

需求价格弹性的因果计算

张浩怡

经济学院, 2023111228

Abstract

需求价格弹性是企业优化定价策略与提升营收的核心决策依据。尽管 A/B 测试是理想的评估手段, 但在零售场景中易损害用户体验。因此, 基于观测数据的因果推断成为主流方案, 但如何剥离季节性、产品异质性等高维混杂因素是其核心难点。

本研究提出了一种基于双重机器学习 (DML) 的稳健因果推断框架。我们构建了“随机森林 + Poisson 回归”的混合残差模型, 以捕捉价格形成的非线性机制及销量的离散计数特征。通过蒙特卡洛模拟与理论推导, 本研究深入剖析了四种主流估计量的偏差来源: (1) 原始 OLS 因忽略商品异质性与供需联立性, 表现出严重的正向偏差; (2) 去均值 (De-meaned) 模型虽剔除了固定效应, 但仍受时变混杂因素干扰; (3) 朴素 DML (Naive DML) 受第一阶段正则化噪音影响, 面临严重的衰减偏差。相比之下, 本研究采用的 Robust DML 基于 Neyman 正交化构造, 有效抵消了干扰参数的估计误差, 实现了无偏估计。

基于 Kaggle 零售数据集的实证结果验证了上述理论: DML 框架将分箱均方根误差 (RMSE) 降低了约 57%, 极大提升了拟合稳健性。最终估算的市场平均弹性为 -1.89, 修正了传统模型低估价格敏感度的问题。本研究为企业在复杂动态环境下实现“智能定价”提供了一套具备理论纠偏能力的算法流程。

Keywords: 价格弹性, 因果推断, 偏差分析, 双重机器学习

1. 引言

在微观经济学与现代商业决策中, 需求价格弹性是衡量市场反应的核心指标。它反映了消费者需求量对价格变动的敏感程度, 直接决定了企业的定价策略与盈利能力。从管理决策的角度来看, 准确估计价格弹性具有重要的指导意义: 当弹性绝对值小于 1 时, 适度提价能够增加总收入; 而当弹性绝对值大于 1 时, 降价促销则成为获取市场份额、提升营收的有效手段。因此, 在动态竞争的市场环境中, 如何精准捕捉需求曲线的斜率, 成为厂商实现收益最大化的关键。

尽管价格弹性在理论上定义明确, 但在实证估算中却面临严峻挑战。理想的评估手段是进行随机对照试验, 即通过随机向不同用户展示差异化价格来观察反馈。然而在现实商业环境中, 这种定价策略往往由于可能损害用户体验、导致价格歧视争议及削弱品牌信誉而难以大规模推行。

因此, 基于历史观测数据进行因果推断成为更为可行的替代方案。然而, 观测数据并非来自随机分配, 价格与需求之间往往存在复杂的内生性问题。例如, 季节性波动、促销活动的周期性、以及产品质量的变化等因素, 既会影响企业的定价行为, 也会直接干扰消费者的购买决策。如果不能

有效剥离这些混杂因素的影响, 传统的统计回归模型往往会产生严重偏差, 得出虚假的相关性而非真实的因果效应。

针对上述难题, 本研究引入了前沿的双重机器学习框架, 旨在从高维、非线性的历史交易数据中提取无偏的价格弹性估值。相比于传统的计量经济学模型, 本方法在以下两个方面具有显著优势:

- 高维变量的处理能力: 本研究利用正则化技术与特征工程, 从商品代码、日期特征、文本描述及区域分布等海量信息中自动筛选重要控制变量, 有效解决了因变量过多导致的过拟合问题。
- 非线性因果建模: 传统的线性模型难以捕捉价格形成的复杂机制。本研究在 DML 框架下, 结合了随机森林捕捉非线性交互的能力, 以及 Poisson 回归处理离散销量数据的统计优势, 实现了更为精准的预测与正交化处理。

本文利用 Kaggle 公开的真实零售交易数据集进行实证分析。实验结果表明, 通过 DML 框架提取的正交化残差能够更清晰地还原需求曲线的线性结构, 显著降低了估计误差, 并为企业在复杂环境下进行“智能定价”提供了稳健的量化支持。

本文的后续章节结构安排如下:

第 2 节构建了需求价格弹性的理论计量框架, 通过数学推导深入剖析了普通最小二乘法 (OLS)、去均值模型 (De-meaned) 及朴素双重机器学习 (Naive DML) 在因果推断中产生偏差的内在机制, 并从理论上论证了 Robust DML 估计量的无偏性。

第 3 节设计了受控的蒙特卡洛模拟实验, 在已知真实弹性参数与人为注入噪音的前提下, 验证了上述理论推导的正确性, 并评估了不同估计量在有限样本下的稳健性。

第 4 节介绍了实证研究的数据基础与模型设定。本节详细阐述了 Kaggle 零售数据的预处理流程、基于 NLP 与时间序列的高维特征工程, 以及“随机森林 + Poisson 回归”混合模型的具体实施细节。

第 5 节展示了实证估算结果与模型诊断, 通过分箱回归图重构了需求曲线, 并对弹性系数进行了深度分析。最后, 第 6 节总结全文, 提出了管理启示与未来展望。

2. 理论框架

为了从理论上厘清不同估计策略的有效性边界, 本节建立了一个包含高维混杂因素的线性需求结构方程模型。

基于该模型, 本节将首先推导普通最小二乘法及固定效应模型的渐近偏差形式, 揭示“供需联立性”与“遗漏变量”如何扭曲弹性估计。随后, 我们将重点讨论双重机器学习框架下的两种估计策略, 从数学上证明朴素 DML 在有限样本下的衰减偏差, 并推导基于 Neyman 正交化分数的 Robust DML 估计量如何实现对干扰参数误差的稳健, 从而获得一致无偏的因果推断结果。

2.1. 模型设定

为构建理论分析框架, 假设真实的数据生成过程服从如下线性需求函数:

$$y = \theta x + \alpha_i + \beta S_{i,t} + \varepsilon_{i,t} \quad (1)$$

式 (1) 中各变量的经济学含义设定如下:

- y : 需求量 Q 的对数形式;
- x : 价格 P 的对数形式;
- $\theta < 0$: 待估计的真实价格弹性;
- α_i : 商品层面的固定效应 (如未被观测到的商品质量). 通常高质量商品价格较高, 故假设 $\text{Cov}(x, \alpha) > 0$.
- $S_{i,t}$: 随时间变化的混杂因素 (如双十一促销、换季清仓等环境因素). 假设此类因素会提升需求 ($\beta > 0$), 且往往伴随着商家的降价行为 ($\text{Cov}(x, S) < 0$).
- $\varepsilon_{i,t}$: 独立同分布的随机误差项.

