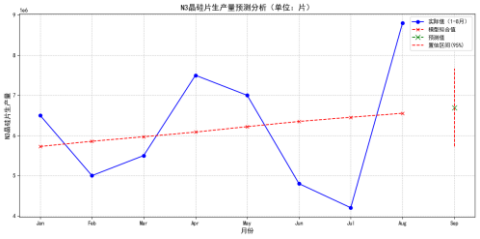
佳水平平滑因子（α）和趋势平滑因子（β），以四型硅片销售量为例，指数平滑预测结果如下：



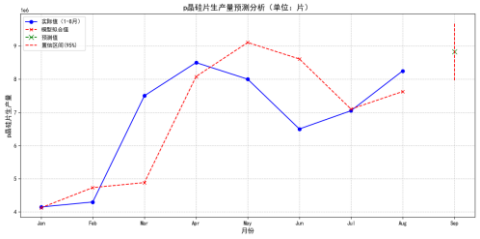
（a）N1 晶硅片销量



（b）N2 晶硅片销量



（c）N3 晶硅片销量



（d）P 晶硅片销量

图 8 四型硅片销售量预测结果

预测结果如下：

表 6 基于 Holt- Winters 指数平滑预测结果

因子	平均预测误差	9 月预测值	9 月变化区间
N1 晶硅片销量	16.28%	7750200	[6297984，9202416]
N2 晶硅片销量	11.80%	4376631	[3957131，4766131]
N3 晶硅片销量	14.38%	6696161	[5728362，7663961]
P 晶硅片销量	13.12%	8827595	[7964622，9690567]
电单价	1.74%	0.624	[0.623，0.625]
电耗	12.15%	332593	[288397，376789]
电费	12.36%	193762	[166303，221222]
水单价	0.00%	1.669	[1.669，1.670]
用水量	5.38%	8897	[7344，10449]
水费	5.36%	14856	[13969，15742]

基于灰色预测和 Holt- Winters 指数平滑预测结果可得出四型硅片的销量、单价、单晶方棒单价理论单耗、水单价及水耗、电单价及单耗的预测值及变化区间结果见下表：

表 7 所有重要因子预测结果及区间

重要因子	9 月预测值	9 月变化区间
N1 销量	7750200	[6297984，9202416]
N2 销量	4376631	[3957131，4766131]
N3 销量	6696161	[5728362，7663961]

重要因子	9 月预测值	9 月变化区间
P 销量	8827595	[7964622，9690567]
N1 晶硅片售价	1.63	[1.48，1.75]
N2 晶硅片售价	1.64	[1.58，1.69]
N3 晶硅片售价	1.50	[1.45，1.55]
P 晶硅片售价	1.28	[1.21，1.36]
N1、N2、N3 单晶方棒单价	59.38577	[56.70, 62.08]
P 单晶方棒单价	49.20084	[46.55, 51.85]
N1 理论单耗	0.01417	[0.01, 0.01]
N2 理论单耗	0.01867	[0.02, 0.02]
N3 理论单耗	0.01616	[0.02, 0.02]
P 理论单耗	0.02075	[0.02, 0.02]
电单价	0.624	[0.623，0.625]
电耗	332593	[288397，376789]
电费	193762	[166303，221222]
水单价	1.669	[1.669，1.670]
用水量	8897	[7344，10449]
水费	14856	[13969，15742]

4.3 问题三模型建立与求解

4.3.1 目标函数与变量约束

结合问题一的分析结果，明确了影响企业利润的主要因素，包括四型硅片销量（Q_k）、产品售价（P_k）、单晶方棒单价(G_k)与单耗（S_k）、用电单价（p_e）及电耗（e）、用水单价（p_w）及电耗（w）。在利润模型中，这些变量通过共同作用于企业的总利润，问题一利润计算模型如下：

$$\pi = \sum_{k=1}^{4} Q_k \times (P_k - G_k \times S_k - 0.02) - p_e \times e - p_w \times w - (755.76Q_1 + 1093.14Q_2 + 871.71Q_3 + 941.73Q_4)/10000 - 2973012$$

结合上述利润计算模型及问题二得出的重要影响因子变化区间，可建立如下目标—约束模型：

$$min \quad -\pi$$

$$s.t.\begin{cases} uQ_k = [9200000, 4800000, 7700000, 9700000] \\ lQ_k = [\,6300000, 4000000, 5700000, 8000000] \\ uP_k = [1.75, 1.69, 1.55, 1.36] \\ lP_k = \ [1.48, 1.58, 1.45, 1.21] \\ uG_k = [62, 62, 62, 52] \\ lG_k = [57, 57, 57, 47] \\ uS_k = \ [0.0142, 0.0187, 0.0162, 0.0208] \\ lS_k = [0.014, 0.0185, 0.016, 0.0207] \\ p_e \in [0.623，\ 0.625] \\ e \in [288397，\ 376789] \\ p_w \in [1.669,1.670] \\ w \in [7344，\ 10449] \end{cases}$$

其中 u 表示区间最大值；l表示区间最小值。

4.3.2 模型建立与参数设置

在明确问题三为多变量非线性优化问题的基础上，本文选用粒子群优化算法（Particle Swarm Optimization, PSO）作为求解方法。PSO 算法具有全局搜索能力强、参数较少、易于实现等优点，可用于求解利润最大化类的连续优化问题。算法通过模拟群体中粒子的协同进化行为，在解空间中不断迭代寻优，能够有效跳出局部最优，获得近似全局最优解[5]。主要数学模型理论如下：

在粒子群优化算法中，粒子当前位置即为它在当前搜索空间中的解，其位置会随着每次迭代而更新，更新过程基于当前速度和位置变化，位置更新公式为：

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

其中，x_i^{t+1}为粒子 i 在 t+1 时刻的位置，v_i^{t+1}为粒子 i 在 t+1 时刻的速度。粒子的速度决定了他在搜索空间中移动的方向和步长，其更新公式如下：

$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot \mathrm{rand}_1 \cdot (pBest_i - x_i^t) + c_2 \cdot \mathrm{rand}_2 \cdot (gBest - x_i^t)$$

其中：w：惯性权重；c_1：个体学习因子；c_2：社会学习因子；rand_1、rand_2：[0,1]之间的随机数。$pBest_i$：粒子 i 的个体最优位置；$gBest$：全局最优位置。

