数据要素市场

机器学习模型交易完整流程

Author: 高文韬 钱满亮 王芷瑜

Date: 2025-07-26

sp25 Semester

Table of Contents

1 问题描述

随着机器学习技术的快速发展，数据已成为推动 AI 模型性能提升的核心要素。然而，如何在保证公平性和激励相容性的前提下，构建一个有效的数据交易市场仍然是一个重要挑战。本文研究的核心问题是设计一个**双边数据市场**（two-sided data market），其中：

- **数据卖家**：拥有各种特征数据的提供者，希望通过出售数据获得收益
- **数据买家**：需要数据来训练机器学习模型以提升预测精度的用户

该市场面临的主要挑战包括：

1. **定价机制设计**：如何为不同质量和价值的数据制定合理价格
2. **激励相容性**：如何确保买家诚实报告其对数据的真实估值
3. **公平收益分配**：如何根据不同卖家数据的边际贡献公平分配收益
4. **数据质量控制**：如何防止数据复制和低质量数据的恶意投放

本文提出了一个完整的市场框架，通过结合拍卖理论、博弈论和 Shapley 值理论，解决了上述关键问题。

2 文章的整体思路

Summary

文章采用了一个**三步骤的系统性方法**来构建数据市场：

2.1 第一步：诚实机器学习模型拍卖机制

基于 **Myerson 拍卖理论**设计了一个激励相容的拍卖机制：

- **分配函数 AF***：根据买家出价决定数据质量。如果出价低于市场定价，则对数据添加噪声进行“降级”
- **支付函数 RF***：采用 Myerson 支付规则 $r_n = b_n \cdot G(b_n) - \int_0^{b_n} G(z)\,\mathrm{d}z$，确保买家诚实出价是占优策略

该机制的核心思想是通过**数据质量分级**来实现价格歧视，激励买家诚实报告估值。

2.2 第二步：动态定价算法

由于市场环境的不确定性，采用了两种在线学习算法进行动态定价：

- **MWU 算法**（Multiplicative Weights Update）：将价格空间离散化为多个“专家”，通过专家权重更新机制学习最优定价策略
- **UCB 算法**（Upper Confidence Bound）：基于置信区间的探索-利用策略，平衡价格探索与收益最大化

这些算法能够在**零后悔**（zero regret）的理论保证下，逐步收敛到最优价格。

2.3 第三步：基于 Shapley 值的收益分配

为了公平分配卖家收益，文章提出了**鲁棒 Shapley 值**方法：

- **标准 Shapley 值**：衡量每个卖家数据对模型预测精度的边际贡献
- **鲁棒性改进**：通过相似性度量和指数加权，防止数据复制攻击，确保分配的公平性

该方法解决了传统 Shapley 值计算复杂度高和容易被操纵的问题。

3 关键定理的描述

3.1 定理 5.1：真实性保证

定理陈述：对于分配函数 AF，当且仅当假设 **1** 成立时，性质 **3.1**（真实性）可以实现，此时 **RF** 保证真实性。

Proof

证明思路：这是 Myerson 支付函数的经典结果。证明分为两个方向：

1. **正向**：假设买家真实出价 $b_n = mu_n$，通过单调性条件证明偏离策略不会带来更高收益
2. **反向**：如果机制是真实的，则增加的分配不能降低准确性

关键在于买家效用函数的设计，即买家的效用仅来自于获得的估计质量 Y_n，而非特定的数据集分配。

3.2 定理 5.2：收益最大化

定理陈述：设假设 1、3、4 成立，设 $p_{n,n\in[N]}$ 为算法 1 的输出。通过选择算法超参数 $\varepsilon = \left(L\sqrt{N}\right)^{-1}$，$\delta = \sqrt{\frac{\log(|\mathcal{B}_{\text{net}}(\varepsilon)|)}{N}}$，总平均后悔有界为：