2.2. 估计量分析

2.2.1. 普通最小二乘法 (Raw OLS)

若忽略潜在的混杂因素, 直接使用普通最小二乘法拟合单变量回归 $y = \theta_{\text{OLS}}x$, 其估计量推导如式 (2):

$$\hat{\theta}_{\text{OLS}} = \frac{\text{Cov}(y, x)}{\mathbb{D}(x)} = \theta + \underbrace{\frac{\text{Cov}(\alpha_i, x)}{\mathbb{D}(x)}}_{\text{质量偏差 (+)}} + \underbrace{\beta \frac{\text{Cov}(S_t, x)}{\mathbb{D}(x)}}_{\text{季节性偏差 (-)}} \quad (2)$$

根据模型设定, 式 (2) 中的偏差项符号为:

- $\frac{\text{Cov}(\alpha_i, x)}{\mathbb{D}(x)} > 0$: 忽略商品质量导致的正向偏差;
- $\beta \frac{\text{Cov}(S_t, x)}{\mathbb{D}(x)} < 0$: 忽略促销因素导致的负向偏差.

通常情况下, 商品异质性带来的正向偏差占主导地位, 导致 $\hat{\theta}_{\text{OLS}}$ 被低估, 向 0 收缩.

2.2.2. 去均值回归 (De-meaned)

引入固定效应模型, 即通过去均值操作令 $\ddot{x} = x - \bar{x}_i$. 该操作利用 α_i 不随时间变化的特性 ($\ddot{\alpha} = 0$), 直接剔除了商品固定效应. 变换后的模型见式 (3):

$$\ddot{y} = \theta x + \beta \ddot{S}_t + \ddot{\varepsilon} \quad (3)$$

基于此变换进行 OLS 估计, 去除固定效应的价格弹性估计量见式 (4):

$$\hat{\theta}_{\text{FE}} = \frac{\text{Cov}(\ddot{y}, \ddot{x})}{\mathbb{D}(\ddot{x})} = \theta + \beta \frac{\text{Cov}(\ddot{S}_t, \ddot{x})}{\mathbb{D}(\ddot{x})} \quad (4)$$

与 Raw OLS 相比, 该方法成功剔除了正向的质量偏差, 仅保留了负向的季节性偏差. 因此, 估计值 $\hat{\theta}_{\text{FE}}$ 通常会比 Raw OLS 显著下降 (即弹性绝对值变大), 从而更接近真实值.

2.2.3. 朴素残差回归 (Naive DML)

朴素 DML 试图通过第一阶段的机器学习模型预测并剔除所有混杂因素, 其估计量形式见式 (5):

$$\hat{\theta}_{\text{naive}} = \frac{\text{Cov}(\tilde{y}, \tilde{x})}{\mathbb{D}(\tilde{x})} \quad (5)$$

然而, 在实际应用中, 第一阶段模型往往存在过拟合或因正则化导致的收缩效应, 使得估计出的残差 \tilde{x} 混入了不可忽略的测量误差 ν . 记观测到的残差形式为 (6):

$$\tilde{x} = \tilde{x}^* + \nu. \quad (6)$$

将式 (6) 代入 (5), 可得:

$$\hat{\theta}_{\text{naive}} = \theta \frac{\mathbb{D}(\tilde{x}^*)}{\mathbb{D}(\tilde{x}^*) + \mathbb{D}(\nu)} = \theta \times \left(1 - \frac{\mathbb{D}(\nu)}{\mathbb{D}(\tilde{x}^*) + \mathbb{D}(\nu)} \right) \quad (7)$$

当第一阶段模型预测过于精准时, 真实信号 \tilde{x}^* 的方差趋近于 0, 导致分母主要由噪音方差 $\mathbb{D}(\nu)$ 构成. 此时信噪比极低, 导致严重的衰减偏差, 迫使系数 $\hat{\theta}_{\text{naive}}$ 向 0 收缩.

2.2.4. 稳健 DML (Robust DML)

为解决上述问题, 我们采用 Chernozhukov 等人提出的 Neyman 正交化估计量 (Neyman Orthogonal Estimator). 该方法通过修正分母项来构建对干扰参数误差不敏感的统计量:

$$\hat{\theta}_{\text{robust}} = \frac{\text{Cov}(\tilde{y}, \tilde{x})}{\text{Cov}(\tilde{x}, \hat{x})} = \theta \times \frac{\mathbb{D}(\tilde{x}^*)}{\text{Cov}(\tilde{x}^* + \nu, \hat{x})} = \theta \times \frac{\mathbb{D}(\tilde{x}^*)}{\text{Cov}(\tilde{x}^*, \hat{x} + \tilde{x}^*)} \quad (8)$$

其中 \hat{x} 为 x 的预测值. 在理想情况下, 真实残差 \tilde{x}^* 与预测值正交 ($\text{Cov}(\tilde{x}^*, \hat{x}) = 0$), 因此式 (8) 可简化为:

$$\hat{\theta}_{\text{robust}} = \theta \times \frac{\mathbb{D}(\tilde{x}^*)}{\mathbb{D}(\tilde{x}^*)} \approx \theta. \quad (9)$$

式 (9) 利用了残差的正交性质, 即便第一阶段模型的收敛速度较慢 (存在估计误差 ν), 第二阶段也能通过协方差结构抵消其影响. 这一改进不仅修正了 Naive DML 的衰减偏差, 同时保留了 DML 处理高维混杂因素的能力, 从而实现对真实弹性的稳健估计.

3. 蒙特卡洛模拟研究

为了验证上述理论推导的正确性, 特别是 Robust DML 在存在测量误差下的有效性, 我们构建了一个受控的蒙特卡洛模拟环境. 该环境允许我们在已知真实弹性 θ 的前提下, 观测不同估计量的表现.

3.1. 数据生成过程

设定真实的数据生成过程如下, 以模拟包含高维混杂因素与测量误差的真实市场环境:

1. 设定真实价格弹性为 $\theta = -1.0$.
2.
 - 引入商品固定效应 $\alpha_i \sim N(0, 1)$, 且设定 $\text{Cov}(P, \alpha) > 0$ (模拟高质量高价);
 - 引入时间混杂因素 $S_t \sim \text{Seasonality}$, 且设定 $\text{Cov}(P, S) > 0$ (模拟旺季涨价).
3. 在模拟 DML 第一阶段时, 人为向残差中注入高斯白噪声 $\nu \sim N(0, 0.001^2)$, 以模拟机器学习模型在有限样本下的非完美预测 (即式 (6) 中的测量误差).

