数据要素市场 实验报告

实现类选题二：机器学习模型交易完整流程

参考论文：A Marketplace for Data: An Algorithmic Solution

组员：朱致远 孟祥烨 黄晗

代码实现内容：论文中机器学习模型交易的全流程，以及前端页面

一、问题描述

在当今机器学习广泛应用于商业决策的背景下，数据成为了一种极具价值的商品。高质量的训练数据直接决定了模型的预测性能，然而，大多数企业，尤其是处于早期阶段的企业，难以自行获取足够且相关的数据集。此外，预测任务高度多样化，数据价值具有强烈的组合性（即多个数据集的联合价值非简单可加），并且在未实际应用之前，无法预先评估数据的有效性。更为复杂的是，数据本身是一种可无限复制的数字商品，这种特性使得传统的市场机制（如在线广告拍卖、预测市场）无法直接适用。

本文正是针对这一背景，提出了一个核心问题：如何设计一个在理论上合理且在计算上可行的数据交易市场，使得数据买卖双方能够通过市场机制实现真实报价、动态定价与收益合理分配？

我们将论文尝试解决的问题概括为以下：

1. 如何为数据定价，尤其是具有组合价值的数据集合？
2. 如何激励买家如实揭示其对预测精度的真实估值？
3. 如何将由预测增益所产生的收益合理地分配给数据卖家？
4. 如何设计上述机制的算法，使其在大规模在线市场中实时可运行？

因此，我们把实现目标分为三个步骤：第一步是诚实的机器学习模型拍卖，第二步是 MWU 算法动态定价，第三步是收益分配。 最终将三个步骤连接，完成完整流程的所有功能。

二、文章整体思路

文章提出了一个完整的数据市场架构，并将其建模为一个**双边市场（two-sided market）**：

- **卖家**: 每个卖家提供一个数据特征（如时间序列），其本质为可复制的数字商品；
- **买家**: 每个买家带有一个特定的预测任务（如时间序列预测）以及对预测精度提高的**主观估值（μ）**。

以下是论文涉及的关键术语：

符号/变量	程序中体现	实际含义
μ	`mu = buyer["mu"]`	买家愿意为 1 单位提升支付的价格
b	`b = mu`	买家出价（我们简化设为 μ）
p_n	`p_n = price_updater.choose_price()`	当前市场定价
X	`X = buyer["X"]`	所有卖家提供的特征矩阵
$\tilde{X}$	`X_tilde = allocation_function(...)`	根据 p 和 b 分配的数据
Y	`Y = buyer["Y"]`	预测目标
$\hat{Y}$	`Y_hat = train_and_predict(...)`	预测值
G	`gain = gain_function(...)`	模型的预测效果提升
RF	`revenue = revenue_function(...)`	实际支付金额
Shapley	`allocate_revenue(...)`	卖家的边际贡献分配收益

文章从理论上提出并分析了一个**满足以下四个核心性质**的机制：

- **真诚性（Truthfulness）**：买家会被激励说出其真实的估值；
- **收益最大化（Revenue Maximization）**：定价机制在长期下具有零后悔性能；
- **公平分配（Shapley Fairness + Robustness to Replication）**：卖家根据特征对精度提升的真实贡献获得报酬，且机制不被复制数据操纵；

- **计算高效（Computational Efficiency）**：所有机制均可在合理时间复杂度下在线运行。

为实现上述目标，设计了如下的关键模块（将要在程序中全部实现）

1. **基于Myerson机制的支付规则**，用于确保买家报价真实；
2. **乘性权重更新法（MWU）**，用于实现价格的动态调整，保证长期最优；
3. **Shapley值近似算法与鲁棒修正机制**，用于公平且抗复制地分配收益；
4. **基于加噪或子采样的特征分配策略**，使市场能调控数据“质量”，实现差异化售卖。

基于以上，整个市场流程设计为一个实时在线机制，每当一个新买家到达时，市场根据当前的价格策略进行以下步骤（ `main.py` 中串联）

```
买家进场（带着估值μ、预测任务Y）
         ↓

市场使用 MWU 算法 → 给出当前价格p_n
         ↓

买家出价 b = μ

         ↓

市场分配数据（加噪、截断等）X_tilde
模型训练 → 得到预测结果 Y_hat

         ↓

 计算准确率提升 G
 根据 RF 收费（Myerson）
         ↓

更新价格策略（MWU 权重调整）
         ↓

用 Shapley 分配给每个特征的收益

         ↓
市场统计分析（价格演化、卖家收益）
```

为方便理解，接下来我们将用一个实例清晰的展示论文中阐释的完整流程

实例

卖家（2人）：提供数据特征

卖家 A 提供特征 `X[:,0]` ，表示“天气温度”

卖家 B 提供特征 `X[:,1]` ，表示“人流量”

买家（3人）：有预测任务与估值

买家编号	任务内容	真实估值 μ	特征 X（2列）	标签 Y
Buyer 1	预测冰淇淋销量	80.0	$\begin{bmatrix} 30 & 50 \\ 32 & 55 \end{bmatrix}$	$[100, 110]$
Buyer 2	预测超市客流峰值	60.0	$\begin{bmatrix} 25 & 100 \\ 26 & 110 \end{bmatrix}$	$[200, 220]$
Buyer 3	预测咖啡销量	40.0	$\begin{bmatrix} 15 & 20 \\ 18 & 25 \end{bmatrix}$	$[80, 85]$

我们把 X 理解为特征矩阵，2 列对应 2 个卖家的数据。

模拟流程：

(1) Buyer 1 到达（预测冰淇淋销量）

- 估值：$\mu = 80 \to$ 每多提升 1 的准确率，愿意支付 80
- 报价：$b = \mu = 80$
- 市场当前定价：假设 $p_1 = 60$（由 MWU 定出来）

发生：

- $b > p$，买家获得了完整的 X（无加噪）。使用 X 训练模型 → 得到预测 Y_hat
- 设 gain = 0.85（预测准确率提升 85%），revenue 由 RF* 计算，约为：

$$\mathrm{RF}^* = b \cdot G - \int_0^b G(z)\,dz \approx 80 \cdot 0.85 - \text{area under gain curve} \Rightarrow \text{revenue} = 65$$

- 分配收益（Shapley）： 计算特征 0 和 1 的边际贡献， 设特征 0 对提升贡献 70%，特征 1 贡献 30%。 卖家 A 得到 $45.5，B 得到 $19.5

```
shapley_weights = {0: 0.7, 1: 0.3}
revenue = 65
seller_revenue[0] += 45.5
seller_revenue[1] += 19.5
```

(2) Buyer 2 到达（预测超市客流）

- mu = 60, b = 60 当前价格 p_2 = 70（MWU 提高了价格）
- 因为 b < p，X 被部分掩盖，例如将特征变为：

$$\tilde{X} = \begin{bmatrix} 0 & 100 \\ 0 & 110 \end{bmatrix}$$

→ 特征 0（天气）被屏蔽，买家只能用人流量特征预测
- 预测提升下降为 gain = 0.45，收费 RF ≈ 38 ，Shapley 分配： 只有特征 1 有效 → 卖家 B 拿走全部 38