每个粒子在搜索过程中的最佳位置为个体的最优解，个体最优解是该例子历史上找到的最优解，主要基于目标函数计算，全局最优是所有例子群体找到的最优解。在构建优化模型时，将影响企业利润的关键因子作为粒子的位置变量输入到模型中，包括四型硅片的销量、产品售价、单晶方棒的单价与单耗、用电单价及耗电量、用水单价及耗水量，共设定 20 个决策变量，构成粒子在搜索空间中的维度。

为了提高粒子群算法的收敛速度与求解精度，本文采用网格搜索算法对 PSO

中的超参数进行系统性调优。主要调节参数包括个体学习因子、社会学习因子、惯性权重，通过设定多个候选值组合进行网格遍历，作为最优参数组合此外，为保证全局搜索能力的同时，兼顾算法的稳定性和收敛效率，粒子数设为 200，最大迭代次数设为 500。

4.3.2 模型求解

根据网络搜索算法得到个体学习因子$c_1 = 0.2$，社会学习因子$c_2 = 0.2$，惯性权重$w = 0.6$。

九月最大利润为 12546425.61 元，采用改进的粒子群算法各影响因子最佳组合为：

表 8 因子最佳组合

影响因子	最佳取值	影响因子	最佳取值
N1 售价	1.6998	N1 销量	9190115
N2 售价	1.6498	N2 销量	4504797
N3 售价	1.5997	N3 销量	7426559
P 售价	1.4499	P 销量	8953282
N1 单晶棒单价	57.8926	N1 单晶棒单耗	0.0141
N2 单晶棒单价	57.2827	N2 单晶棒单耗	0.0186
N3 单晶棒单价	57.5031	N3 单晶棒单耗	0.0160
P 单晶棒单价	47.9580	P 单晶棒单耗	0.0207
用电单价	0.5998	电耗	339790
用水单价	1.6663	水耗	10000

但本模型预测结果基于企业内部数据的理论值，企业产销策略及预测利润仍然会受到政策及市场等综合因素的影响，外部条件影响难以忽略。因此，亟待建立融合大语言模型、考虑市场政策变化的利润优化模型。

4.4 问题四模型建立与求解

4.4.1 市场、政策数据爬取

从这些指定的网站上爬取数据：中国政府网（www.gov.cn）、中国工业信息网（www.miit.gov.cn）、中国光伏行业协会（www.chinapv.org.cn）、仪器信息网（www.instrument.com.cn）、产业发展研究网（www.chinaidr.com），采用 python 爬虫，其中搜索的关键词为：晶硅、硅片（通过构造包含关键词的 URL 来实现按关键词搜索）。获取爬虫返回的数据（标题及链接）。返回获取的数据标题以及链接具体如下：

表 9 爬虫获取的数据标题以及链接

标题	链接
《国家发展改革委、工业和信息化部办公厅关于联合开展我国太阳能光伏产业总体情况调查的通知》	https://www.miit.gov.cn/jgsj/dzs/wjfb/art/2020/art_35c14d02ad794e82aea78cb74ed6cd0a.html
《新能源新材料生产企业专题对接活动成功举行》	https://www.miit.gov.cn/jgsj/zfs/cyzy/art/2020/art_cc50e46050f64b3b85a65710a0f3d6cf.html
《《光伏制造行业规范条件（2024 年本）》及管理办法解读》	https://www.miit.gov.cn/zwgk/zcjd/art/2024/art_2c28f1fbdfcd4bed93a11ecb77f4ae8c.html
《工业和信息化部等六部门关于推动能源电子产业发展的指导意见》	https://www.miit.gov.cn/zwgk/zcwj/wjfb/yj/art/2023/art_5fe76c58f263450ebc92c903427a6d12.html
《中国光伏行业标准报批公示》	https://www.chinapv.org.cn/Standardization/content_57.html
……	……

将获取到的页面文本逐个输入到 python 中实现 1）按照段落实现段落分句，2）匹配关键词 3）记录关键词所属的句子以及其位置，输出最终所属的句子。（每一个句子字符长度不超过 50 个字符），为方便后续的微调过程。经过处理后的句子如下表所示：

表 10 文本处理

处理后句子
硅片因阶段性供不应求，企业开工率提升（75%-90%），但全年产能过剩预期压制销量增长潜力。
短期供需紧张推动 M10 硅片价格周环比涨幅 26.4%（6.22 元/片），但全年过剩趋势限制积极情绪。
硅料价格随硅片需求短期上涨，但长期供应过剩预期导致硅棒成本波动性较强。
将新建单晶硅光伏电池、组件项目平均效率指标分别由 23%、20%提升至新建 P 型电池
光伏度电成本降低 6.8%。现在，正瞄准半导体单晶硅国产化进行科技攻关。
太阳能工业用超白玻璃、多晶硅、单晶硅等产品产量分别增长 63.3%、84.6%和 79.9%。
为促进国产多晶硅料“量”与“质”同步提升，推动产业链上下游对接，更好满足我国

处理后句子
光伏和半导体产业快速发展需求。 第 22 组件晶硅回收高效技术的研究 ……

4.4.2 数据标注与微调

针对于不同的变量（销量、售价、单晶方棒成本、水电费等）其数据标注也不相同，其中数据标准是指该文本所蕴含的情感类别以及情感的强烈程度，例如消极标注为-1，中性标准为 0，积极标注为 1，数据标注完成之后需要输入到 Bert 模型中实现情感类别的分类以及情感分数评价的模型参数**微调**，情感分数取值为[-1,1]。-1 至-0.6 为极度消极，-0.6 至-0.2 为略微消极，-0.2 至 0.2 为中性，0.6 至 1 为极度积极，0.2 至 0.6 为略微消极。Bert 的情感分数评价如下所示：

表 11 文本处理

文本	各变量情感分数				
序号	销量	售价	单晶方棒单价	理论单耗	水电费
01	-0.3	0.5	0.1	0.4	0
02	-0.7	-0.8	-0.6	0.3	-0.2
03	-0.4	-0.5	0.0	0.7	0.2
04	-0.6	-0.9	-0.3	0.5	-0.1
05	-0.5	-0.4	0.2	0.8	0.3
……	……	……	……	……	……

4.4.2 Bert 模型输出及情感得分量化

BERT（Bidirectional Encoder Representations from Transformers）是一种基于 Transformer 架构的预训练语言模型，通过双向编码器同时学习上下文信息，能够更全面理解文本语义。其具有计算参数相对较少（110 M 参数）、易于本地部署和计算高效等优点，适合在算力资源有限的环境中运行。同时，其预训练与微调机制能够快速适应具体任务，减少开发成本，提升应用灵活性与实用性。在下游任务中，BERT 可通过微调适应分类、问答、情感分析等多种自然语言处理任务，具有优越的泛化能力与表现效果[6]。其模型架构如图 9。