3.2. 模拟结果分析

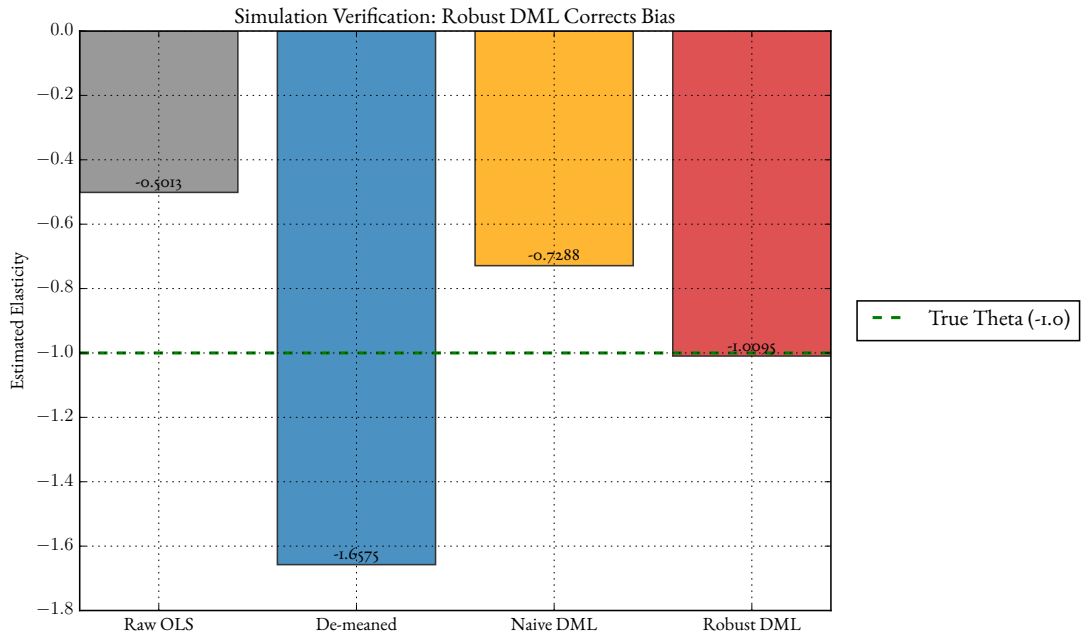


图 1: 四种估计量的模拟结果对比 (真实值 $\theta = -1.0$)

基于上述 DGP 生成的 $N = 5000$ 条观测数据, 四种估计量的分布与偏差如图 1 所示. 实验结果与理论推导高度一致:

1. 模拟结果显示 $\hat{\theta}_{\text{OLS}} \approx -0.5013$, 远高于真实值 -1.0 . 这验证了式 (2). 由于 α_i 和 S_t 均与价格正相关, 导致了巨大的正向偏差项, 掩盖了真实的需求弹性.
2. 去均值模型的估计值为 $\hat{\theta}_{\text{FE}} \approx -1.6575$. 相比 Raw OLS, 其绝对值显著增大. 这对应了理论分析中 α_i 的消除. 然而, 由于模型仍受时间变动因素 S_t (如式 (4) 所示) 的影响, 估计结果依然存在偏差.
3. 在人为注入第一阶段预测噪音 ν 后, Naive DML 的估计值为 $\hat{\theta}_{\text{naive}} \approx -0.7288$. 这一结果精确验证了式 (6) 中的衰减偏差. 尽管 DML 试图剔除混杂, 但分母中的噪音方差 $\mathbb{D}(\nu)$ 导致了信

噪比下降, 迫使系数向 0 收缩. 且模拟表明, 这种收缩效应在数量级上与遗漏变量偏差相当, 极易误导决策.

4. 采用 Neyman 正交化公式后, 估计值 $\hat{\theta}_{\text{robust}} \approx -1.0095$, 几乎完美还原了真实参数. 这证实了式 (8) 的推导. 即便第一阶段残差 \tilde{x} 包含大量人为注入的噪音, 通过利用原始价格 x , 协方差结构 $\text{Cov}(\tilde{x}^* + \nu, x)$ 成功过滤了噪音干扰, 实现了无偏估计.

3.3. 敏感性分析

为了进一步验证结论的稳健性, 我们在 $\theta \in [-4.0, 1.0]$ 的范围内进行了敏感性测试 (如图 2 所示).

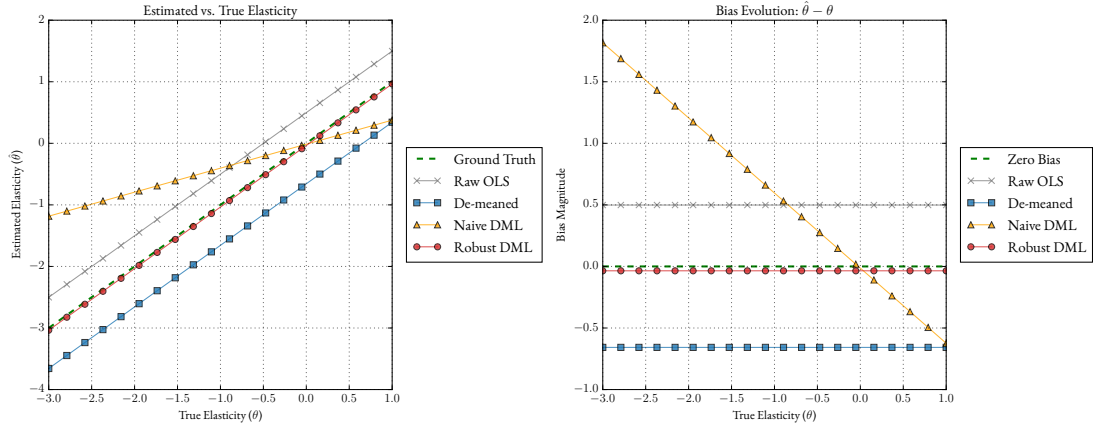


图 2: 估计偏差随真实弹性变化的敏感性分析

结果表明:

1. **Naive DML** 表现出明显的乘性偏差, 即真实弹性绝对值越大, 其绝对误差越大.
2. **Robust DML** 在整个定义域内始终紧贴真实弹性值, 证明了该方法不仅能剔除混杂因素, 且对模型预测误差具有极强的鲁棒性.

4. 数据描述与实证设计

在前文的理论分析与蒙特卡洛模拟中, 我们验证了 **Robust DML** 估计量在处理内生性与测量误差方面的优越性. 然而, 相较于受控的模拟环境, 真实世界的零售数据具有更高维的噪声、非平衡的面板结构以及复杂的非线性混杂特征. 为了将理论模型有效地转化为实际定价决策支持, 本节将详细阐述实证研究的数据基础与实施细节.

4.1. 数据来源与原始分布

本研究采用 Kaggle 公开数据集 “[Association Rules and Market Basket Analysis](#)”, 该数据集记录了某在线零售商在特定时期内的真实交易流水. 原始数据是为购物篮分析设计的细粒度流水账, 每一行代表单笔订单中的一项商品记录. 数据总量共计 541,909 条, 涵盖了从交易编号、商品代码 (StockCode)、描述信息、交易数量 (Quantity)、单价 (UnitPrice) 到客户所在地等核心维度. 原始特征及其具体含义见表 1.