(3) Buyer 3 到达（预测咖啡限量）

- mu = 40，b = 40，当前 p_3 = 30（MWU 降低价格）
- X 完整分配，模型预测结果提升 gain = 0.7 revenue = 28
- Shapley 权重 {0: 0.6, 1: 0.4}：卖家 A 得到 16.8，B 得到 11.2

最终结果

```
市场价格动态：
买家 1: p = 60.00
买家 2: p = 70.00
买家 3: p = 30.00

卖家总收益分配（基于边际贡献）：
特征 0: 收益 = 62.3
特征 1: 收益 = 68.7
```

三、关键定理的描述

Truthfulness

在该论文的模型中，需要满足诚实性，即每个买家的最优报价策略为其估值，可以表示为

$$\mu_n = \arg\max_{z \in \mathbb{R}_+} \mathcal{U}(z, Y_n)$$

满足这个性质的前提是函数$\mathcal{AF}$的单调性：即对于任意$b^{(1)} < b^{(2)}$，成立

$$\mathcal{G}(Y_n, \mathcal{M}(\widetilde{X}_M^{(1)})) \leq \mathcal{G}(Y_n, \mathcal{M}(\widetilde{X}_M^{(2)}))$$

证明

对单调性假设进行化简，将噪声后的特征向量展开：

$$\mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(b_1, p_n))) \leq \mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(b_2, p_n)))$$

展开函数 $\mathcal{U}(z, Y_n)$

$$\mathcal{U}(z, Y_n) = \mu_n \cdot \mathcal{G}(Y_n, \hat{Y}_n) - \mathcal{RF}^*(p_n, z, Y_n)$$

同时已知

$$\mathcal{RF}^*(p_n, b_n, Y_n) = b_n \cdot \mathcal{G}(Y_n, \hat{Y}_n) - \int_0^{b_n} \mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(z, p_n)))\, dz.$$

代入上式

$$\mathcal{U}(z, Y_n) = \mu_n \cdot \mathcal{G}(Y_n, \hat{Y}_n) - \left[z \cdot \mathcal{G}(Y_n, \hat{Y}_n) - \int_0^{z} \mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(t, p_n)))\, dt\right].$$

$$\mathcal{U}(z, Y_n) = (\mu_n - z) \cdot \mathcal{G}(Y_n, \hat{Y}_n) + \int_0^{z} \mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(t, p_n)))\, dt.$$

接下来我们求z在什么时候能够使函数为最大值。 因此对 $\mathcal{U}(z, Y_n)$求导：

$$\frac{d}{dz}\mathcal{U}(z, Y_n) = -\mathcal{G}(Y_n, \hat{Y}_n) + \mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(z, p_n))) + (\mu_n - z) \cdot \frac{d}{dz}\mathcal{G}(Y_n, \hat{Y}_n).$$

由定义知， $\mathcal{G}(Y_n, \hat{Y}_n) = \mathcal{G}(Y_n, \mathcal{M}(\mathcal{AF}^*(z, p_n)))$，因此上式只需要保留最后一项：

$$\frac{d}{dz}\mathcal{U}(z, Y_n) = (\mu_n - z) \cdot \frac{d}{dz}\mathcal{G}(Y_n, \hat{Y}_n).$$

而由单调性假设，我们可以知道 $\mathcal{G}(Y_n, \hat{Y}_n)$单调，因此可以判断

$$\frac{d}{dz}\mathcal{G}(Y_n, \hat{Y}_n) \geq 0.$$

因此当$\mu_n = z$时，$\mathcal{U}(z, Y_n)$取最大值

Revenue Maximization

在论文的3.2中给出了收益最大化的定义：

$$\lim_{N \to \infty} \frac{1}{N} \left[\sup_{\{(b_n, Y_n)\}} \left(\sup_{p^* \in \mathbb{R}_+} \sum_{n=1}^{N} \mathcal{RF}(p^*, b_n, Y_n) - \sum_{n=1}^{N} \mathcal{RF}(p_n, b_n, Y_n)\right)\right] = 0$$

即当N趋近于无穷时，事后最佳价格与设置价格的平均差值趋近于0，此时收益最大。而在满足5.1中的假设1、3、4的前提下可以实现该条件。

假设1：对于任意$b^{(1)} < b^{(2)}$，成立

$$\mathcal{G}(Y_n, \mathcal{M}(\widetilde{X}_M^{(1)})) \leq \mathcal{G}(Y_n, \mathcal{M}(\widetilde{X}_M^{(2)}))$$

假设3：收益函数$\mathcal{RF}$满足$lipschitz$ 对于任意的 $Y_n, b_n, p^{(1)}, p^{(2)}$:

$$|\mathcal{RF}^*(p^{(1)}, b_n, Y_n) - \mathcal{RF}^*(p^{(2)}, b_n, Y_n)| \leq \mathcal{L}|p^{(1)} - p^{(2)}|.$$

假设4：报价b_n全部在一个封闭有界区间B

证明

已知算法1的权重更新为$w_{n+1}^i = w_n^i(1 + \delta g_n^i)$ ，通过改变权重来调整市场价格的出现概率。 而总的市场权重可以化简为

$$W_{n+1} = \sum_i w_{n+1}^i = \sum_i w_n^i(1 + \delta g_n^i) = W_n + \delta \sum_i w_n^i g_n^i = W_n(1 + \delta g_n^{\text{alg}})$$

同时我们又可以知道第一轮的权重为$W_1 = |\mathcal{B}_{\text{net}}(\epsilon)|$ 因此我们可以得到递推式：

$$W_{N+1} = W_1 \prod_{n=1}^{N}(1 + \delta g_n^{\text{alg}}) = |\mathcal{B}_{\text{net}}(\epsilon)| \cdot \prod_{n=1}^{N}(1 + \delta g_n^{\text{alg}})$$

取对数，进行放缩可得：

$$\log W_{N+1} = \log |\mathcal{B}_{\text{net}}(\epsilon)| + \sum_{n=1}^{N} log(1+\delta g_n^{\text{alg}}) \leq \log |\mathcal{B}_{\text{net}}(\epsilon)| + \delta \sum_{n=1}^{N} g_n^{\text{alg}}$$

又由W_{N+1}的定义知，

$$\log W_{N+1} \geq \log w_{N+1}^i = \sum_{n=1}^{N} \log(1+\delta g_n^i)$$

再次进行放缩可得：

$$\log W_{N+1} \geq \delta \sum_{n=1}^{N} g_n^i - \delta^2 N$$

结合前式，可以得到关于g_n^i、g_n^{alg}的不等式

$$\delta \sum_{n=1}^{N} g_n^{\text{alg}} \geq \delta \sum_{n=1}^{N} g_n^i - \log |\mathcal{B}_{\text{net}}(\epsilon)| - \delta^2 N$$