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

Proof

证明思路：

1. 首先证明固定价格的后悔界限
2. 利用 MWU 算法的归纳关系式，建立权重更新的递推关系
3. 通过对数不等式和泰勒展开，得到期望收益的渐近界限

该定理说明算法 1 是零后悔算法，收益界限与特征数量 M 无关。

3.3 定理 5.3：收益分配中的公平性

定理陈述：设 $\psi_{n,\text{shapley}}$ 为满足性质 3.3（Shapley 公平性）的唯一向量。对于算法 2，选择超参数 $K > (M\log(2/\delta))/(2\varepsilon^2)$，其中 $\delta, \varepsilon > 0$，则以概率 $1-\delta$，算法 2 的输出 $\hat{\psi}_n$ 满足：

$$\|\psi_{n,\text{shapley}} - \hat{\psi}_n\|_{\text{infinity}} < \varepsilon$$

Proof

证明思路：这是对 Shapley 值的 ε-近似。证明基于：

1. 将 Shapley 值表示为期望形式
2. 利用有界支撑和独立性，应用 Hoeffding 不等式

3. 通过联合界限控制所有坐标的误差

该定理给出了计算 Shapley 值的有限样本保证，使得实际应用中可以通过有限采样获得理论保证的近似解。

3.4 定理 5.4：抗复制鲁棒性

定理陈述：设假设 2 成立。对于算法 3，选择超参数 $K \geq (M\log(2/\delta))/(2(\varepsilon/3)^2)$，$\lambda = \log(2)$，其中 $\delta, \varepsilon > 0$，则以概率 $1-\delta$，算法 3 的输出 ψ_n 是“ε-抗复制鲁棒”的，即满足性质 3.4（抗复制鲁棒性）。

Proof

证明思路:

1. 构造包含复制特征的增广集合 S^+
2. 证明指数惩罚函数 $\exp\left(-\lambda \sum_{j\in[M]\setminus\{m\}} \mathcal{SM}(X_m, X_j)\right)$ 的鲁棒性条件
3. 利用相似性度量的性质，证明复制特征会被指数加权显著降权

该定理确保了即使存在数据复制攻击，收益分配仍然保持公平性。

4 实现的核心算法介绍

4.1 基础复现

4.1.1 main.py

Summary

总体目标：**模拟一次完整的数据市场交易流程**，包括：

1. 市场定价(动态或 UCB 学习)
2. 买家诚实出价
3. 市场根据出价决定是否降噪并训练模型
4. 计算买家的收益(预测精度)
5. 市场向买家收取费用(基于 Myerson 公式)
6. 市场更新价格模型
7. 使用 Shapley 值分配卖家收入(带鲁棒性)

步骤	对应模块	简要说明
0.生成市场数据(特征 X 和标签 Y)	generate_data(from rmse)	创建模拟训练数据，用于收益计算和模型评估
1.动态定价或 UCB 定价	choose_price()(from DynamicPricer/UCBPricer)	通过专家权重选最优价格
2&3.买家出价	按买家诚实出价的假设进行模拟，不涉及具体模块	设定买家的估值和出价
4&5.训练模型并计算预测增益	get_prediction_gain()(from HonestAuction)	使用降噪数据训练模型，计算预测精度
6.计算买家收益和市场收入	calculate_revenue()(from HonestAuction)	基于 Myerson 支付规则计算买家应付费用
7.更新市场定价模型	update_weights()/ update_stats()(from DynamicPricer/UCBPricer)	使用刚获得的收入更新专家权重
8.分配卖家收入(Shapley 值)	shapley_robust()(from RevenueDivider)	使用鲁棒 Shapley 值方法对卖家贡献进行归因和分配收益

caption: main.py 各部分对应模块一览

4.1.2 数据准备与模型准备：**rmse.py**

Summary

负责为市场提供数据生成、模型训练和预测增益计算、收益计算与分配等功能。

4.1.2.1. generate_data(M=10, T=100) 模拟数据生成器

功能说明:

模拟多个数据卖家提供特征数据，目标值由这些特征加权（线性模型）生成并加入噪声。

核心算法:

```
for each seller i in M:
    generate feature vector X[i] of length T
generate true weights w_true of length M
Y = dot(w_true, X) + gaussian_noise
return X, Y
```

4.1.2.2. ml_model = LinearRegression()

功能说明:

使用线性回归模型来学习卖家提供数据与目标值 Y 的关系。

核心算法：

```
initialize ml_model as LinearRegression
```

4.1.2.3. gain_function_rmse(y_true, y_pred)

功能说明：

评估模型预测质量，返回一个增益值（越接近 1 表示模型越好）。

核心算法：

```
rmse = sqrt(mean_squared_error(y_true, y_pred))
y_std = std(y_true)
if y_std == 0:
    return 1.0
normalized_rmse = rmse / y_std
gain = max(0, 1 - normalized_rmse)
return gain
```

4.1.3 诚实的机器学习模型拍卖：HonestAuction.py

Summary

实现了论文中的诚实拍卖机制，包括分配函数 (AF*) 和收益函数 (RF*)。

4.1.3.1. 分配函数(_allocation_function)

功能说明：根据买家的出价 b_n 与市场定价 p_n 的差值，决定是否对数据加入噪声，从而“降级”数据质量。

机制来源：论文 Example 4.1 的 AF* 实现。

核心算法：

```
if bid >= price: return X else: return X + noise(price - bid)
```

4.1.3.2. 预测增益函数(get_prediction_gain)

功能说明：训练模型并计算预测结果与真实结果之间的 预测增益 G。

核心算法：

```
X_tilde = AF*(X, p, b) → model.fit → gain = G(y_true, y_pred)
```

4.1.3.3. 收益函数(calculate_revenue)

功能说明：按 Myerson 支付规则，积分求信息租金，再计算收益

核心算法：Myerson 支付规则

$$r_n = b_n \cdot G(b_n) - \int_0^{b_n} G(z)dz$$

含义：

- 第一项 $b_n \cdot G(b_n)$：买家愿意为当前增益支付的金额；
- 第二项 $\int_0^{b_n} G(z)dz$：信息租金，确保机制诚实激励；

两者相减就是市场应收费用。

数值实现方法：

Tip

这里一开始用的是 scipy.integrate.quad 进行区间积分，但老遇到达到最大次数限制后仍然无法收敛的问题。

后来手动优化成使用 Simpson 积分法，数值稳定性更好,也没有再报错了。

```
z_vals = np.linspace(0, b_n, 100)
y_vals = [G(z) for z in z_vals]
integral = simpson(y_vals, z_vals)
```

4.1.4 DynamicPricer.py

Summary

该程序的目的是实现动态定价策略：

在数据市场中，根据拍卖反馈不断调整对价格的“信心”，从而自动寻找最优价格点。

4.1.4.1 核心算法：**MWU(Multiplicative Weights Update)**：

算法步骤如下：

1. 将价格空间 [min_price, max_price] 离散为 num_experts 个价格(每个价格对应一个“专家”)；
2. 初始时，每个专家权重设为 1；
3. 在每一轮交易中：
   - 根据专家权重的归一化分布，随机抽取一个价格 p_n(相当于让一个专家出主意)；
   - 得到市场反馈(通过 auction_mechanism 得到的收益)；
   - 所有专家根据反馈更新权重(收益越好，权重涨得越多)；

更新公式：

$$w_i^{n+1} \equiv w_i \cdot (1 + \delta \cdot \hat{g}_i)$$

其中，$\hat{g}_i$ 是专家 i 的归一化虚拟收益，δ 是学习率。

4.1.5 RevenueDriver.py

实现了一个 基于 Shapley 值的收益分配系统，用于衡量和分配多个“卖家”或“特征提供者”在一个预测模型中所做出的边际贡献。

4.1.5.1. `shapley_approx` — 近似 Shapley 值

功能说明:

使用 K 次随机排列，近似计算每个卖家对预测任务的 Shapley 边际贡献。

核心算法

```
函数 shapley_approx(X, Y, K):
    初始化每个卖家的 shapley 值为 0
    对于 k in 1 到 K:
        生成一个随机排列 perm
        初始化 gain_predecessors = 0
        对于 perm 中的每个卖家 i:
            gain_current = get_gain_for_subset(前 i 个 + 当前卖家)
            marginal = gain_current - gain_predecessors
            将 marginal 加入当前卖家的 shapley 值
            gain_predecessors = gain_current
    返回 shapley 值的平均
```