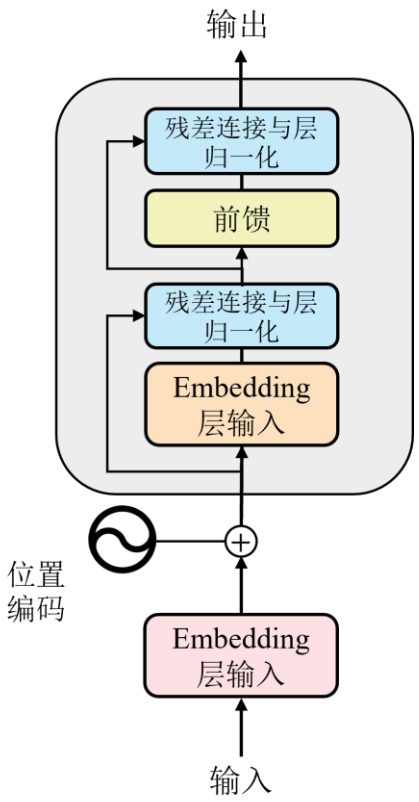


图 9 Bert 模型架构

经过微调后的 Bert 模型输入个人拟定的文本（测试集）：“终端需求疲软叠加库存上升，晶硅片销量明显放缓”、“部分上游材料短缺推高价格，晶硅产品报价再创新高”“供需错位引发结构性涨价，销售节奏受影响”、“晶硅产品价格持续上涨，下游企业成本压力倍增”。

情感得分的量化需要耦合到问题三改进的粒子群算法的变量约束中，参考房地产企业的信心指数量化方案，其量化指数∂取值（0.1，0.3），由于新兴市场（如中国）的∂可能高于成熟市场（如美国），且高频数据（如月度）的∂通常小于低频数据（如年度）[7]，则∂的取值做居中处理取值为 0.2 建立量化指标方程：

$$P_t = P_0 \cdot (1 + \partial \cdot S_t)$$

其中P_t是预测数据；P_0是历史数据（一般取过往三个月的数据的均值）；S_t为情感得分。

基于表 11 得出的各个变量的情感得分状况取均值，然后基于量化指标方程进行量化处理，各个变量的量化后的平均情感得分（保留三位小数）以及各个变量的取值依次为：

销量：

$$\alpha = ave(S_t^{(1)}) = \partial \cdot \sum_{i=1}^{n} S_{ti}^{(1)} / n = 0.042$$

$$Q_k' = (1+\alpha)Q_k$$

售价：

$$\beta = ave(S_t^{(2)}) = \partial \cdot \sum_{i=1}^{n} S_{ti}^{(2)} / n = -0.015$$

$$P_k' = (1+\beta)P_k$$

单晶方棒价格：

$$\gamma = ave(S_t^{(3)}) = \partial \cdot \sum_{i=1}^{n} S_{ti}^{(3)} / n = 0.008$$

$$G_k' = (1+\gamma)G_k$$

理论单耗：

$$\delta = ave(S_t^{(4)}) = \partial \cdot \sum_{i=1}^{n} S_{ti}^{(4)} / n = -0.026$$

$$S_k' = (1+\delta)S_k$$

水电费：

$$\varepsilon = ave(S_t^{(5)}) = \partial \cdot \sum_{i=1}^{n} S_{ti}^{(5)} / n = 0.033$$

$$ew = (1+\varepsilon)(p_e \times e + p_w \times w)$$

新的目标函数（上下界更新为原上下界的值乘对应参数）为：

$$\pi = \sum_{k=1}^{4} Q_k' \times (P_k' - G_k' \times S_k' - 0.02) - ew - (755.76Q_1' + 1093.14Q_2' + 871.71Q_3' + 941.73Q_4')/10000 - 2973012$$

4.4.3 融合 Bert 模型的产销计划输出

经过 Bert 耦合后的新目标函数和约束条件，采用改进的 PSO 粒子群算法实现耦合非结构化数据的最优产销计划：最大利润为 11733734.73，此时各因子最佳组合为：

表 12 融合非结构化数据因子组合

影响因子	最佳取值	影响因子	最佳取值
N1 售价	1.6346	N1 销量	9191464
N2 售价	1.6039	N2 销量	4738438
N3 售价	1.5674	N3 销量	7897040

影响因子	最佳取值	影响因子	最佳取值
P 售价	1.3901	P 销量	9398603
N1 单晶棒单价	57.4344	N1 单晶棒单耗	0.0140
N2 单晶棒单价	58.0198	N2 单晶棒单耗	0.0183
N3 单晶棒单价	59.1610	N3 单晶棒单耗	0.0160
P 单晶棒单价	47.4142	P 单晶棒单耗	0.0200
用电单价	0.6140	电耗	356677
用水单价	1.6878	水耗	9788

考虑到测试集反映出的政策和市场趋势整体趋于负面，可能对企业利润产生压制作用。融合 BERT 模型的优化结果中，利润较基于历史数据的模型略低，但更符合当前复杂市场环境下的实际走势。在关键变量“售价上升、销量下降”的背景下，融合 BERT 后的预测模型捕捉到了政策紧缩、市场需求疲软等外部非结构化信息对企业销售产生的抑制效应。相比之下，仅基于历史数据构建的模型可能高估销量表现，导致利润预期偏乐观。因此，融合 BERT 模型后所得结果更具现实针对性和风险敏感性，能够为企业提供更加稳健、前瞻的生产与营销策略建议。

5 模型评价

5.1 模型优点

优点一：利润计算模型在结构上仍基于收入与成本的传统核算框架，逻辑清晰，便于理解与实际应用。

优点二：通过引入弹性分析方法，系统评估各因素对利润的边际影响，有助于科学筛选关键影响因子，提高建模的针对性与解释力。

优点三：采用网格搜索算法对粒子群优化模型的超参数进行系统调优，有效避免了主观设定带来的不确定性，提升了模型的鲁棒性与优化效果。

5.2 模型缺点

缺点一：利润计算模型涉及变量较多、结构较为复杂，增加了模型求解的计算负担。

缺点二：模型难以克服粒子群算法的固有局限性，仍存在易陷入局部最优、对初始粒子分布较为敏感等问题，导致每次运行结果存在一定波动。

参考文献

[1] 王友钱.边际成本在财务管理中的应用[J].中国产经,2024,(24):173-175.