两边除以 δN 并设 $\delta = \sqrt{\log |\mathcal{B}_{\text{net}}(\epsilon)|/N}$：

$$\frac{1}{N}\sum_{n=1}^{N} g_n^{\text{alg}} \geq \frac{1}{N}\sum_{n=1}^{N} g_n^i - 2\sqrt{\frac{\log |\mathcal{B}_{\text{net}}(\epsilon)|}{N}} \quad (2)$$

由假设3和$\mathcal{B}_{net}$的定义我们可以知道，归一化收益满足：

$$|g_n^{\text{opt}} - g_n^i| \leq \mathcal{L}\epsilon/\mathcal{B}_{\max}$$

因此可以代入上式

$$\frac{1}{N}\sum_{n=1}^{N} g_n^{\text{alg}} \geq \frac{1}{N}\sum_{n=1}^{N} g_n^{\text{opt}} - 2\sqrt{\frac{\log |\mathcal{B}_{\text{net}}(\epsilon)|}{N}} - \frac{\mathcal{L}\epsilon}{\mathcal{B}_{\max}}$$

选择 $\epsilon = \frac{1}{\mathcal{L}\sqrt{N}}$，并利用 $|\mathcal{B}_{\text{net}}(\epsilon)| \leq \frac{3\mathcal{B}_{\max}}{\epsilon}$：

$$\frac{1}{N}\mathbb{E}[\mathcal{R}(N)] \leq 2\mathcal{B}_{\max}\sqrt{\frac{\log(3\mathcal{B}_{\max}\mathcal{L}\sqrt{N})}{N}} + \frac{\mathcal{B}_{\max}}{\sqrt{N}}$$

化简，得到最终结论：

$$\frac{1}{N}\mathbb{E}[\mathcal{R}(N)] \leq C\mathcal{B}_{\max}\sqrt{\frac{\log(\mathcal{B}_{\max}\mathcal{L}\sqrt{N})}{N}}$$

Fairness

Shapely Fairness

property3.3定义了模型中的公平性，通过对$\mathcal{PD}$进行约束来保证卖家间的公平分配：

1. **balance**：$\sum_{m=1}^{M} \psi_n(m) = 1$
2. **Symmetry**：$\forall\ m, m' \in [M], \forall\ S \subset [M] \setminus \{m, m'\}$, if $\mathcal{PD}(S \cup m, Y_n) = \mathcal{PD}(S \cup m', Y_n)$, then $\psi_n(m) = \psi_n(m')$.
3. **Zero Element**：$\forall\ m \in [M], \forall\ S \subset [M]$, if $\mathcal{PD}(S \cup m, Y_n) = \mathcal{PD}(S, Y_n)$, then $\psi_n(m) = 0$.
4. **Additivity**：支付分配函数 $\mathcal{PD}([M], Y_n^{(1)}), \mathcal{PD}([M], Y_n^{(2)})$ 的输出分别为$\psi_n^{(1)}$、$\psi_n^{(2)}$，令 ψ_n' 为$\mathcal{PD}([M], Y_n^{(1)} + Y_n^{(2)})$的输出，那么$\psi_n' = \psi_n^{(1)} + \psi_n^{(2)}$。

由于shapely值的计算需要$O(2^M)$的时间复杂度，因此实际算法2中采用了随机随机采样方法将复杂度降至$O(M^2)$，在定理5.3中，证明了算法2的输出至少有$1-\delta$的概率满足$||\psi_{n,\text{shapley}} - \hat{\psi}_n||_\infty < \epsilon$

Robustness to Replication

而由于现实中数据是可以零成本复制的，因此常规的shapely值会导致复制者获得更多的收益，因此需要分配机制对复制行为由鲁棒性，因此提出了property3.4来保障鲁棒性。

property3.4：对于所有特征 $m \in [M]$，用 m_i^+ 表示 m 的第 i 个复制副本（即 $X_{m_i^+} = X_m$）。定义包含原始特征和复制特征的集合为：

$$[M]^+ = \bigcup_m \left(m \cup_i m_i^+\right)$$

令 $\psi_n^+ = \mathcal{PD}([M]^+, Y_n)$ 为对扩展特征集的支付分配函数输出。则称一个市场机制是 ϵ-抗复制的，如果对于所有 $n \in [N]$ 和所有 Y_n，支付分配函数 $\mathcal{PD}$ 满足：

$$\psi_n^+(m) + \sum_i \psi_n^+(m_i^+) \leq \psi_n(m) + \epsilon$$

其中：

- $\psi_n(m)$ 是原始市场（无复制）中对特征 m 的分配值
- $\psi_n^+(m)$ 和 $\psi_n^+(m_i^+)$ 分别是扩展市场中对原始特征及其复制副本的分配值
- $\epsilon \geq 0$ 为允许的误差容限

在定理5.4中，证明了在假设2的条件下，复制数据不会改变预测精度，算法3能有效抑制数据复制，确保分配公平。

5.3证明

根据shapely值的定义，将其转化为

$$\psi_{n,shapley}(m) = E_{\sigma \sim Unif(\sigma_{S_n})}[G_n(Y_n, M_n(X_{[\sigma<m]\cup m})) - G_n(Y_n, M_n(X_{[\sigma<m]}))]$$

而随机变量ψ̂_n^k(m)按以下方式分布：

$$P(\hat{\psi}_n^k(m) = G_n(Y_n, M(X_{[\sigma_k<m]\cup m})) - G_n(Y_n, M(X_{[\sigma_k<m]})); \sigma \in \sigma_{S_n}) = 1/S_n!$$

然后我们有：

$$E[\hat{\psi}_n(m)] = (1/K)\Sigma_{k=1}^K E[\hat{\psi}_n^k(m)] = \psi_{n,shapley}$$

由于$\hat{\psi}_n(m)$的支持在0和1之间有界，且$\hat{\psi}_n^k(m)$是独立同分布的，我们可以应用Hoeffding不等式得到以下界限：

$$P(|\psi_{n,shapley} - \hat{\psi}_n(m)| > \varepsilon) < 2exp(-2\varepsilon^2/K)$$

通过对所有$m \in S_n \leq M$应用联合界，我们有：

$$P(\|\psi_{n,shapley} - \hat{\psi}_n\|_\infty > \varepsilon) < 2Mexp(-2\varepsilon^2/K)$$

设$\delta = 2Mexp(-2\varepsilon^2/K)$并解出$K$:

$$K > (Mlog(2/\delta))/(2\varepsilon^2)$$

5.4证明

假设2:对于任意子集 $S \subset [M]$、任意预测任务 Y_n 以及任意特征 $m \in S$，若 m_i^+ 表示特征 m 的第 i 次复制（即 $X_{m,i}^+ = X_m$），且 $S^+ = \cup_m(m \cup_i m_i^+)$ 表示原始特征及其复制特征的集合，则以下等式成立：

$$\mathcal{G}(Y_n, \mathcal{M}(X_S)) = \mathcal{G}(Y_n, \mathcal{M}(X_{S^+}))$$

在假设2的前提下，进行证明。

设$S = X_1, X_2, ..., X_K$表示没有复制的原始分配特征集。设$S^+ = X_{(1,1)}, X_{(1,2)}, ..., X_{(1,c_1)}, X_{(2,1)}, ..., X_{(K,c_K)}$（其中$c_i \in N$）是附加了原始特征复制版本的S的扩展版本，即$X_{(m,i)}$是特征$X_m$的第$(i-1)$次复制副本。