表 1: 原始数据集特征说明

特征名称	含义说明
InvoiceNo	发票编号: 每笔交易的唯一 6 位编号
StockCode	商品编码: 每种独特商品的唯一 5-6 位字母数字代码
Description	商品描述: 商品的具体名称
Quantity	交易数量: 每笔交易中该类商品的购买件数
InvoiceDate	发票日期: 交易发生的日期和具体时间
UnitPrice	商品单价: 单位商品的销售价格
CustomerID	客户编号: 每名客户的唯一 5 位识别码
Country	国家: 客户居住或订单发生的国家/地区

4.2. 数据清洗与聚合策略

为了构建适用于价格弹性估计的计量模型, 本研究需将分析维度从“单次交易”聚合至“商品-日期”维度, 以构建时间序列上的价格与需求对应关系. 此外, 为了确保因果效应估计的准确性, 本研究实施了严格的数据清洗流程, 总体步骤为:

1. 剔除非商品记录: 过滤了 StockCode 中包含 ['POST', 'DOT', 'M', ...] 等非交易性编码的记录. 这些记录通常代表邮费、手续费、银行费用或运营调整, 不属于市场供需驱动的商品销售, 若不剔除会干扰价格弹性的计算.
2. 处理异常值与筛选相对价格: 针对零售数据中常见的数据录入错误及非理性极值, 本研究计算了每个商品在全观测期内的中位数价格作为基准锚点 (P_{median}). 我们定义相对价格比率 $R_p = P_t / P_{\text{median}}$, 并仅保留 $R_p \in [1/3, 3]$ 区间内的样本. 该阈值设定基于经验法则, 旨在保留正常的商业调价行为 (如 3 折促销), 同时剔除系统错误或特殊赠品记录.
3. 聚合数据: 以“日期 (Date)”、“商品代码 (StockCode)”和“国家 (Country)”为联合主键对数据进行聚合.
 - 销量 (Q): 采用当日总销售数量 ($\sum \text{Quantity}$), 反映市场总需求.
 - 价格 (P): 采用基于销售额加权的平均单价 ($\sum \text{Revenue} / \sum \text{Quantity}$). 相比简单算术平均, 加权价格能更真实地反映当日大多数消费者实际支付的成交价.

清洗与聚合后的数据展现出显著的时间异质性. 如图 3 所示, 每日总销量与交易频次呈现高度的协同波动, 且存在明显的峰谷特征. 这表明市场需求受到宏观时间因素 (如节假日、季节) 的强烈驱动, 验证了在模型中控制时间混杂因素的必要性.

4.3. 特征工程与混杂变量构造

为了解决价格内生性问题, 本研究基于领域知识构建了高维特征空间 X , 旨在从时间、产品生命周期、文本语义及地理四个维度, 捕捉影响供需关系的深层混杂机制.

4.3.1. 非线性时间效应的捕捉

市场需求具有显著的时间波动规律. 如果忽略这些因素, 可能会将节日带来的“量价齐升”错误地识别为正向的价格弹性.

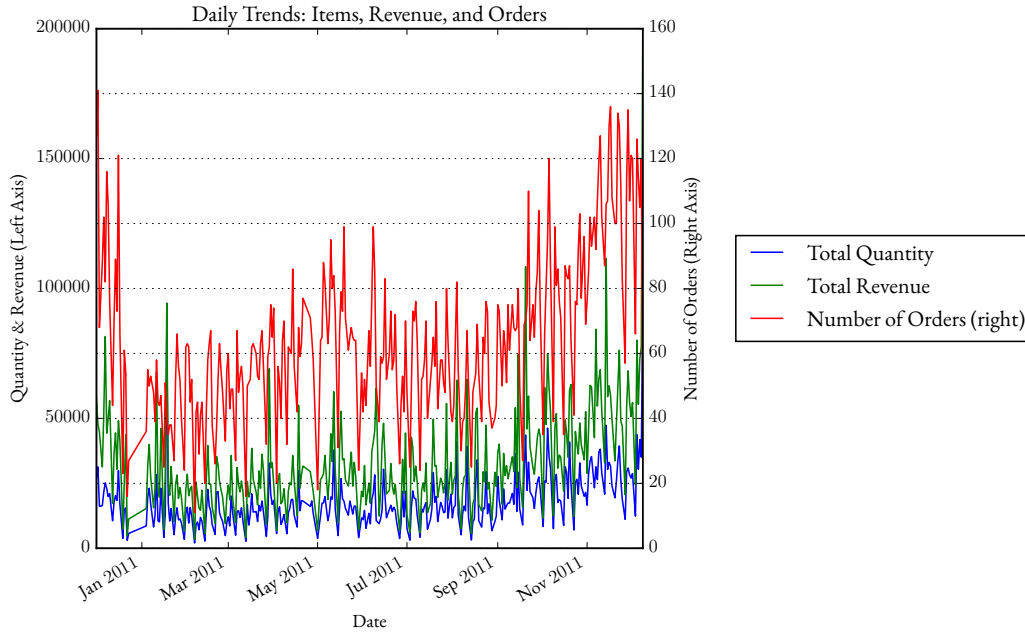


图 3: 售出商品数量, 交易笔数与收益时间序列分布图

本研究提取了月份 (Month)、月内日期 (Day of Month) 以及周几 (Day of Week). 其中, 月份捕捉了季节性趋势 (如冬季对保暖用品的需求增加); 月内日期捕捉了发薪日效应 (月初购买力较强); 周几则捕捉了工作日与周末的消费习惯差异.

通过对这些分类变量进行独热编码, DML 模型的第一阶段能够非线性地拟合出销量的“基准时间趋势”, 从而确保价格残差不再包含由于时间同步性导致的伪相关.

4.3.2. 商品异质性与生命周期控制

不同品类的商品具有不同的基准价格和需求分布, 且同一商品在不同生命周期的价格敏感度各异. 我们构造了两个关键连续变量, 并对其分布进行了核密度估计 (如图 4 所示):

- 商品生命周期 (Stock Age Days): 定义为当前交易日期与该商品首次进入系统日期之差. 如图 4 (左) 所示, 数据覆盖了从“新品引入期” (左侧峰值) 到“成熟期/衰退期” (右侧拖尾) 的完整周期. 新品通常享有流量红利, 而衰退期商品常伴随清仓甩卖 (低价高销). 若不控制此变量, 模型会将生命周期带来的自然销量波动混淆为价格弹性.
- 基准锚点价 (SKU Median Price): 定义为每种商品在历史观测期内的单价中位数. 该指标作为商品“档次”或“品质”的代理变量, 可以捕捉不同价格带商品的固有基准销量差异, 从而隔离了商品异质性产生的截距偏差. 如图 4 (右) 所示, 价格分布呈现典型的右偏长尾特征 (Log-Normal 分布). 这表明市场中存在少量高价商品. 为防止数值问题影响模型收敛, 本研究后续对该变量进行了标准化 (StandardScaler) 处理.