[2] 蒋士藩,张颖杰,陆娟,等.基于 ARIMA 模型预测 ICU 耐碳青霉烯类肺炎克雷伯菌流行趋势[J].中华医院感染学杂志,2025,35(06):933-938.

[3] 孙倩,胡军国,刘玉琴,等.基于灰色预测模型 GM(1,1)的 2010-2025 年甘肃省肿瘤登记地区胰腺癌发病率预测[J].中华肿瘤防治杂志,2025,32(06):323-328.

[4] 杨国华,郑豪丰,张鸿皓,等.基于 Holt-Winters 指数平滑和时间卷积网络的短期负荷预测[J].电力系统自动化,2022,46(06):73-82.

[5] 郁旸.改进粒子群算法在能耗预测中的应用[J].计算机时代,2023,(07):42-45.

[6] 翁克瑞,周雅洁,於世为. 基于 BERT 的多层次特征融合的舆情文本政策意愿识别模型研究[J]. 中国地质大学学报（社会科学版）,2025,25(1):131-140.

[7] 财政部财政科学研究所课题组. 宏观经济形势与财政调控:从短期到中长期的分析认识[J]. 经济研究参考,2012(61):3-50.

附录

附录 A 灵敏度分析

```
import matplotlib.pyplot as plt
import numpy as np
import os

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

QN1 = 8300000
PriceN1 = 2.3
QN2 = 3300000
PriceN2 = 2.11
QN3 = 6500000
PriceN3 = 2.11
QP = 4150000
PriceP = 2.1
DjcostN1 = 110
LlN1 = 0.014406699
```

```
DjcostN2 = 110
LlN2 = 0.018981899
DjcostN3 = 110
LlN3 = 0.01643184
DjcostP = 95
LlP = 0.021289091
EC = 0.634842018
EuN1 = 400
EuN2 = 536.68
EuN3 = 520.4
EuP = 520.4
EU = 350000
Wprice = 1.669726392
Wuse = 10000
DdjgPrice = 19.05181962
DdjguseN1 = 34.96518424
DdjguseN2 = 50
DdjguseN3 = 43
DdjguseP = 47.73926812
WaterqgPrice = 18.24714976
WaterqguseN1 = 8.111922744
WaterqguseN2 = 13.94181122
WaterqguseN3 = 8.111922744
WaterqguseP = 8.111922744
MwjdPrice = 1.85
Mwjduse = 201.8225185
YichunPrice = 8.428318584
Yichunuse = 277.5059629
TubuPrice = 3783.185841
Tubuuse = 3
KaicaoPrice = 862.8318584
Kaicaouse = 13

def equation(QN1, PriceN1, QN2, PriceN2, QN3, PriceN3, QP, PriceP,
                    DjcostN1, LlN1, DjcostN2, LlN2, DjcostN3, LlN3, DjcostP, LlP,
                    EC, EuN1, EuN2, EuN3, EuP, EU,
                    Wprice, Wuse, DdjgPrice, DdjguseN1, DdjguseN2, DdjguseN3, DdjguseP,
                    WaterqgPrice, WaterqguseN1, WaterqguseN2, WaterqguseN3, WaterqguseP,
                    MwjdPrice, Mwjduse, YichunPrice, Yichunuse, TubuPrice, Tubuuse,
                    KaicaoPrice, Kaicaouse):
    income = (QN1 * PriceN1 + QN2 * PriceN2 + QN3 * PriceN3 + QP * PriceP)
    cost1 = (DjcostN1 * LlN1 * QN1 + DjcostN2 * LlN2 * QN2 + DjcostN3 * LlN3 * QN3 +
DjcostP * LlP * QP)
    cost2 = EC * (EuN1 + EuN2 + EuN3 + EuP + EU)
```

```
    cost3 = Wprice * Wuse
    cost4 = (DdjgPrice * DdjguseN1 * QN1/10000 + DdjgPrice * DdjguseN2 * QN2 /10000+
DdjgPrice * DdjguseN3 * QN3 /10000+    DdjgPrice * DdjguseP * QP/10000)
    cost5 = WaterqgPrice * (WaterqguseN1 * QN1 /10000+ WaterqguseN2 * QN2/10000 +
WaterqguseN3 * QN3/10000    + WaterqguseP * QP/10000)
    cost6 = MwjdPrice * Mwjduse
    cost7 = YichunPrice * Yichunuse
    cost8 = TubuPrice * Tubuuse
    cost9 = KaicaoPrice * Kaicaouse
    cost10 = (QN1+QN2+QN3+QP)*0.04

    return income - cost1 - cost2 - cost3 - cost4 - cost5 - cost6 - cost7 - cost8 - cost9 - cost10

def calculate_all_elasticity():
    variables = [QN1, PriceN1, QN2, PriceN2, QN3, PriceN3, QP, PriceP,
                 DjcostN1, LlN1, DjcostN2, LlN2, DjcostN3, LlN3, DjcostP, LlP,
                 EC, EuN1, EuN2, EuN3, EuP, EU,
                 Wprice, Wuse, DdjgPrice, DdjguseN1, DdjguseN2, DdjguseN3,
DdjguseP,
                 WaterqgPrice, WaterqguseN1, WaterqguseN2, WaterqguseN3,
WaterqguseP,
                 MwjdPrice, Mwjduse, YichunPrice, Yichunuse, TubuPrice, Tubuuse,
                 KaicaoPrice, Kaicaouse]
    variable_names = ['销量 N1', '售价 N1', '销量 N2', '售价 N2', '销量 N3', '售价 N3', '销量
P', '售价 P',
                      '单晶棒成本 N1', '理论单耗 N1', '单晶棒成本 N2', '理论单耗 N2',
'单晶棒成本 N3', '理论单耗 N3', '单晶棒成本 P', '理论单耗 P',
                      '电费', '电量 N1', '电量 N2', '电量 N3', '电量 P', '电耗',
                      '水单价', '用水量', '电镀金刚价格', '电镀金刚用量 N1', '电镀金刚
用量 N2', '电镀金刚用量 N3', '电镀金刚用量 P',
                      '水性切割液价格', '水性切割液用量 N1', '水性切割液用量 N2', '
水性切割液用量 N3', '水性切割液用量 P',
                      '美纹胶带价格', '美纹胶带用量', '乙醇价格', '乙醇用量', '涂布价
格', '涂布用量',
                      '开槽价格', '开槽用量']
    elasticity_coefficients = []

    base_y = equation(*variables)

    for i, var in enumerate(variables):
        new_var = var * 1.01
        new_variables = variables.copy()
        new_variables[i] = new_var
        new_y = equation(*new_variables)
```

```
            percentage_change_y = (new_y - base_y) / base_y
            percentage_change_var = 0.01
            elasticity = percentage_change_y / percentage_change_var
            elasticity_coefficients.append(elasticity)

        abs_elasticity = np.abs(elasticity_coefficients)
        sorted_indices = np.argsort(abs_elasticity)[::-1]
        sorted_names = np.array(variable_names)[sorted_indices]
        sorted_elasticity = np.array(elasticity_coefficients)[sorted_indices]

        print("所有变量的价格弹性系数（按绝对值从高到低排序）：")
        for name, elasticity in zip(sorted_names, sorted_elasticity):
            print(f"{name}: {elasticity}")

        plt.figure(figsize=(20, 8))
        plt.bar(sorted_names, sorted_elasticity, color='b')
        plt.xlabel('变量')
        plt.ylabel('价格弹性系数')
        plt.title('价格弹性系数排序')
        plt.xticks(rotation=90)
        plt.grid(False)
        plt.tight_layout()

        x_position = 19.5
        y_min, y_max = plt.ylim()
        plt.plot([x_position, x_position], [y_min, y_max], 'r--')

        plt.savefig('jiage', dpi=300)
        plt.show()

        return elasticity_coefficients

calculate_all_elasticity()
```