设$\hat{\psi}, \hat{\psi}^+$分别是S, S^+的算法3第1步的各自输出。卖家m在原始和复制场景中的总收入分配由以下给出：

$$\psi(m) = \hat{\psi}(m) \exp\left(-\lambda \sum_{j \in S \setminus m} \mathcal{SM}(X_m, X_j)\right)$$

$$\psi^+(m) = \sum_{i=1}^{c_m} \hat{\psi}^+((m,i)) \exp\left(-\lambda \sum_{(j,k) \in S^+ \setminus (m,i)} \mathcal{SM}(X_{(m,i)}, X_{(j,k)})\right)$$

对于性质3.4成立，需要证明$\psi^+(m) \leq \psi(m) + \varepsilon$。我们有：

$$\sum_{i \in c_m} \hat{\psi}^+((m,i)) \exp\left(-\lambda \sum_{(j,k) \in S_m^+ \setminus \{(m,i)\}} \mathcal{SM}(X_{(m,i)}, X_{(j,k)})\right)$$

$$\leq \sum_{i\in c_m} \hat{\psi}^+((m,i)) \exp\left(-\lambda \sum_{l\in[c_m]\setminus i} \mathcal{SM}(X_{(m,i)}, X_{(m,l)})\right) \exp\left(-\lambda \sum_{l\in S_m\setminus\{m\}} \mathcal{SM}(X_{(m,i)}, X_{(l,1)})\right)$$

$$= \sum_{i\in c_m} \hat{\psi}^+((m,i)) \exp\left(-\lambda(c_m-1)\right) \exp\left(-\lambda \sum_{l\in S_m\setminus\{m\}} \mathcal{SM}(X_m, X_l)\right)$$

$$\leq c_m \left(\hat{\psi}^+((m,1)) + \frac{1}{3}\epsilon\right) \exp\left(-\lambda(c_m-1)\right) \exp\left(-\lambda \sum_{j\in S\setminus\{m\}} \mathcal{SM}(X_m, X_j)\right)$$

$$\leq c_m \left(\hat{\psi}^+((m,1)) + \frac{1}{3}\epsilon\right) \exp\left(-\lambda(c_m-1)\right) \exp\left(-\lambda \sum_{j\in S\setminus\{m\}} \mathcal{SM}(X_m, X_j)\right)$$

因此需要证明$c_m(\hat{\psi}^+((m,1)) + (1/3)\varepsilon)exp(-\lambda(c_m-1)) \leq \psi(m) + \varepsilon \forall c_m \in N$。我们有：

$$c_m \exp\left(-\lambda(c_m-1)\right) \left(\hat{\psi}^+((m,1)) + \frac{1}{3}\epsilon\right)$$

$$\leq c_m \exp\left(-\lambda(c_m-1)\right) \left(\frac{\psi(m)}{c_m} + \frac{2}{3}\epsilon\right)$$

$$\leq c_m \exp\left(-\lambda(c_m-1)\right) \left(\psi(m) + \frac{2}{3}\epsilon\right)$$

$$\leq c_m \exp\left(-\lambda(c_m-1)\right) \left(\hat{\psi}(m) + \epsilon\right)$$

$$\leq \left(\hat{\psi}(m) + \epsilon\right)$$

其中通过选取$\lambda = \log(2)$使得$c_m \exp\left(-\lambda(c_m-1)\right) \leq 1 \ \forall c_m \in \mathbb{N}$。

ψ_n继续满足性质3.3的条件2-4以ϵ精度的事实，直接来自定理5.3和ψ_n的构造。

命题5.1

如果市场中卖家的身份是匿名的，则性质3.3的平衡条件和性质3.4不能同时成立。 我们通过举反例来说明，预设三种场景。

在第一种场景中，市场恰好有两个卖家A, B，每个都出售相同的特征，即$X_A = X_B$。根据性质3.3的条件1和2，两个卖家必须获得相同的分配，即$\psi_1(A) = \psi_1(B) = \frac{1}{2}$，对于任何预测任务。

现在考虑第二种场景，市场还是相同的两个卖家A和B，但这次卖家A复制其特征一次并在市场上再次出售为A'。由于假设卖家的身份是匿名的，为满足性质3.3的"平衡"条件，我们需要$\psi_2(A) = \psi_2(B) = \psi_2(A') = \frac{1}{3}$。因此，卖家$A$的总分配为$\psi_2(A) + \psi_2(A') = \frac{2}{3} > \frac{1}{2} = \psi_1(A)$，即性质3.4不成立。

最后考虑第三种场景，市场包含三个卖家A, B和C，每个都出售相同的特征，即$X_A = X_B = X_C$。很容易看出，为达到"平衡"，我们需要$\psi_3(A) = \psi_3(B) = \psi_3(C) = \frac{1}{3}$。

由于市场无法区分A'和C，我们要么有平衡性，要么有性质3.4即"抗复制鲁棒性"。

Efficiency

property3.5定义了高效性：对于每个步骤都能以多项式时间内运行，且计算复杂度不能随着N增长。 该性质表明了当前算法的可应用性，而在5.5中验证了算法的高效性。

算法	时间复杂度	说明
$\mathcal{AF}^*$	$O(M)$	分配函数（如添加噪声）线性时间
$\mathcal{RF}^*$	$O(M)$	收入函数（调用 $\mathcal{G}, \mathcal{M}$ 常数次）
$\mathcal{PF}^*$	$O(M)$	价格更新（基于Multiplicative Weights）
$\mathcal{PD}_a^*$ (Algorithm 2)	$O(M^2)$	Shapley近似（随机采样 $K \propto M$）
$\mathcal{PD}_b^*$ (Algorithm 3)	$O(M^2)$	鲁棒性分配（增加相似性惩罚）

四、实现的核心算法介绍

为实现第一部分提到的完整市场机器学习模型交易流程， 我们设计项目结构如下：

```
data-marketplace/
│
├── data/                    # 存放模拟数据（买家传入信息）
│   └── buyer_data.json
│
├── models/                  # 机器学习模型 预测部分
│   └── learner.py
│
├── market/                  # 拍卖、定价、收益分配逻辑
│   ├── auction.py           # 诚实机制（AF + RF）
│   ├── pricing.py           # MWU 动态定价
│   └── revenue.py           # Shapley 收益分配
│
├── utils/                   # 公共函数
│   └── metrics.py           # 指标数学计算
│   └── io.py                # 加载测试数据
│
└── main.py                  # 主程序入口：输入一个买家 → 运行三步流程
```

接下来我们将分为三个核心功能模块，分别介绍关键算法：

(1) 诚实的机器学习模型拍卖(`auction.py`)