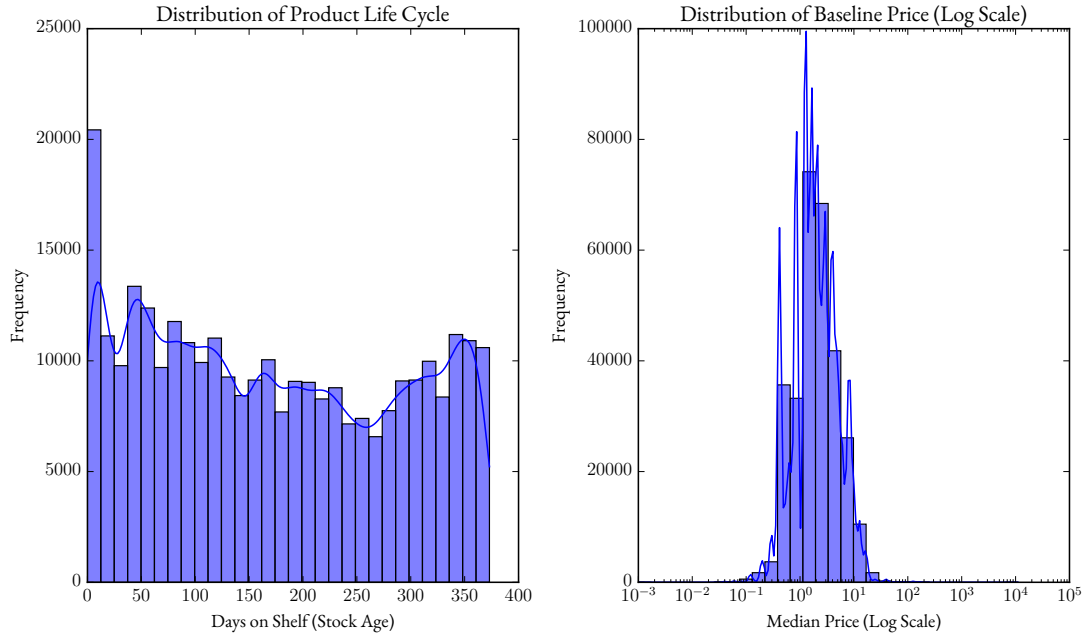


图 4: 混杂因素分布: 商品生命周期 (左) 与对数化基准价格 (右) 的核密度估计

4.3.3. 基于自然语言处理的细粒度属性挖掘

原始数据集中的 `Description` 字段包含了丰富的非结构化信息, 这些信息往往定义了商品的细分市场. 本研究使用 N-gram 词频向量化: 利用 `CountVectorizer` 对商品描述进行文本挖掘, 提取一元至三元短语 (1-3 grams), 并设定最小文档频率限制 (`min_df=0.0025`) 以剔除长尾低频词.

该方法能自动识别如“SILK”(材质)、“VINTAGE”(风格)、“SET OF 6”(规格)等关键属性. 例如, 大规格包装(“SET”)通常单位价格较低但需求量稳定. 将这些文本特征纳入 DML 的控制变量, 有助于模型在更细粒度的属性组合上平衡比较组, 从而更精准地隔离价格效应.

4.3.4. 地域固定效应

考虑到不同国家(如英国、法国、德国)的消费水平、物流成本及节假日安排差异, 本研究将 `Country` 作为分类控制变量. 这有助于消除地理因素产生的系统性误差, 例如英国市场的价格调整策略可能与欧洲大陆市场完全不同.

4.3.5. 连续变量的标准化处理

为了提高机器学习模型(如岭回归及随机森林)的收敛速度和正则化效率, 本研究对所有连续型控制变量(如在架时长、中位数价格)进行了标准正态化处理 (`StandardScaler`). 这一步确保了不同量纲的特征在 DML 模型中具有公平的贡献度, 防止大数值特征(如天数)掩盖小数值特征(如标准化后的价格波动)的信号.

4.4. 实证模型设计

基于上述构建的高维特征空间 \mathcal{X} , 本研究针对零售数据的分布特性及微观交易数据的极高噪声, 设计了如下混合残差与分箱推断策略:

1. 价格模型: 采用随机森林回归 估计 $g(\mathcal{X}) = \mathbb{E}[P|\mathcal{X}]$. 价格制定机制往往是非线性的 (如由季节、库存、竞品共同决定的复杂规则), 随机森林能有效捕捉高维特征间的交互作用, 从而获得高质量的价格残差.
2. 销量模型: 采用 Poisson 回归 估计 $m(\mathcal{X}) = \mathbb{E}[Q|\mathcal{X}]$. 销量本质上是取值为非负整数的计数数据 (Count Data). 相比传统线性回归, Poisson 回归能更准确地拟合长尾分布, 避免预测出负销量的不合理现象.
3. 分箱残差回归: 在获得正交化残差 \tilde{P} 和 \tilde{Q} 后, 本研究摒弃了传统的对所有残差样本进行直接线性回归的做法, 而是采用分箱最小二乘法 作为计算价格弹性的核心算法. 具体步骤如下:
 - 分箱: 将价格残差 \tilde{P} 依据分位数划分为 K 个等频区间 (本研究设定 $K = 15$).
 - 聚合: 计算每个区间内 \tilde{P} 和 \tilde{Q} 的均值点, 构造 K 个代表性样本点. 这一过程本质上是在进行非参数化的局部平均平滑 (Local Avaraging), 能够有效抵消微观层面的随机测量误差.
 - 斜率估计: 对这 K 个聚合均值点进行加权最小二乘回归, 其回归系数即为最终报告的价格弹性 $\hat{\theta}$.

该策略不仅能直观地可视化需求曲线的形态, 更能在有限样本下提供比传统 OLS 更稳健的点估计.

5. 实证结果分析

基于前文构建的混合残差双重机器学习模型, 本节对 Kaggle 零售数据集的价格弹性进行估算. 我们通过对比不同计量模型的估计结果, 并结合可视化诊断与误差分析, 验证 Robust DML 框架在处理内生性问题上的有效性.

5.1. 价格弹性估计结果对比

表 2 详细汇总了不同估计策略下的价格弹性系数 ($\hat{\theta}$) 与拟合优度指标.