4.1.5.2. `shapley_robust` — 鲁棒 Shapley 值

功能说明:

使用 K 次随机排列，近似计算每个卖家对预测任务的 Shapley 边际贡献。

核心算法

```
函数 shapley_robust(X, Y, K, λ):
    approx_shapley = shapley_approx(X, Y, K)
    计算 X 中每个卖家特征的 cosine 相似度矩阵 S
    对于每个卖家 i:
        total_sim = sum(S[i]) - 1
        penalty = exp(-λ * total_sim)
        robust_shapley[i] = approx_shapley[i] * penalty
    返回 robust_shapley
```

4.2 进阶功能

4.2.1 UCBPricer.py

Summary

实现了更多样的定价机制： UCB(Upper Confidence Bound)定价策略。

4.2.1.1.核心算法

UCB 算法是多臂赌博机问题中一种经典的基于置信区间的探索-利用策略。其核心思想是为每个臂的奖励估计构建一个置信区间上界，选择上界最大的臂，从而在探索和利用之间自动平衡。

对于每个候选臂 $k = 1, \cdots, K$，**令** $Q_1(a_k) = 0$, $N_1(a_k) = 0$
for $t = 1, \cdots, T$ **do**
  if $t \leqslant K$ **then**
    初始化顺序选择每个臂
  else
    选择 $a_t = \arg\max_a (Q_t(a) + \sqrt{\frac{2 \ln t}{N_t(a)}})$
  end if
  观察奖励 r_t, $N_{t+1}(a_t) = N_t(a_t) + 1$, $Q_{t+1}(a_t) = Q_t(a_t) + \frac{r_t - Q_t(a_t)}{N_{t+1}(a_t)}$
end for

caption: UCB 定价策略的代码参考了这个课内 slide

`choose_price`: 基于 UCB 算法选择一个当前最优价格

UCB 核心算法:

```
average_reward + sqrt((c * log(total_rounds)) / count)
```

`update_stats`：某轮价格尝试后，用实际收益 reward 更新该价格（专家）的平均收益。

核心算法:

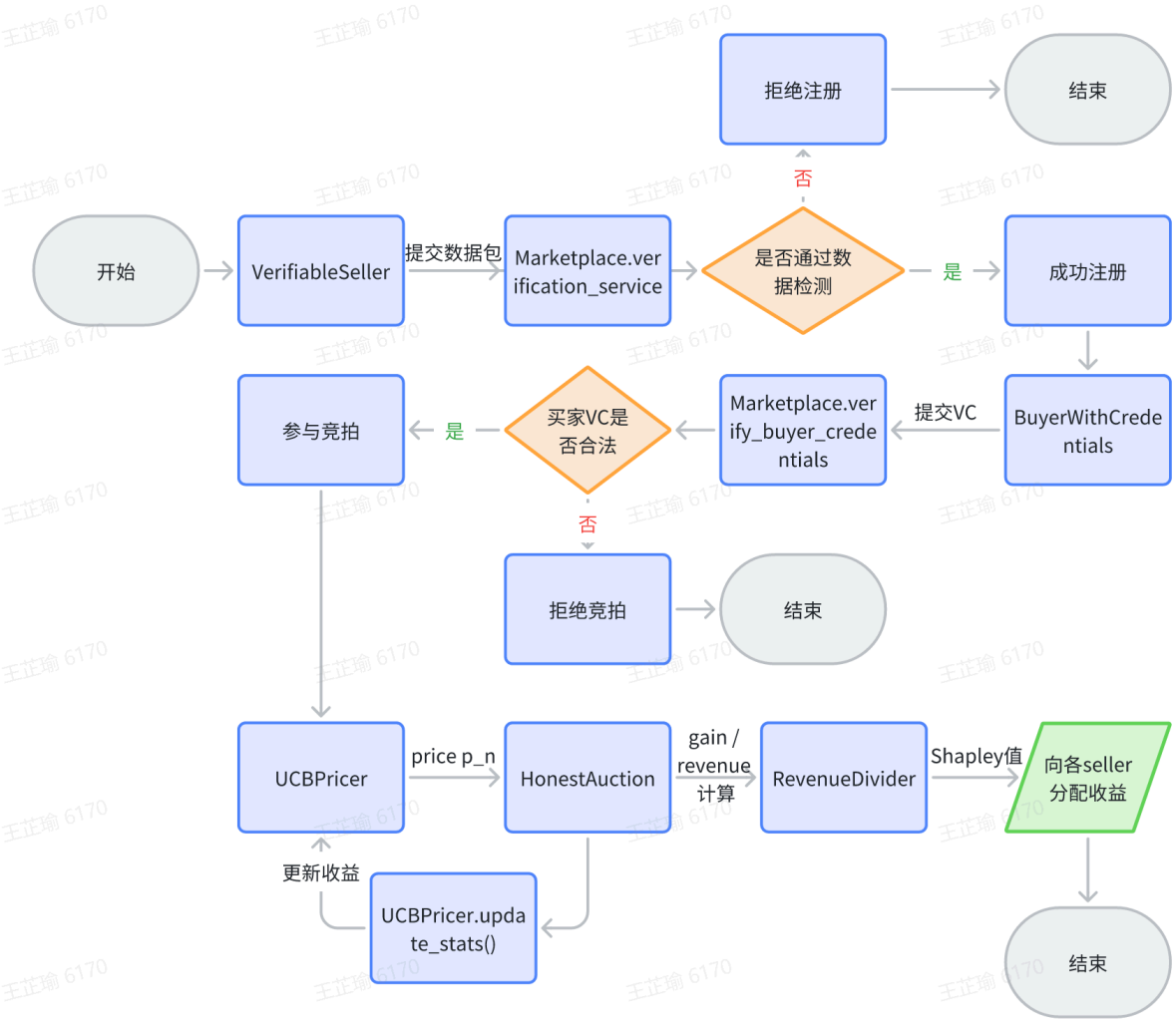
```
counts[i] += 1
values[i] = ((counts[i] - 1)/counts[i]) * old_value + (1/counts[i]) * reward
```

4.2.1.2.对比 **UCB** 定价 与 **Dynamic** 定价(基于 **ML** 回归预测)

对比维度	UCB 定价机制	Dynamic 定价机制(ML 回归)
核心原理	多臂老虎机 + 置信上界	机器学习模型预测最优价格/收益
是否依赖标签	不依赖精确标签, 基于经验更新收益估计	需要监督数据(历史价格 → 收益)
探索能力	强(自动探索未尝试价格)	弱(依赖已有训练数据)
适用场景	• 在线广告投放价格 • 即时拍卖价格选择	• 产品销售预测定价 • 卖家特征驱动的市场机制
响应速度	初期慢，长期收敛	初期快, 易过拟合或失效于新环境

4.2.2 Security.py

4.2.2.1. 总流程



caption: 安全模式+main 综合后 的工作流程, 出于工作量考虑只做了一种定价策略的实现

步骤	对应模块	功能简述
1.初始化市场	`Marketplace` (market_mechanisms.py)	提供市场整体逻辑，包括注册验证服务、VC 验证等
2&3.创建卖家对象	`VerifiableSeller` (market_participants.py) `VerifiableSeller.set_data()`	卖家类，具备：生成数据、提交注册数据包(含签名+ZKP)的能力,设置好数据
4.生成注册包并提交	`VerifiableSeller.get_data_registration_package()`	提供数据包(含数据、签名、ZKP)提交市场
5.数据注册并验证	`Marketplace.verification_service.register_data()`	使用伪造检测+重复检测机制，确保数据来源独立真实

6~8.创建定价器、竞拍器、收入分配器	同 4.1	同 4.1
9.创建买家对象	`BuyerWithCredentials` (market_participants.py)	买家类，支持 VC 认证管理和出价行为
10.添加买家 VC	`BuyerWithCredentials.add_credential()`	添加买家凭证(如医疗研究员资格)
11.验证买家资格	`Marketplace.verify_buyer_credentials()`	检查某买家是否持有合法 VC
12~23.正常进行后续拍卖流程	同 4.1	同 4.1