该模块需要实现的功能如下

1. 接收买家提交的预测任务 Y_n 和出价 b_n
2. 分配特征数据 $\widetilde{X}_M = AF(p_n, b_n; X_M)$，即根据信噪比决定特征质量（添加噪声或mask）
3. 用模型 M 进行预测 $\hat{Y}_n = M(\widetilde{X}_M)$,计算预测增益 $G(Y_n, \hat{Y}_n)$
4. 利用 Myerson Payment Function 计算支付金额 RF^*

算法 1：Allocation Functions (Buyer’s Perspective)

数据分配函数（AF∗）： 根据论文Assumption 1， 数据分配的“质量”应与买家的出价相关，若出价高于市场定价，则可获得完整无噪声数据；反之，数据会被适当降质（例如添加噪声或随机掩码）。

在论文的 Example 4.1 中，AF∗ 被具体实现为对输入特征添加噪声：

$$\tilde{X}_j(t) = X_j(t) + \max(0, p_n - b_n) \cdot \mathcal{N}(0, \sigma^2)$$

即：若 $b_n < p_n$，则按比例添加高斯噪声。

实现如下：

```python
def allocation_function(X: np.ndarray, pn: float, bn: float, noise_std: float = 1) -> np.ndarray:
    if bn >= pn:
        return X.copy()
    else:
        noise_scale = min(1.0, max(0.1, (pn - bn) / (pn)))
        noise = np.random.normal(0, noise_std * noise_scale, size=X.shape)
        return X + noise
```

其中，我们通过不断测试基于选定训练模型（ `model = LinearRegression()` 线性拟合 ） 下输入数据，即改变buyer.json中传入的买家信息，发现添加噪声过大过小都会导致和预期实际效果有差别：

- 噪声过大：会导致数据完全失去原有相关性，采用$1 - MSE$方式得到的预期收益 `G` 有可能因为均方差过大直接变为负数!
- 噪声过小：即使噪声干扰下的X_alloc依旧能被训练达到较高的拟合度（预期增益 `G` 未明显削减），尤其是对于鲁棒性更强大的训练模型（猜测）！导致出价 `bn` 较小的代价变得微乎其微，Myerson支付函数 $\mathrm{RF}^*(p_n, b_n, Y_n) = b_n \cdot G(Y_n, \hat{Y}_n) - \int_0^{b_n} G(Y_n, \hat{Y}(z))dz$ 右式 减数(累积边际增益积分) 和 被减数（边际增益 × 出价）大小相近，导致最后买家需要支付的revenue非常小，不符合现实情况

因此我们针对噪声做了如下优化：相当于对高斯噪声进行合理化限制

- 使用 $(pn - bn)/(pn)$ 归一化比值控制噪声幅度；
- 将噪声强度限制在合理范围 $[0.1, 1.0]$，避免噪声过小或过大，提升了鲁棒性；

这保证了 AF* 是一个大样本下的单调函数（出价越高，预测质量越高），满足论文对 truthful bidding 的关键假设。

算法 2：Revenue Functions (Buyer's Perspective)

论文式 (3) 给出了基于 Myerson 机制的支付函数：

$$\mathrm{RF}^*(p_n, b_n, Y_n) = b_n \cdot G(Y_n, \hat{Y}_n) - \int_0^{b_n} G(Y_n, \hat{Y}(z))dz$$

- 第一项：买家按其出价乘以模型真实的预测提升付费；
- 第二项：从 0 到出价的累计精度提升积分（作为折扣）。
- $G(Y, \hat{Y})$：衡量预测质量的"增益函数"，可采用分类准确率或 $1 - \mathrm{RMSE}$；

实现如下：

```python
def revenue_function(X, Y, pn, bn, model_func, gain_func, steps=10):
    # 第一项：计算在出价 bn 下的预测效果
    X_alloc = allocation_function(X, pn, bn)
    Y_hat = model_func(X_alloc, Y)
    G_bn = gain_func(Y, Y_hat)

    # 第二项：使用梯形法近似积分 ∫₀^bn G(z) dz
    zs = np.linspace(0, bn, steps)
    integral = 0.0
    prev_G = None
    for z in zs:
        X_z = allocation_function(X, pn, z)
        Y_z_hat = model_func(X_z, Y)
        G_z = gain_func(Y, Y_z_hat)
        if prev_G is not None:
            integral += 0.5 * (G_z + prev_G) * (bn / (steps - 1))
        prev_G = G_z

    revenue = bn * G_bn - integral
    return revenue, X_alloc, Y_hat, G_bn, integral
```

使用 `train_and_predict` 训练模型并获得预测 $\hat{Y}$，这里我们使用线性拟合的模型 `model = LinearRegression()` 。同时使用 `gain_function` 计算精度；

```python
def gain_function(y_true, y_pred, task='regression'):
    if task == 'regression':
        return 1.0 - rmse(y_true, y_pred)
    elif task == 'classification':
        return np.mean(y_true == y_pred)
    else:
        raise ValueError("Unsupported task type.")
```

积分项我们采用梯形法数值求和来近似积分计算，通过 `steps` 参数来自行调控微分近似精度

该函数输出的不仅包括最终的 revenue，我们还增加了预测输出、分配后的特征 $\tilde{X}$、增益值与积分项，便于我们后续调试,以及更重要的是与后续的定价权重更新函数、主函数等提供传输关键量的接口

(2) MWU 算法动态定价(`pricing.py`)

该模块需要实现的功能如下

1. 维护一个价格候选集合（通过离散化 bidder bid 范围构造 epsilon-net）
2. 初始化每个价格候选的权重为 1
3. 每轮 buyer 到达时：
   - 根据当前权重采样一个价格 p_n
   - 用该价格进行拍卖（运行第一步流程），得到收入 r_n

- 根据该轮的 revenue 更新权重，设 $g_i^n = \frac{\mathrm{RF}(c_i, b_n, Y_n)}{B_{\max}}$：第 i 个价格在当前买家的表现，更新第 i 个价格的权重：

$$w_i^{n+1} = w_i^n \cdot (1 + \delta \cdot g_i^n)$$

4. 收敛时达到 regret 最小化

```
def __init__(self, B_max: float, L: float, N: int, epsilon: Optional[float] = None):
    self.B_max = B_max
    self.L = L
    self.N = N
```

初始化函数接收四个参数：B_max表示买家可能的最大估值，L是收益函数的Lipschitz常数，N是预期的买家总数，epsilon是可选的网格精度参数。这些参数共同决定了算法的学习能力和定价精度。

初始化

```
if epsilon is None:
    epsilon_value = 1 / (L * np.sqrt(N))
else:
    epsilon_value = epsilon
self.epsilon = float(epsilon_value)

self.candidates = np.arange(0, B_max + self.epsilon, self.epsilon)
self.num_candidates = len(self.candidates)
self.weights = np.ones(self.num_candidates)
self.delta = np.sqrt(np.log(self.num_candidates) / N)
```