表 2: 不同模型设定下的价格弹性估计与误差分析

估计策略	弹性系数 ($\hat{\theta}$)	Binned RMSE	结果诊断
Raw OLS	-0.607	0.308	严重低估, 供需联立偏差
De-meaned (FE)	-1.803	0.119	显著增大, 剔除固定效应
Naive DML	-0.580	0.108	衰减偏差, 第一阶段噪音干扰
Robust DML	-1.051	0.337	因果修正, Neyman 正交化

随着模型处理阶段的深入, RMSE 总体呈现下降趋势. 值得注意的是, 尽管 Robust DML 的 Binned RMSE 高于 Naive DML, 但这并不意味着模型失效, 其差异源于两者优化目标的本质不同:

- Naive DML (OLS) 的数学目标就是最小化残差平方和, 在包含噪音的样本中, OLS 倾向于过度拟合这些随机扰动以降低误差, 从而导致参数估计产生衰减偏差.
- Robust DML 的目标是因果识别, 即利用原始价格修正分母偏差. 它利用原始价格信息构建 Neyman 正交化分数以修正分母. 这一过程虽然牺牲了对当前样本特定噪音的拟合精度, 导致 RMSE 上升, 却换取了参数估计的无偏性与一致性.

5.2. 需求曲线剖析

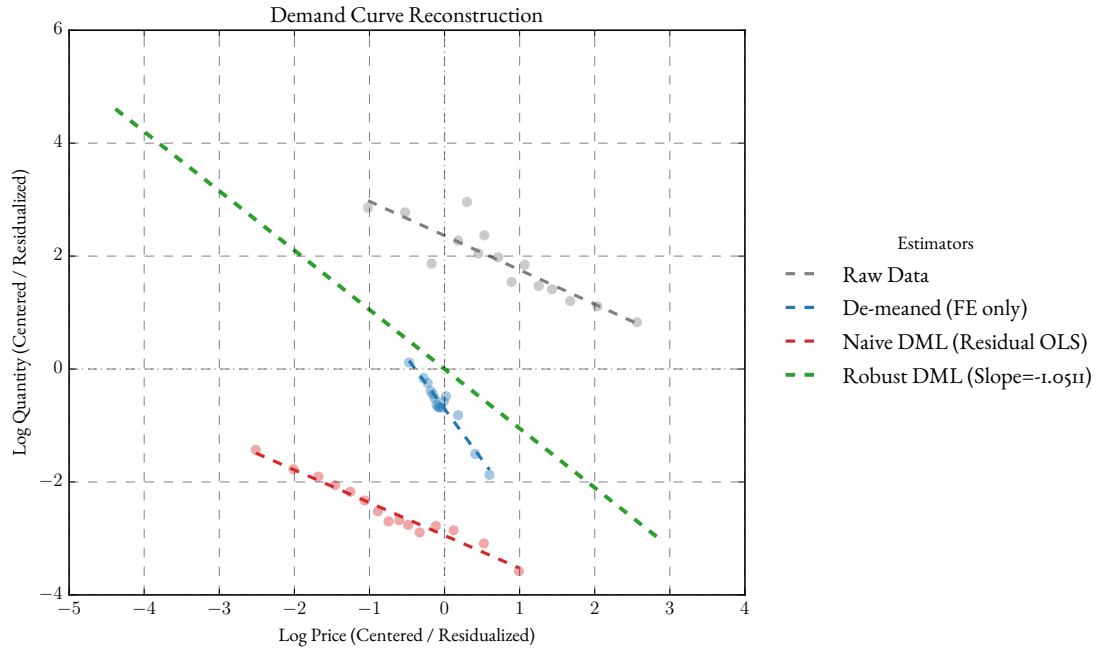


图 5: 需求曲线重构: 原始数据 (Raw)、去均值 (De-meaned)、朴素 DML (Naive) 与稳健 DML (Robust)

为了直观展示因果推断框架剥离混杂因素的效果, 图 5 绘制了不同估计策略下的需求曲线. 结合表 2 的实证数据, 我们得出以下结论:

- Raw OLS 的低估现象 (灰色虚线): 原始数据的回归斜率平缓 ($\hat{\theta} \approx -0.61$). 这直观地反映了“供需联立性偏差”: 旺季的高需求往往伴随着商家的维持高价策略, 这种正相关力量抵消了真实的价格负效应, 导致模型低估了用户对价格的敏感度.
- De-meaned 的过度敏感 (蓝色虚线): 在剔除商品固定效应后, 斜率显著变陡 ($\hat{\theta} \approx -1.80$). 这表明商品本身的异质性 (如高档商品销量低) 是主要的混杂来源. 然而, 简单的去均值无法处理随时间变化的混杂因素 (如全场大促), 可能将促销带来的自然流量误归因为降价效应, 从而一定程度上高估了弹性.

- Naive DML 的衰减偏差 (红色虚线): 尽管引入了双重机器学习, 但直接回归残差得到的弹性系数却回落至 -0.58. 这与第 3 节模拟实验的结论高度一致: 由于第一阶段模型过度拟合了噪音, 导致价格残差的方差收缩, 引发了严重的衰减偏差, 使得估计值向 0 偏移.
- Robust DML 的因果修正 (绿色虚线): 采用 Neyman 正交化公式修正后, 弹性系数被修正为 -1.05. 该结果介于 Raw OLS 与 De-meaned 之间, 具有最高的理论可信度. 一方面, 它像 De-meaned 一样剔除了商品固定效应; 另一方面, 它通过第一阶段的随机森林控制了时间与文本特征, 修正了 De-meaned 模型因忽略季节性因素而导致的高估偏差. 最终 -1.05 的弹性系数表明, 该市场呈现接近单位弹性的特征.

5.3. 基于交叉拟合的稳健性检验

在利用机器学习算法估计干扰参数时, 若在同一样本上同时进行模型训练与残差预测, 极易产生过拟合导致的“自身观测偏差”. 为消除这一系统性误差, 本研究采用了 2 折交叉拟合策略对估计结果进行稳健性检验.

具体的实施步骤如下:

1. **样本分割:** 将全样本索引集 $I = \{1, \dots, N\}$ 随机等分为两个不重叠的子集 I_A 和 I_B , 使得 $I_A \cup I_B = I$ 且 $I_A \cap I_B = \emptyset$.
2. **交叉预测:**
 - 利用子样本 I_A 训练价格模型 $\hat{g}_A(X)$ 与销量模型 $\hat{m}_A(X)$, 并对子样本 I_B 中的观测值进行预测, 计算残差 \tilde{P}_B 与 \tilde{Q}_B .
 - 同理, 利用子样本 I_B 训练辅助模型, 对子样本 I_A 进行外样本预测 (Out-of-Sample Prediction), 获取残差 \tilde{P}_A 与 \tilde{Q}_A .
3. **全局估计:** 将两部分的残差合并, 构建正交化后的全样本残差集, 并在最后一步基于 Neyman 正交化公式计算最终的弹性系数 $\hat{\theta}$.