4.2.2.2.ZKP 验证卖家数据所有权

主要功能：`MarketVerificationService` 验证并注册卖家的数据包，防止数据抄袭，使用零知识证明（ZKP）确认所有权

核心算法： ZKP 验证 `MarketVerificationService.register_data()`

```
If data_hash in registered_data:
  Reject
If not verify_zkp(public_key, data_hash, zkp):
    Reject
Add data_hash to registered_data
Accept
```

流程:

- hashlib.sha256(…) 计算数据的不可逆摘要，作为数据唯一表示
- sign(…) 使用私钥对哈希签名（如使用 ECDSA），表明“我拥有此数据”
- generate_zkp_of_signature(…) 构造一个零知识证明，证明签名是由拥有该公钥对应私钥者生成的，而不暴露私钥

4.2.2.3. VC 验证买家身份

主要功能:

- `BuyerWithCredentials.wallet` 买家存储从平台获取的访问凭证 VC
- `BuyerWithCredentials.present_credential()`向市场出示某一类凭证，供验证逻辑使用

核心算法:

- `wallet`：一个字典，存储类型化的凭证（如 “Access”, “Reputation”, “Score”）
- `add_credential(...)`：添加凭证，例如来自市场管理员

- `present_credential(...)`：买家在交易时出示凭证

Hint

这段和论文的关系就比较小了，但是鉴于信安专业+课上还是提过一些隐私计算所以想到这个方向后还是毅然决然地做了。

在 `cryptography_lib` 的部分遇到了比较大的麻烦（也留下了比较大的坑）

最后成功调用出简单的 ZKP 验证功能。在这里记录一下两次尝试：

- 尝试 1：Zokrates 编译 zok 文件（更接近生产环境）

  编译的时候由于环境原因一直报错：

```
Compilation failed:

root.zok: --> 1:39
  |
1 | def main(private field a, field b) -> field {
  |                                       ^---
  |
  = expected ty_array
```

  尝试解决无果，重新配环境工作量过大，放弃。

- 尝试 2：使用 zksk 库实现 ZKP 验证（建议的 ZKP 验证，实现成功）

  此处遇到了大量环境配置问题，具体解决方案已写在 Code/README.md 中。完成环境配置后更新代码

  具体实现方式以调库为主，详见代码：

```python
def generate_zkp_of_signature(data_hash, signature, public_key):
    """
    使用 zksk 构造“知道签名者私钥”的零知识证明。
    模拟离散对数知识证明：证明知道私钥 x，使得 pk = g^x
    """
    group = EcGroup()
    order = group.order()
    g = group.generator()
    x_value = order.random()
    h = g.pt_mul(x_value)
    x = Secret()
    stmt = DLRep(h, x * g)
    proof = stmt.prove({x: x_value})

    return {
        "zkp_proof": proof,
```

```
            "g": g.export(),
            "h": h.export()
        }

    def verify_zkp(public_key, data_hash, zkp_dict):
        """
        验证零知识证明
        """
        group = EcGroup()
        g = group.generator()
        h = EcPt.from_binary(zkp_dict["h"], group)
        x = Secret()
        stmt = DLRep(h, x * g)
        proof = zkp_dict["zkp_proof"]
        return stmt.verify(proof)
```

5 实验结果与分析

5.1 基础复现

5.1.1 实验结果

```
Practice2/DataMarket/Code$ python main.py
--- 开始模拟单次交易 ---
使用的定价器: DynamicPricer
步骤 1: 市场设定价格 p_n = 357.89
步骤 2 & 3: 买家到达，真实估值 μ_n = 200.00，诚实出价 b_n =
200.00
步骤 4 & 5: 市场分配数据并计算预测增益...
  - 基准增益 (无噪声): 0.9398
  - 实际增益 (根据出价): 0.0482
步骤 6: 市场向买家收取收益 r_n = 1.94
步骤 7: 市场价格模型权重已更新。
步骤 8: 市场计算并分配收益给卖家...
  - 计算出的Shapley值 (归一化前): [0.003  0.0003 0.0003 0.00
26 0.0004 0.     0.0012 0.0007 0.0001 0.0007]
  - 各卖家的收益分配: [0.63 0.05 0.05 0.55 0.08 0.01 0.25 0.
14 0.02 0.15]
--- 模拟结束 ---
```