算法自动计算价格网格的精度epsilon，当用户未提供时，根据Lipschitz常数L和买家数量N自动确定。

算法创建从0到B_max的价格候选网格，初始化所有候选价格的权重为1。学习率δ根据候选价格数量和总买家数动态调整，这种设置保证了权重更新的稳定性。

价格选择

```
def choose_price(self) -> float:
    total_weight = np.sum(self.weights)
    prob = self.weights / total_weight
    chosen_idx = np.random.choice(self.num_candidates, p=prob)
    return float(self.candidates[chosen_idx])
```

价格选择采用基于权重的随机采样策略。每个候选价格被选中的概率与其当前权重成正比，首先计算所有权重的总和，然后将每个权重转换为概率值，最后按这些概率随机选择一个价格。

权重更新

```
def update_weights(self, chosen_price: float, b_n: float, Y: np.ndarray,
                   X: np.ndarray, revenue_func) -> None:
    for i, c_i in enumerate(self.candidates):
        RF_i = revenue_func(X, Y, c_i, b_n)
        g_i = RF_i / self.B_max
        self.weights[i] *= (1 + self.delta * g_i)
```

对于每个候选价格，计算其在该交易中的潜在收益RF_i，将其归一化为g_i后，按比例增加权重。表现越好的价格获得的权重增加越大，使算法能够收敛于高收益价格区域，实现收益最大化。

(3) 收益分配(`revenue.py`)

该模块需要实现的功能如下

利用 Shapley sampling 估计每个特征的边际贡献：

- 随机排列 seller 顺序
- 对每种排列，计算某特征的“加入前后”预测增益
- 多次采样后求平均，得到近似 Shapley 值

类结构概览： `DataMarketplace`

```python
class DataMarketplace:
    def __init__(self, X: np.ndarray, Y: np.ndarray, lambda_penalty: float = 0.1):
```

- 这是整个数据市场模型的载体。
- 参数含义：
  - `X`：所有特征数据，维度为 `(n_samples, n_features)`，对应论文中的 `X_M`。
  - `Y`：所有标签（预测目标），对应论文中的 `Y_n`。
  - `lambda_penalty`：用于在 PD*（Shapley-Robust）机制中惩罚特征冗余，初始化为0，尚未启用。

> 对应论文第 2.1 节：“Feature set M”, “Label Y”, 构成训练数据的基础。

`marginal_gain()` —— 特征的边际增益

```python
def marginal_gain(self, subset, feature, pn, bn):
```

- **论文对应**：第 4.2 节，Shapley 值定义中的边际贡献计算：

$$\Delta_f(S) = RF(S \cup \{f\}) - RF(S)$$

- 用于构造 Shapley 值的关键步骤，表示某一特征加入到子集中的收益提升。

`approximate_shapley()` —— Shapley 值近似算法

```python
def approximate_shapley(self, pn, bn, K=1000):
```

- **论文对应**：算法 2，PD-A：Permuted Data Shapley（Shapley 近似）

> 原文：“Sample K permutations of M, compute marginal gain of each feature in the permutation, average over K.”

实现细节：

1. `perm = random.sample(...)`：随机打乱特征顺序，得到排列 `π`；
2. 对于排列 `π` 中的每个特征 `m`，计算其边际增益；
3. 累积后除以 `K` 得到近似 Shapley 值。

> 对应论文中定理 5.3：该算法近似精度随 `K` 上升而收敛。

`allocate_revenue()` —— 收益归因与归一化分配

```python
def allocate_revenue(self, pn, bn, method='approximate', K=1000, total_revenue=1.0, enforce_non_negative=True):
```

- **论文对应**：第 3.3 节 Shapley Fairness 分配原理

功能解析：

1. **调用 Shapley 分配方法**：

   ```python
   shapley_values = self.approximate_shapley(...)
   ```

2. **惩罚项（尚未启用）**：

   ```python
   penalty = self.lambda_penalty * 0  # 预留接口
   ```

3. **非负性处理**：

   ```python
   if enforce_non_negative:
       payouts = {i: max(v, 0.0) for i, v in payouts.items()}
   ```

防止由于数值误差造成的负分配。

4. **归一化为总收益**：

```python
normalized_payouts = {i: (v / total) * total_revenue for i, v in payouts.items()}
```