附录 B 利润计算器

```
def get_float_input(prompt):
    while True:
        try:
            return float(input(prompt))
        except ValueError:
```

```
                print("输入错误，请输入数字！")

def get_list_input(prompt, length=4):
    while True:
        try:
            values = [float(x.strip()) for x in input(prompt).split(",")]
            if len(values) != length:
                print(f"需要输入正好{length}个数值，请重新输入！")
                continue
            return values
        except ValueError:
            print("格式错误，请按 数值,数值,数值,数值 的格式输入！")

def main():
    print("\n 硅片利润计算系统")
    print("当前固定支持 4 种硅片类型\n")

    print("【请输入 4 种硅片数据】")
    Qk = get_list_input("各类销量（Q1-Q4，用逗号分隔）：")
    Pk = get_list_input("销售单价（P1-P4，用逗号分隔）：")
    Gk = get_list_input("硅棒单价（G1-G4，用逗号分隔）：")
    Sk = get_list_input("理论单耗（S1-S4，用逗号分隔）：")

    print("\n【请输入水电数据】")
    pe = get_float_input("电费单价：")
    e = get_float_input("总用电量：")
    pw = get_float_input("水费单价：")
    w = get_float_input("总用水量：")

    income = sum(q * p for q, p in zip(Qk, Pk))
    silicon_cost = sum(q * g * s    for q, g, s in zip(Qk, Gk, Sk))
    electricity_cost = pe * e
    water_cost = pw * w
    extra_cost = sum(q * 0.02 for q in Qk)
    fixed_cost = 2973012

    additional_costs = (
            Qk[0] * 755.76 +
            Qk[1] * 1093.14 +
            Qk[2] * 871.71 +
            Qk[3] * 941.73
    )/10000

    total_profit = income - silicon_cost - electricity_cost - water_cost \
```

```
- extra_cost - fixed_cost - additional_costs

    # 打印详细报表
    print("\n 利润计算报表 ")
    print(f"{'收入':<20}{income:>15.2f}")
    print(f"{'硅棒成本':<20}{silicon_cost:>15.2f}")
    print(f"{'电费成本':<20}{electricity_cost:>15.2f}")
    print(f"{'水费成本':<20}{water_cost:>15.2f}")
    print(f"{'额外成本(2%)':<20}{extra_cost:>15.2f}")
    print(f"{'固定成本':<20}{fixed_cost:>15.2f}")
    print(f"{'附加硅片成本':<20}{additional_costs:>15.2f}")
    print("-" * 35)
    print(f"{'最终利润':<20}{total_profit:>15.2f}")
    print("=" * 35)

if __name__ == "__main__":
    main()
```

附录 C 灰度预测模型

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

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

def gm11(x0):
    x1 = np.cumsum(x0)
    n = len(x0)
    z = (x1[:-1] + x1[1:]) / 2
    B = np.vstack([-z, np.ones(n - 1)]).T
    Y = x0[1:].reshape((-1, 1))
    # 最小二乘法求解参数
    ab = np.linalg.inv(B.T @ B) @ B.T @ Y
    a = ab[0, 0]
    b = ab[1, 0]
    # 预测累加序列
    x1_hat = lambda k: (x0[0] - b / a) * np.exp(-a * k) + b / a
    x0_hat = np.zeros(n)
    x0_hat[0] = x0[0]
    for i in range(1, n):
        x0_hat[i] = x1_hat(i) - x1_hat(i - 1)
```

```
    next_value = x1_hat(n) - x1_hat(n - 1)
    return next_value, x0_hat, a, b

def accuracy_test(x0, x0_hat):
    e = x0 - x0_hat
    relative_error = np.abs(e / x0)
    avg_relative_error = np.mean(relative_error)

    s1 = np.std(x0)
    s2 = np.std(e)
    c = s2 / s1
    p = np.sum(np.abs(e - np.mean(e)) < 0.6745 * s1) / len(e)
    return avg_relative_error, c, p

def residual_normality_test(e):
    _, p = stats.shapiro(e)
    return p

def main():
    x0 = np.array([0.014406699, 0.014377357, 0.014392013, 0.014348135, 0.014333568,
0.014283379, 0.014214782, 0.014200365])

    next_value, x0_hat, a, b = gm11(x0)
    print(f"第九个值的预测值: {next_value:.5f}")

    avg_relative_error, c, p = accuracy_test(x0, x0_hat)
    print(f"平均相对误差: {avg_relative_error * 100:.2f}%")
    print(f"后验差比值: {c:.2f}")
    print(f"小误差概率: {p:.2f}")

    e = x0 - x0_hat
    p_value = residual_normality_test(e)
    print(f"残差正态性检验 p 值: {p_value:.2f}")
    if p_value < 0.05:
        print("残差不服从正态分布，不适用残差预估区间。")
        error_range = np.abs(e).max()
        lower_bound = next_value - error_range
        upper_bound = next_value + error_range
    else:
        print("残差服从正态分布，可以使用残差预估区间。")
        lower_bound = next_value - np.std(e)
        upper_bound = next_value + np.std(e)

    print(f"预测区间: [{lower_bound:.2f}, {upper_bound:.2f}]")
```

```
    plt.figure(figsize=(10, 6))
    plt.plot(np.arange(1, len(x0) + 1), x0, 'bo-', label='真实值')
    plt.plot(np.arange(1, len(x0) + 1), x0_hat, 'rx-', label='拟合值')
    plt.plot(len(x0) + 1, next_value, 'g*', label='预测值')

    plt.plot([len(x0) + 1, len(x0) + 1], [lower_bound, upper_bound], 'm--', label='预测区间')

    plt.xlabel('月份')
    plt.ylabel('数据值')
    plt.title('真实值、拟合值、预测值及预测区间')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    main()
```

附录 D 指数平滑预测模型