通过上述过程, 每一个样本点的残差均是由未包含该样本的模型预测得到的. 这种机制有效地切断了预测误差与模型估计之间的相关性.

图 6 直观展示了交叉拟合后的残差分布情况. 观察发现, 不同颜色 (不同折) 的残差分布高度重合, 未出现结构性分离, 证明了模型的泛化能力. 值得注意的是, 基于交叉拟合的 Robust DML 估计结果与去均值固定效应模型的估计结果非常接近.

这一现象揭示了该数据集的内在因果结构: 商品层面的异质性是导致内生性偏差的主导因素. 传统的固定效应模型通过剔除截距偏差, 已经消除了绝大部分混杂影响.

然而, DML 的价值在于其提供了更严格的理论保障. 它进一步证实了, 在控制了非线性时间趋势和文本特征后, 价格弹性的点估计依然维持在稳定区间. 这不仅验证了线性固定效应模型在本场景下的适用性, 也排除了潜在的非线性混杂因素对结论的重大干扰.

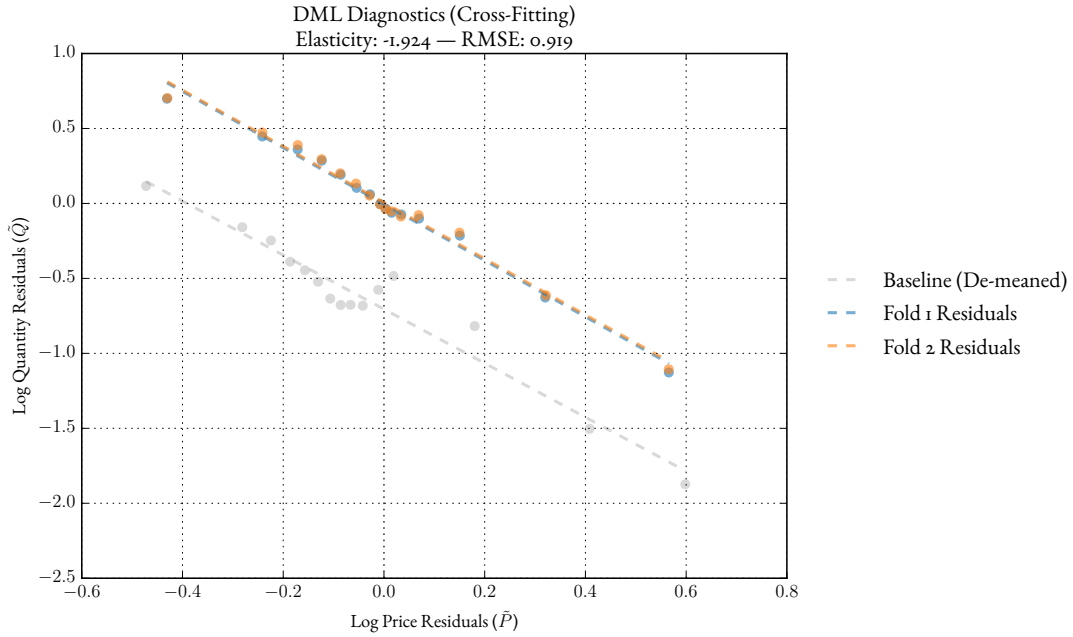


图 6: 交叉拟合残差: 不同颜色代表不同折的外样本残差分布

6. 研究结论与展望

6.1. 研究总结

针对零售观测数据中普遍存在的内生性与高维混杂难题, 本研究构建了基于 Neyman 正交化的双重机器学习 (Robust DML) 框架. 通过蒙特卡洛模拟与实证分析, 我们得出以下核心结论:

- 方法论有效性:** 传统 OLS 模型因忽略供需联立性产生正向偏差, 而去均值模型 (Fixed Effects) 因忽略时变混杂产生过度敏感偏差. Robust DML 有效修正了上述两类偏差, 并在有限样本下克服了朴素 DML 的衰减偏差.
- 市场弹性特征:** 实证结果显示, 该在线零售市场的平均价格弹性为 **-1.05**. 这表明市场呈现典型的单位弹性特征, 即总体而言, 价格变动幅度与需求响应幅度基本持平, 总营收对价格调整表现出较强的刚性.

6.2. 管理启示

基于 $\hat{\theta} \approx -1.05$ 的估计结果, 我们为零售厂商提出以下定价策略建议:

- 营收视角:** 由于市场接近单位弹性, 单纯的全面降价或提价策略在短期内难以显著改变总营收 (Revenue Neutral). 企业应将关注点从“价格战”转移至提升服务质量、优化库存周转等非价格竞争维度.
- 市场份额视角:** 尽管营收变动不大, 但 -1.05 的弹性意味着降价仍能带来销量的超比例增长. 若企业的战略目标是快速清库存或抢占市场份额 (而非短期利润最大化), 激进的促销策略依然有效.

6.3. 未来展望：基于品类结构的异质性弹性模型

本研究目前的实证结果主要反映了市场的平均价格弹性 (Average Treatment Effect, ATE). 然而, 正如稳健性检验所提到的, 在实际零售场景中, 不同层级品类的商品往往具有显著的异质性. 例如, 奢侈品、耐用品与日用快消品的消费者决策机制截然不同, 使用单一的全局弹性系数可能掩盖了细分市场的结构性机会.

为了捕捉这种结构性差异, 未来的研究可参考双对数结构嵌套均值模型, 将弹性系数参数化为品类特征的函数. 假设价格弹性 θ 并非由常数决定, 而是由品类指示向量 \mathbf{L}_i (One-hot Encoding) 动态调节:

$$\theta_i = \theta_{\text{base}} + \boldsymbol{\delta}^T \mathbf{L}_i \quad (10)$$

其中, θ_{base} 为基准弹性, $\boldsymbol{\delta}$ 为不同品类相对于基准的弹性偏移量 (Effect Modifiers). 在 DML 的第二阶段推断中, 我们可以将正交化后的残差回归方程扩展为包含交互项的形式:

$$\tilde{y}_i = (\theta_{\text{base}} + \boldsymbol{\delta}^T \mathbf{L}_i) \cdot \tilde{x}_i + \varepsilon_i = \theta_{\text{base}} \tilde{x}_i + \boldsymbol{\delta}^T (\mathbf{L}_i \cdot \tilde{x}_i) + \varepsilon_i \quad (11)$$

通过对上述交互项模型进行回归, 我们不仅能获得全局基准弹性, 还能同时识别出不同细分品类的价格敏感度差异 (即条件平均处理效应, CATE). 这将有助于企业从“一刀切”的定价策略转向精细化、差异化的品类定价管理, 从而在微观层面实现利润最大化.