确保所有特征收益之和为指定总收入。

五、实验结果与分析

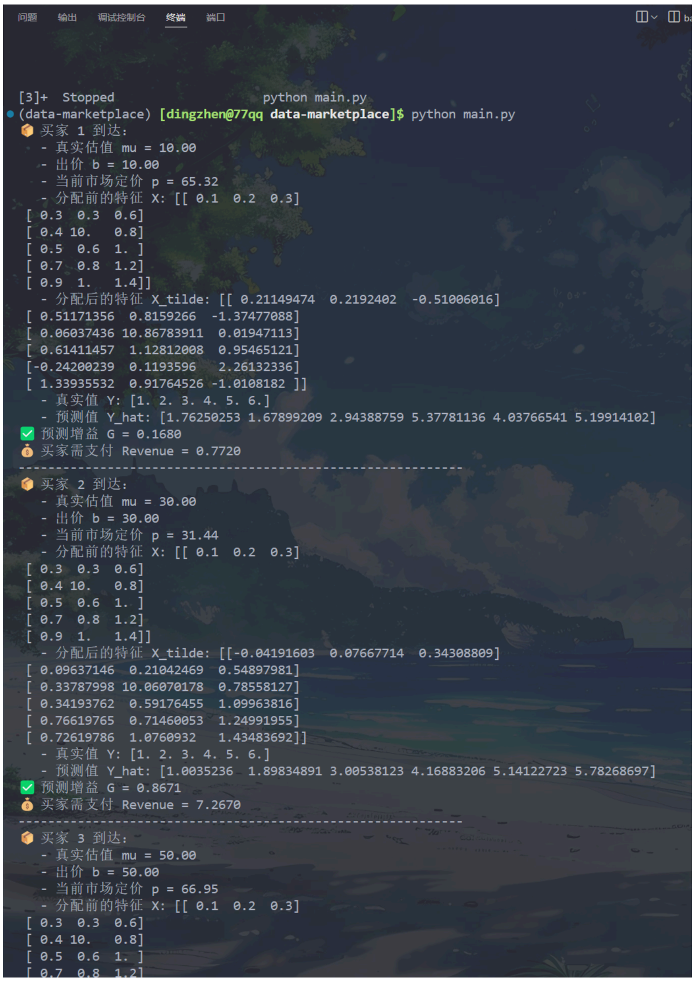
下面是一次运行结果（6个买家）

```
📦 买家 1 到达:
  - 真实估值 mu = 10.00
  - 出价 b = 10.00
  - 当前市场定价 p = 65.32
  - 分配前的特征 X: [[ 0.1  0.2  0.3]
 [ 0.3  0.3  0.6]
 [ 0.4 10.   0.8]
 [ 0.5  0.6  1. ]
 [ 0.7  0.8  1.2]
 [ 0.9  1.   1.4]]
  - 分配后的特征 X_tilde: [[ 0.21149474  0.2192402  -0.51006016]
 [ 0.51171356  0.8159266  -1.37477088]
 [ 0.06037436 10.86783911  0.01947113]
 [ 0.61411457  1.12812008  0.95465121]
 [-0.24200239  0.1193596   2.26132336]
 [ 1.33935532  0.91764526 -1.0108182 ]]
  - 真实值 Y: [1. 2. 3. 4. 5. 6.]
  - 预测值 Y_hat: [1.76250253 1.67899209 2.94388759 5.37781136 4.03766541 5.19914102]
✅ 预测增益 G = 0.1680
💰 买家需支付 Revenue = 0.7720
------------------------------------------------------------
📦 买家 2 到达:
  - 真实估值 mu = 30.00
  - 出价 b = 30.00
  - 当前市场定价 p = 31.44
  - 分配前的特征 X: [[ 0.1  0.2  0.3]
 [ 0.3  0.3  0.6]
 [ 0.4 10.   0.8]
 [ 0.5  0.6  1. ]
 [ 0.7  0.8  1.2]
 [ 0.9  1.   1.4]]
  - 分配后的特征 X_tilde: [[-0.04191603  0.07667714  0.34308809]
 [ 0.09637146  0.21042469  0.54897981]
 [ 0.33787998 10.06070178  0.78558127]
 [ 0.34193762  0.59176455  1.09963816]
 [ 0.76619765  0.71460053  1.24991955]
 [ 0.72619786  1.0760932   1.43483692]]
  - 真实值 Y: [1. 2. 3. 4. 5. 6.]
  - 预测值 Y_hat: [1.0035236  1.89834891 3.00538123 4.16883206 5.14122723 5.78268697]
✅ 预测增益 G = 0.8671
💰 买家需支付 Revenue = 7.2670
------------------------------------------------------------
📦 买家 3 到达:
  - 真实估值 mu = 50.00
  - 出价 b = 50.00
  - 当前市场定价 p = 66.95
  - 分配前的特征 X: [[ 0.1  0.2  0.3]
 [ 0.3  0.3  0.6]
 [ 0.4 10.   0.8]
 [ 0.5  0.6  1. ]
 [ 0.7  0.8  1.2]
 [ 0.9  1.   1.4]]
  - 分配后的特征 X_tilde: [[0.38069874 0.15354388 0.17920563]
 [0.54315009 0.34592382 0.70235099]
 [0.3858814  9.98228627 0.97035262]
 [0.47203428 0.74626736 0.92957346]
 [0.5769036  0.68942123 1.29632301]
 [0.90059431 0.89855971 1.55921198]]
  - 真实值 Y: [1. 2. 3. 4. 5. 6.]
  - 预测值 Y_hat: [0.68459372 2.67714075 3.01336972 3.70225707 5.13178454 5.7908542 ]
✅ 预测增益 G = 0.6565
💰 买家需支付 Revenue = 12.5099
------------------------------------------------------------
📦 买家 4 到达:
  - 真实估值 mu = 70.00
  - 出价 b = 70.00
  - 当前市场定价 p = 31.03
  - 分配前的特征 X: [[ 0.1  0.2  0.3]
```

```
[ 0.3  0.3  0.6]
[ 0.4 10.   0.8]
[ 0.5  0.6  1. ]
[ 0.7  0.8  1.2]
[ 0.9  1.   1.4]]
  - 分配后的特征 X_tilde: [[ 0.1  0.2  0.3]
[ 0.3  0.3  0.6]
[ 0.4 10.   0.8]
[ 0.5  0.6  1. ]
[ 0.7  0.8  1.2]
[ 0.9  1.   1.4]]
  - 真实值 Y: [1. 2. 3. 4. 5. 6.]
  - 预测值 Y_hat: [0.84906803 2.22521263 3.00237066 3.9738569  4.97444956 5.97504223]
✅ 预测增益 G = 0.8878
💰 买家需支付 Revenue = 8.0321
------------------------------------------------------------
📦 买家 5 到达:
  - 真实估值 mu = 100.00
  - 出价 b = 100.00
  - 当前市场定价 p = 50.62
  - 分配前的特征 X: [[ 0.1  0.2  0.3]
[ 0.3  0.3  0.6]
[ 0.4 10.   0.8]
[ 0.5  0.6  1. ]
[ 0.7  0.8  1.2]
[ 0.9  1.   1.4]]
  - 分配后的特征 X_tilde: [[ 0.1  0.2  0.3]
[ 0.3  0.3  0.6]
[ 0.4 10.   0.8]
[ 0.5  0.6  1. ]
[ 0.7  0.8  1.2]
[ 0.9  1.   1.4]]
  - 真实值 Y: [1. 2. 3. 4. 5. 6.]
  - 预测值 Y_hat: [0.84906803 2.22521263 3.00237066 3.9738569  4.97444956 5.97504223]
✅ 预测增益 G = 0.8878
💰 买家需支付 Revenue = 24.8306
------------------------------------------------------------
📦 买家 6 到达:
  - 真实估值 mu = 10.00
  - 出价 b = 10.00
  - 当前市场定价 p = 66.14
  - 分配前的特征 X: [[ 0.1  0.2  0.3]
[ 0.3  0.3  0.6]
[ 0.4 10.   0.8]
[ 0.5  0.6  1. ]
[ 0.7  0.8  1.2]
[ 0.9  1.   1.4]]
  - 分配后的特征 X_tilde: [[-0.80331951 -0.64296333 -0.44708463]
[-0.65877304  0.18778284  1.95372797]
[ 1.45937627  9.64673807 -0.50364472]
[ 1.30149164  1.76709551  0.88268473]
[ 1.03535097  1.65160362  0.90881529]
[ 1.55429948  0.45929059  1.50516335]]
  - 真实值 Y: [1. 2. 3. 4. 5. 6.]
  - 预测值 Y_hat: [0.98296473 2.072631   2.96472768 4.81756422 4.37582733 5.78628504]
✅ 预测增益 G = 0.5698
💰 买家需支付 Revenue = 2.3591
------------------------------------------------------------

📊 市场价格动态:
买家 1: p = 65.32
买家 2: p = 31.44
买家 3: p = 66.95
买家 4: p = 31.03
买家 5: p = 50.62
买家 6: p = 66.14

🏦 卖家总收益分配（基于边际贡献）:
特征 0: 收益 = 25.5556
```

```
特征 1: 收益 = 1.6048
特征 2: 收益 = 28.6104
------------------------------------------------------------

📊 市场价格动态:
买家 1: p = 35.93
买家 2: p = 17.15
买家 3: p = 0.82
买家 4: p = 89.00
买家 5: p = 86.96
买家 6: p = 81.24

🏦 卖家总收益分配（基于边际贡献）:
特征 0: 收益 = 12.5762
特征 1: 收益 = 14.0319
特征 2: 收益 = 23.2290
```