```
import numpy as np
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
import matplotlib as mpl

mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['axes.unicode_minus'] = False

data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug'],
    'Production': [8300000, 6000000, 5000000, 3300000, 8000000, 7800000, 7500000,
7600000]
}
df = pd.DataFrame(data)

alpha_values = np.arange(0.01, 1, 0.01)
beta_values = np.arange(0.01, 1, 0.01)

best_model = None
best_alpha = None
best_beta = None
lowest_error = float('inf')
```

```
def calculate_mape(model_fit, actual_values):
    forecast_values = model_fit.fittedvalues
    mape = np.mean(np.abs((actual_values - forecast_values) / actual_values)) * 100
    return mape

for alpha in alpha_values:
    for beta in beta_values:
        try:
            model = ExponentialSmoothing(
                df['Production'],
                trend='add',
                seasonal=None,
                initialization_method='estimated'
            )
            model_fit = model.fit(smoothing_level=alpha, smoothing_trend=beta)
            mape = calculate_mape(model_fit, df['Production'])

            if mape < lowest_error:
                lowest_error = mape
                best_alpha = alpha
                best_beta = beta
                best_model = model_fit
        except:
            continue
if best_model is not None:
    print(f"最佳参数: Alpha={best_alpha:.2f}, Beta={best_beta:.2f}")
    print(f"最低 MAPE: {lowest_error:.2f}%\n")

    fitted_values = best_model.fittedvalues
    error_percentages = []
    print("各月预测误差百分比:")
    for idx, (actual, pred) in enumerate(zip(df['Production'], fitted_values)):
        error_pct = ((actual - pred) / actual) * 100
        error_percentages.append(error_pct)
        print(f"{df['Month'][idx]}: {error_pct:.2f}%")

    mae_pct = np.mean(np.abs(error_percentages))
    print(f"\n 平均绝对误差百分比(MAE): {mae_pct:.2f}%")

    forecast = best_model.forecast(steps=1)   # 预测 9 月
    forecast_sep = forecast.iloc[0]

    try:
```

```
            forecast_obj = best_model.get_forecast(steps=1)
            ci = forecast_obj.conf_int().iloc[0]   # 获取 9 月的置信区间
            ci_method = "模型置信区间(95%)"
        except:
            residuals = df['Production'] - fitted_values
            std_dev = np.std(residuals)
            ci = (forecast_sep – 1.96 * std_dev, forecast_sep + 1.96 * std_dev)
            ci_method = "残差分析区间(95%)"

        print(f"\n9 月预测结果:")
        print(f"预测值: {forecast_sep:,.0f}")
        print(f"{ci_method}: [{ci[0]:,.0f} ~ {ci[1]:,.0f}]")

        plt.figure(figsize=(12, 6), dpi=100)

        plt.plot(df['Month'], df['Production'], 'bo-', label='实际值（1-8 月）')

        plt.plot(df['Month'], fitted_values, 'rx--', label='模型拟合值')
        pred_months = ['Sep']
        plt.plot(pred_months, forecast, 'gx--', markersize=8, label='预测值')

        plt.plot(['Sep', 'Sep'], [ci[0], ci[1]], 'r--', label='置信区间(95%)')   # 置信区间竖直显示

        plt.title('N1 晶硅片生产量预测分析（单位：片）', fontsize=14)
        plt.xlabel('月份', fontsize=12)
        plt.ylabel('N1 晶硅片生产量', fontsize=12)
        plt.legend()
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()
    else:
        print("未找到有效模型")
```

附录 E 改进的粒子群算法

```
import numpy as np
import pyswarms as ps

def objective_function(x):
    Q = x[:, 0:4]
    G = x[:, 4:8]
    P = x[:, 8:12]
    S = x[:, 12:16]
```

```
    pe = x[:, 16]
    pw = x[:, 17]
    w = x[:, 18]
    e = x[:, 19]
    profit = 0
    for k in range(4):
        term = Q[:, k] * (P[:, k] - G[:, k] * S[:, k] - 0.02)
        profit += term
    cost_Q = (755.76 * Q[:, 0] + 1093.14 * Q[:, 1] + 871.71 * Q[:, 2] + 941.73 * Q[:,
3])/10000
    cost_ew = pe * e + pw * w
    cost_ee = pe * (0.04* Q[:, 0]+ 0.053668 * Q[:, 1] + 0.05204 * Q[:, 2] + 0.05204 * Q[:, 3])
    profit -= cost_Q
    profit -= cost_ew
    profit -= cost_ee
    profit -= 2973012

    return -profit

ub = [
    9200000, 4800000, 7700000, 9700000,   # Q1-Q4
    62, 62, 62, 52,   # G1-G4
    1.75, 1.69, 1.55, 1.36,   # P1-P4
    0.0142, 0.0187, 0.0162, 0.0208,   # S1-S4
    0.625,   # pe
    1.670,   # pw
    10449,   # w
    376789   # e
]
lb = [
    6300000, 4000000, 5800000, 8000000,   # Q1-Q4
    57, 57, 57, 47,   # G1-G4
    1.48, 1.58, 1.45, 1.36,   # P1-P4
    0.014, 0.0185, 0.016, 0.0207,   # S1-S4
    0.623,   # pe
    1.669,   # pw
    7344,   # w
    288397   # e
]

assert len(ub) == 20, "变量下限范围数量不匹配"
assert len(lb) == 20, "变量上限范围数量不匹配"

options = {'c1': 0.2, 'c2': 0.2, 'w': 0.6}
```

```
optimizer = ps.single.GlobalBestPSO(
        n_particles=200,
        dimensions=20,
        options=options,
        bounds=(lb, ub)
)
best_cost, best_pos = optimizer.optimize(objective_function, iters=500)

max_profit = -best_cost

print(f"最大利润: {max_profit:.2f}")
print("最优变量值:")
print(f"Q1-Q4: {best_pos[0:4].astype(int)}")
print(f"G1-G4: {best_pos[4:8].round(5)}")
print(f"P1-P4: {best_pos[8:12].round(5)}")
print(f"S1-S4: {best_pos[12:16].round(5)}")
print(f"pe: {best_pos[16]:.4f}, pw: {best_pos[17]:.4f}, w: {best_pos[18]:.4f}, e:
{best_pos[19]:.4f}")
```

附录 F 爬虫代码