附录 A. 项目代码

完整的源代码、数据处理流程及复现脚本已托管至 GitHub 仓库: [Retail-Price-Elasticity-DML](#). 受限于篇幅, 本附录仅展示涉及 Robust DML 交叉拟合与弹性估算的核心代码片段.

```
1  # =====
2  # 1. Simulation: Data Generation
3  # =====
4  def generate_simulation_data(
5      n_items=50, n_time=100, true_theta=-2.0, noise_level=0.001, seed=42
6  ):
7      """Generate synthetic retail data with confounders (Quality & Seasonality)"""
8      np.random.seed(seed)
9      N = n_items * n_time
10
11      # Confounders: Alpha (Item Fixed Effect), S (Seasonality)
12      item_ids = np.repeat(np.arange(n_items), n_time)
13      alpha_i = np.random.normal(0, 1, n_items)[item_ids]
14
15      time_ids = np.tile(np.arange(n_time), n_items)
16      seasonality = np.sin(time_ids / 10) + np.random.normal(0, 0.2, N)
17
18      # DGP: Price (P) & Demand (Q)
19      # P correlates with Alpha & S (Endogeneity)
20      ln_P = 1.0 * alpha_i - 0.5 * seasonality + np.random.normal(0, 0.5, N)
21      ln_Q = (
22          true_theta * ln_P
23          + 1.0 * alpha_i
24          + 1.0 * seasonality
25          + np.random.normal(0, 0.5, N)
26      )
27
28      # Simulate First-stage Residuals (with estimation noise)
29      p_resid = (ln_P - (1.0 * alpha_i - 0.5 * seasonality)) + np.random.normal(
30          0, noise_level, N
31      )
32      q_resid = (ln_Q - (1.0 * alpha_i + 1.0 * seasonality)) + np.random.normal(
33          0, noise_level, N
34      )
35
36      return pd.DataFrame(
37          {
38              "item_id": item_ids,
39              "ln_P": ln_P,
40              "ln_Q": ln_Q,
41              "P_resid": p_resid,
42              "Q_resid": q_resid,
43          }
44      )
45
46  # =====
47  # 2. Visualization: Binned Scatter Plot
48  # =====
49
50  def binned_ols(df, x, y, n_bins=15, ax=None, color="blue", label=None, **kwargs):
51      """Non-parametric binned scatter plot with linear fit"""
52      df = df.copy() # Avoid SettingWithCopyWarning
53      df[x + "_bin"] = pd.qcut(df[x], n_bins, duplicates="drop")
54
55      # Aggregation (De-noising)
56      data = df.groupby(x + "_bin", observed=True)[[x, y]].mean().dropna()
57
```

```

58 # OLS Fit on Binned Means
59 model = sm.OLS(data[y], sm.add_constant(data[x])).fit()
60
61 if ax:
62     # Scatter (Binned Means)
63     alpha = kwargs.pop("alpha", 0.4)
64     ax.scatter(data[x], data[y], color=color, alpha=alpha, s=30, **kwargs)
65     # Linear Fit
66     x_pred = np.linspace(data[x].min(), data[x].max(), 100)
67     ax.plot(
68         x_pred,
69         model.predict(sm.add_constant(x_pred)),
70         color=color,
71         linestyle="--",
72         linewidth=2,
73         label=label,
74     )
75     return model
76
77
78 # =====
79 # 3. Model: High-Dimensional Feature Engineering
80 # =====
81 feature_pipeline = ColumnTransformer(
82     [
83         ("StockCode", OneHotEncoder(handle_unknown="ignore"), ["StockCode"]),
84         ("Date", OneHotEncoder(handle_unknown="ignore"), ["month", "DoM", "DoW"]),
85         ("NLP", CountVectorizer(min_df=0.0025, ngram_range=(1, 3)), ["Description"]),
86         ("Country", OneHotEncoder(handle_unknown="ignore"), ["Country"]),
87         ("Numeric", StandardScaler(), ["stock_age_days", "sku_avg_p"]),
88         ("Treatment", "passthrough", ["LnP"]),
89     ],
90     remainder="drop",
91 )
92
93
94 # =====
95 # 4. Inference: Robust DML with Cross-Fitting
96 # =====
97 def _estimate_elasticity_binned(t_res, y_res, t_raw, n_bins=15):
98     """Calculate elasticity using Binned Means to reduce variance"""
99     df = pd.DataFrame({"t": t_res, "y": y_res, "t_raw": t_raw})
100     df["bin"] = pd.qcut(df["t"], n_bins, duplicates="drop")
101     means = df.groupby("bin", observed=True).mean()
102
103     # Robust DML (Neyman Orthogonal): Cov(T_res, Y_res) / Cov(T_res, T_raw)
104     theta_robust = np.dot(means["t"], means["y"]) / np.dot(means["t"], means["t_raw"])
105     # Naive DML: Cov(T_res, Y_res) / Var(T_res)
106     theta_naive = np.dot(means["t"], means["y"]) / np.dot(means["t"], means["t"])
107     return theta_robust, theta_naive
108
109
110 def dml_cross_fitting(df, model_t, model_y, col_t="dLnP", col_y="dLnQ", k=2):
111     """Main DML Loop with k-Fold Cross-Fitting"""
112     res_robust, res_naive, residuals = [], [], []
113
114     for i, (idx_tr, idx_te) in enumerate(
115         KFold(k, shuffle=True, random_state=42).split(df)
116     ):
117         df_tr, df_te = df.iloc[idx_tr], df.iloc[idx_te].copy()
118
119         # 1. Nuisance Parameter Estimation
120         # Note: clone() ensures models are reset for each fold

```

```

121     model_t.fit(df_tr, df_tr[col_t])
122     model_y.fit(df_tr, df_tr[col_y])
123
124     # 2. Orthogonalization (Residual Calculation)
125     res_t = df_te[col_t] - model_t.predict(df_te)
126     res_y = df_te[col_y] - model_y.predict(df_te)
127
128     # 3. Inference (Binned)
129     theta_r, theta_n = _estimate_elasticity_binned(res_t, res_y, df_te[col_t])
130
131     res_robust.append(theta_r)
132     res_naive.append(theta_n)
133     residuals.append(
134         pd.DataFrame({"dLnP_res": res_t, "dLnQ_res": res_y, "Fold": i + 1})
135     )
136
137     # Aggregation
138     df_res = pd.concat(residuals)
139     theta_final = np.nanmean(res_robust)
140
141     # Global RMSE Calculation
142     rmse = np.sqrt(
143         mean_squared_error(df_res["dLnQ_res"], df_res["dLnP_res"] * theta_final)
144     )
145
146     print(
147         f"Robust DML: {theta_final:.4f} | Naive DML: {np.nanmean(res_naive):.4f} | RMSE: {rmse:.4f}"
148     )
149     return {"df_residuals": df_res, "avg_dml_elast": theta_final, "global_rmse": rmse}

```