下面是运行截图



总览表

买家	mu = b	市场定价 p	特征是否变形	G（预测增益）	Revenue（支付）	解读
1	10	65.32	是（严重扰动）	0.1680	0.7720	市场定价远高 → 买家几乎买不到好特征
2	30	31.44	稍有扰动	0.8671	7.2670	匹配合理，G 高，支付合适
3	50	66.95	中度扰动	0.6565	12.5099	定价偏高但仍有收益，支付较多
4	70	31.03	未扰动	0.8878	8.0321	市场定价远低 → 买家捡漏，低价买完整数据
5	100	50.62	未扰动	0.8878	24.8306	市场定价适中，G 高，支付也高
6	10	66.14	严重扰动	0.5698	2.3591	出价远小于定价，扰动严重

买家逐个详细分析

买家 1（定价远高 → G 低，支付低）

- **mu = b = 10**，**市场定价 p = 65.32**
- 特征经过严重扰动（`X_tilde` 有负值、大幅偏移）
- G = 0.1680 → 模型预测增益极低
- Revenue = 0.7720（Myerson 支付：增益小导致支付小）

论文对应解释：

> 买家出价远低于定价，使用 `AF(p, b)` 导致特征大幅降质，反映论文第 4.1 节中的 *allocation function* 设计。Myerson 支付函数鼓励买家真实出价，但无法惩罚市场定价错误。

买家 2（匹配较好 → G 高，支付中等）

- b = 30, p = 31.44 → 出价 ≈ 定价
- `X_tilde` 与 `X` 非常接近，预测增益 G = 0.8671
- Revenue = 7.2670，较合理（对应较高增益）

论文对应解释：

> 在买家预算与市场定价接近时，allocation 函数几乎不会降质，反映出论文中 allocation 的“单调性”：**出价越高 → 获取特征越完整 → G 越高 → 支付越高**。

买家 3（定价偏高 → 特征退化 → G 中等，支付大）

- b = 50, p = 66.95 → 定价略高，买家“吃亏”
- `X_tilde` 有一定扰动，但还算完整
- G = 0.6565，支付却高达 12.5

论文对应解释：

> 这反映了市场在没有动态调价机制（如 PF 定价）时的“高估问题” → 买家为部分失真的数据支付了不小代价。符合论文中 PF 机制存在的动机：**市场需要学习定价策略，以避免错估导致的效率损失**。

买家 4（市场严重低估 → 捡漏）

- b = 70, p = 31.03 → 出价远高于定价
- 获得完整特征（X_tilde = X）
- G = 0.8878，支付仅 8.03，远低于其出价

论文对应解释：

> 市场定价严重低估买家价值，导致 Myerson 函数计算的支付偏低 → **买家以极低价格买走高价值数据**。这在论文中被视为 **market inefficiency**，动机之一正是防止这种价值泄露。

买家 5（匹配较好 → G 高，支付高）

- b = 100, p = 50.62 → 定价中等偏低，买家仍愿意支付
- 完整特征，G = 0.8878，支付高达 24.8

论文对应解释：

> 定价合理时，预测性能好，支付也最大。符合论文第 4 节中 Myerson 函数的目标：**高出价、高 G → 高支付**，系统收益被充分释放。

买家 6（出价极低，定价极高 → 噪声）

- b = 10, p = 66.14 → 定价过高
- `X_tilde` 出现奇异扰动（负值、结构混乱）
- G = 0.5698，预测竟然不差，但支付仍很低（2.35）

论文对应解释：

> 即使在特征扰动较大时，某些数据维度仍保留结构，模型仍有一定预测力。但此案例体现出：出价低、定价高 → 市场分配质量差 → G 难以进一步提升。

卖家收益归因分析（Shapley）

特征	收益	说明
特征 0	25.56	中等重要
特征 1	**1.60（极低）**	几乎无贡献，可能是冗余或噪声特征
特征 2	**28.61（最高）**	明显是模型预测中最关键的特征

论文对应解释：

> 与论文 Algorithm 2 (PD-A) 相一致，Shapley 值在多次随机排列中体现出边际贡献的差异，**特征 2 获得最多收益**表明其是模型预测的核心维度，说明归因机制符合公平性（Fairness Property 3.3）。

市场定价策略观察

买家	出价 b	定价 p	定价策略偏差
1	10	65.32	**严重过高**
2	30	31.44	较匹配
3	50	66.95	偏高
4	70	31.03	**严重低估**
5	100	50.62	略偏低
6	10	66.14	严重过高

前端搭建展示

六、反思与改进

在实验数据测试过程中，我们对 `buyer.json` 数据规模进行不同维度的调整，最初测试时候我们使用的是2个买家、2个卖家、2个样本的简单情形，发现对于采用线性拟合的简单训练模型来说，这种情况输入数据很有可能导致计算得到某个买家需要支付的revenue为负数，原先我们一直认为是卖家和卖家数量过小的问题，后来在反复调试数据后，才发现**样本量极小（仅仅2个）**其实才是导致revenue为负数的核心原因。

这是因为：对于卖家数为2、样本量为3的情况来说 （X特征矩阵为3*2，Y元素个数为3），例如

```
{
  "buyer_id": 1,
  "mu": 100.0,
  "X": [[0.1, 0.2], [0.3, 0.3], [0.4, 10]],
  "Y": [1.0, 2.0, 3.0]
},
{
  "buyer_id": 2,
  "mu": 100.0,
  "X": [[0.5, 0.1], [0.6, 0.2], [0.7, 0.3]],
  "Y": [0.5, 1.4, 3.5]
},
```

利用模型线性拟合的过程相当于在解三元（斜率a,b 截距c）一次方程，三个方程（三个样本）！由数学常识知道极大概率是有解的，这导致预期收益 $G=1$ 因此倘若在buyer报价bn的位置得到 $G(Y_n,\hat{Y}_n)$ 收益达不到1（方程无解，少数特殊情况）时候，

$$\mathrm{RF}^*(p_n,b_n,Y_n)=b_n\cdot G(Y_n,\hat{Y}_n)-\int_0^{b_n}G(Y_n,\hat{Y}(z))dz$$

第二项$G(z)$在加上噪声后可能反而导致方程有解 $G(z)=1>G(bn)$,从而算得RF为负数！

当我们增大样本量时（更加符合实际情境），这个问题就解决了。

- 另外，关于实现的质量（与现实情境的符合程度）还有提升空间：revenue相对于买家原报价bn有点小，尤其是对于鲁棒性更强大的训练模型（我们的猜测），后续应进一步考虑适当增大noise（但是不能完全失真导致收益函数 `gain_function` 崩溃）来削减积分项里的$G(Y_n,\hat{Y}_n)$（使减数变小）
- 此外我们认为程序的性能还有提升空间：由于RF算法中涉及积分项，我们采用Trapezoidal rule 进行近似，算法时间复杂度高，所以 MWU定价部分更新权重函数 `update_weights` 应该避免再次重新调用 `revenue_func` ，可以直接将主程序中得到的revenue传入更新函数，即新定义接口，同时也能够避免noise随机性对结果的影响。