```
import os.path
import requests # 发送请求库
import urllib.request, urllib.error   # 制定 URL，获取网页数据
from bs4 import BeautifulSoup     # 解析页面
import pandas as pd   # 存入 csv 数据
from time import sleep   # 等待间隔
import random     # 随机库
import re     # 用正则表达式提取 url


# 浏览器请求头
my_headers = [
      "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/39.0.2171.95 Safari/537.36 OPR/26.0.1656.60",
      "Opera/8.0 (Windows NT 5.1; U; en)",
      "Mozilla/5.0 (Windows NT 5.1; U; en; rv:1.8.1) Gecko/20061208 Firefox/2.0.0 Opera 9.50",
      "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; en) Opera 9.50",
      "Opera/9.80 (Macintosh; Intel Mac OS X 10.6.8; U; en) Presto/2.8.131 Version/11.11",
      "Opera/9.80 (Windows NT 6.1; U; en) Presto/2.8.131 Version/11.11",
      "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0",
      "Mozilla/5.0 (X11; U; Linux x86_64; zh-CN; rv:1.9.2.10) Gecko/20100922 Ubuntu/10.10
(maverick) Firefox/3.6.10",
      "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv,2.0.1) Gecko/20100101 Firefox/4.0.1",
```

```
    "Mozilla/5.0 (Windows NT 6.1; rv,2.0.1) Gecko/20100101 Firefox/4.0.1",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.57.2 (KHTML, like Gecko)
Version/5.1.7 Safari/534.57.2",
    "Mozilla/5.0 (iPad; U; CPU OS 4_3_3 like Mac OS X; en-us) AppleWebKit/533.17.9
(KHTML, like Gecko) Version/5.0.2 Mobile/8J2 Safari/6533.18.5",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/39.0.2171.71 Safari/537.36",
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.11 (KHTML, like Gecko)
Chrome/23.0.1271.64 Safari/537.11",
    "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.16 (KHTML, like
Gecko) Chrome/10.0.648.133 Safari/534.16",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_0) AppleWebKit/535.11 (KHTML, like
Gecko) Chrome/17.0.963.56 Safari/535.11",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/30.0.1599.101 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; 360SE)",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.11 (KHTML, like Gecko)
Chrome/20.0.1132.11 TaoBrowser/2.0 Safari/536.11",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko)
Chrome/21.0.1180.71 Safari/537.1 LBBROWSER",
    "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC
6.0; .NET4.0C; .NET4.0E; LBBROWSER)",
    "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; QQDownload
732; .NET4.0C; .NET4.0E; LBBROWSER)",
    "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC
6.0; .NET4.0C; .NET4.0E; QQBrowser/7.0.3698.400)",
    "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; QQDownload
732; .NET4.0C; .NET4.0E)",
    "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.84 Safari/535.11 SE 2.X MetaSr 1.0",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; SV1; QQDownload
732; .NET4.0C; .NET4.0E; SE 2.X MetaSr 1.0)",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; SE 2.X MetaSr 1.0; SE
2.X MetaSr 1.0; .NET CLR 2.0.50727; SE 2.X MetaSr 1.0)",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Maxthon/4.4.3.4000 Chrome/30.0.1599.101 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_0) AppleWebKit/535.11 (KHTML, like
Gecko) Chrome/17.0.963.56 Safari/535.11",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/38.0.2125.122 UBrowser/4.0.3214.0 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
Chrome/55.0.2883.87 UBrowser/6.2.4094.1 Safari/537.36",
    "Mozilla/5.0 (iPad; U; CPU OS 4_2_1 like Mac OS X; zh-cn) AppleWebKit/533.17.9
(KHTML, like Gecko) Version/5.0.2 Mobile/8C148 Safari/6533.18.5",
    "Mozilla/5.0 (iPad; U; CPU OS 4_3_3 like Mac OS X; en-us) AppleWebKit/533.17.9
(KHTML, like Gecko) Version/5.0.2 Mobile/8J2 Safari/6533.18.5",
    "Mozilla/5.0 (hp-tablet; Linux; hpwOS/3.0.0; U; en-US) AppleWebKit/534.6 (KHTML, like
Gecko) wOSBrowser/233.70 Safari/534.6 TouchPad/1.0",
    "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)",
    "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0)",
    "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; TencentTraveler 4.0)",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Avant Browser)",
    "Mozilla/5.0 (SymbianOS/9.4; Series60/5.0 NokiaN97-1/20.0.019; Profile/MIDP-2.1
Configuration/CLDC-1.1) AppleWebKit/525 (KHTML, like Gecko) BrowserNG/7.1.18124",
    "Openwave/ UCWEB7.0.2.37/28/999",
]

def search_in_360(v_keyword, v_result_file, v_start_page, v_max_page):

    # 获取每页搜索结果
    for page in range(v_start_page, v_max_page):
        print('开始爬取第{}页'.format(page + 1))
        wait_seconds = random.uniform(1, 3)  # 等待时长，秒
        print('开始等待{}秒'.format(wait_seconds))
        sleep(wait_seconds)  # 随机等待

        # 伪装成浏览器
        User_Agent = random.choice(my_headers)
        print(User_Agent)
        headers = {
            "User-Agent": User_Agent,
            }

        # 生成每页 url
        url = 'https://www.gov.cn/zhengce/zhengceku/s?q=' + v_keyword +'&pn='+str(page+1)
        r = requests.get(url, headers=headers)
        # print(type(r))  # 检验 r 的是否为 response 类型
        print('相应码是：{}'.format(r.status_code))
        # print(type(r.status_code))
        # 如果状态码不是 200，重新选择 User_Agent 尝试
        if r.status_code != 200:
            try:
                User_Agent = random.choice(my_headers)
```

```
                print(User_Agent)
                headers = {
                    "User-Agent": User_Agent,
                }
                r = requests.get(url, headers=headers)
                print('相应码是：{}'.format(r.status_code))
            except:
                User_Agent = random.choice(my_headers)
                print(User_Agent)
                headers = {
                    "User-Agent": User_Agent,
                }
                r = requests.get(url, headers=headers)
                print('相应码是：{}'.format(r.status_code))

        html = r.text   # 用 text 方法获得 HTML
        print(r.request.url)
        # print(r.text)
        soup = BeautifulSoup(html,'html.parser')
        result_list = soup.find_all(class_='res-list')
        print('正在提取：{}，共查询到{}个结果'.format(url, len(result_list)))

        # 如果提取结果数量为0，重新选择尝试
        if result_list == 0:
            try:
                User_Agent = random.choice(my_headers)
                print(User_Agent)
                headers = {
                    "User-Agent": User_Agent,
                }
                r = requests.get(url, headers=headers)
                print('相应码是：{}'.format(r.status_code))
                html = r.text   # 用 text 方法获得 HTML
                print(r.request.url)
                # print(r.text)
                soup = BeautifulSoup(html, 'html.parser')
                result_list = soup.find_all(class_='res-list')
                print('正在提取：{}，共查询到{}个结果'.format(url, len(result_list)))
            except:
                User_Agent = random.choice(my_headers)
                print(User_Agent)
                headers = {
                    "User-Agent": User_Agent,
                }
```

```
            r = requests.get(url, headers=headers)
            print('相应码是：{}'.format(r.status_code))
            html = r.text   # 用 text 方法获得 HTML
            print(r.request.url)
            # print(r.text)
            soup = BeautifulSoup(html, 'html.parser')
            result_list = soup.find_all(class_='res-list')
            print('正在提取：{}，共查询到{}个结果'.format(url, len(result_list)))


kw_list=[]  # 关键字
page_list=[]    # 页码
title_list=[]   # 标题
href_list=[]
real_url_list=[]    # 真正的链接
desc_list=[]    # 简介
site_list=[]    # 网站名称
content_list=[]
for result in result_list:
    # find 函数是找到第一个值,text 函数是文本内容
    title = result.find('a').text
    print('title is：',title)
    href = result.find('a')['href']
    # real_url = get_real_url(v_url=href)
    # 简介和网站名称可能为空
    try:
        desc=result.p.text
    except:
        desc = ""
    content = result.text
    # 结果集合
    kw_list.append(v_keyword)
    page_list.append(page+1)
    title_list.append(title)
    href_list.append(href)
    # real_url_list.append(real_url)
    desc_list.append(desc)
    content_list.append(content)
    # site_list.append(site)
    # print(title_list)
df = pd.DataFrame(
    {
        '关键词': kw_list,
        '页码': page_list,
```

```
                    '标题': title_list,
                    '360 链接': href_list,
                    # '真实链接': real_url_list,
                    '简介': desc_list,
                    '内容': content_list,
                    # '网站名称':site_list,
                }

        )
        # print(df)
        if os.path.exists(v_result_file):
            header = None
        else:
            header = ['关键词','页码','标题','360 链接','简介','内容']      # csv 文件头
        df.to_csv(v_result_file, mode='a+', index=False, header=header, encoding='utf-8_sig')
        print('结果保存成功：{}'.format(v_result_file))


if __name__ == "__main__":  # 当程序执行时
    search_keyword = '晶硅'      # 搜索的关键词
    # 定义搜索页面
    start_page = 0
    max_page = 20
    result_file = '爬取_{}_前{}页-{}页.csv'.format(search_keyword, start_page+1, max_page)
    # 如果文件存在，先删除
    if os.path.exists(result_file):
        os.remove(result_file)
        print('结果文件{}已存在，已删除'.format(result_file))
    # 开始爬取
    search_in_360(v_keyword=search_keyword, v_result_file=result_file, v_start_page=
start_page, v_max_page= max_page)
```

附录 G Bert 模型代码

```
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import BertTokenizer, BertForSequenceClassification, AdamW
from sklearn.model_selection import train_test_split
import pandas as pd

# 定义数据集类
class CrystallineSiliconDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length):
        self.texts = texts
```

```
            self.labels = labels
            self.tokenizer = tokenizer
            self.max_length = max_length

        def __len__(self):
            return len(self.texts)

        def __getitem__(self, idx):
            text = str(self.texts[idx])
            label = self.labels[idx]
            encoding = self.tokenizer.encode_plus(
                text,
                add_special_tokens=True,
                max_length=self.max_length,
                padding='max_length',
                truncation=True,
                return_tensors='pt'
            )
            return {
                'input_ids': encoding['input_ids'].flatten(),
                'attention_mask': encoding['attention_mask'].flatten(),
                'labels': torch.tensor(label, dtype=torch.long)
            }

# 加载预训练的中文 BERT 模型和分词器
model_name = 'bert-base-chinese'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2)

# 载入包含晶硅销量的数据
try:
    df = pd.read_csv(r'C:\Users\15719\Desktop\华中杯数学建模\lateset.csv')
except FileNotFoundError:
    print("未找到指定的 CSV 文件，请检查文件路径是否正确。")
    import sys
    sys.exit(1)

# 划分训练集和测试集
train_texts, test_texts, train_labels, test_labels = train_test_split(
    df['text'].tolist(), df['label'].tolist(), test_size=0.2, random_state=42
)

# 创建数据集和数据加载器
max_length = 128
```

```
train_dataset = CrystallineSiliconDataset(train_texts, train_labels, tokenizer, max_length)
test_dataset = CrystallineSiliconDataset(test_texts, test_labels, tokenizer, max_length)

train_dataloader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_dataloader = DataLoader(test_dataset, batch_size=32, shuffle=False)

# 定义训练参数
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)

optimizer = AdamW(model.parameters(), lr=0.005)
epochs = 3000

# 训练模型
for epoch in range(epochs):
    model.train()
    total_loss = 0
    for batch in train_dataloader:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        model.zero_grad()
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        total_loss += loss.item()

        loss.backward()
        optimizer.step()

    print(f'Epoch {epoch + 1}/{epochs}, Loss: {total_loss / len(train_dataloader)}')

# 评估模型并输出情感得分
model.eval()
with torch.no_grad():
    for index, row in df.iterrows():
        text = row['text']
        encoding = tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
```

```
        )
        input_ids = encoding['input_ids'].to(device)
        attention_mask = encoding['attention_mask'].to(device)

        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        probabilities = torch.softmax(logits, dim=1)

        negative_score = probabilities[0][0].item()
        positive_score = probabilities[0][1].item()
        # 将得分转换到 -1 到 1 的区间
        sentiment_score = positive_score - negative_score

        sentiment = '积极' if sentiment_score > 0 else '消极'

        print(f'文本: {text}')
        print(f'情感类别: {sentiment}')
        print(f'情感得分: {sentiment_score:.4f}')
        print(f'晶硅销量: {row["sales_volume"]}')
        print('-' * 50)
```