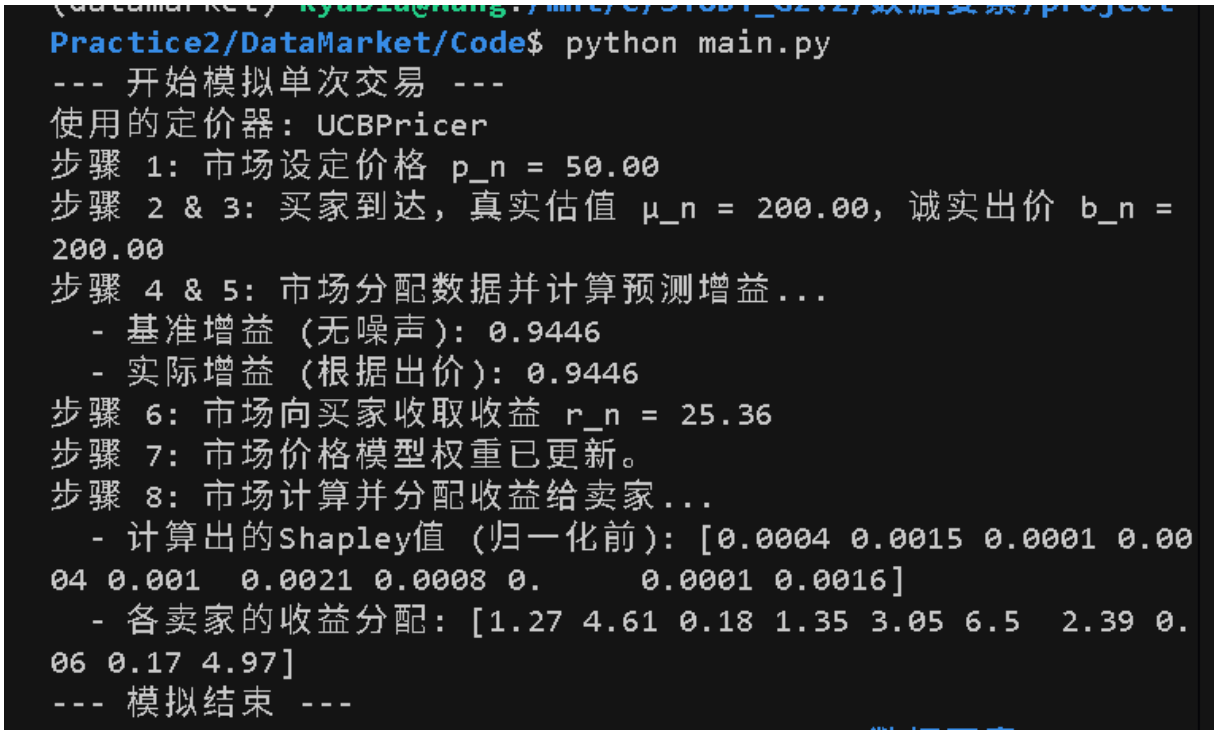
5.1.2 实验分析

按照图示步骤模拟了机器学习市场模型交易完整流程。

其中，诚实的机器学习模型拍卖（Honest Auction），MWU 算法动态定价（Multiplicative Weights Update）和 收益分配（Shapley 值）均在流程中有实现，已在上一节中分别给出核心算法；对于用户数据的模拟则有所简化。

5.2 更多的定价算法

5.2.1 实验结果



5.2.2 实验分析

按照图示步骤模拟了机器学习市场模型交易完整流程。

将定价算法从 DynamicPricer 改为 UCBPricer，使用 UCB 定价策略替代了 MWU 定价策略。在数据生成和定价环节均有变化，可以看到程序成功跑通。

5.3 隐私计算

5.3.1 实验结果

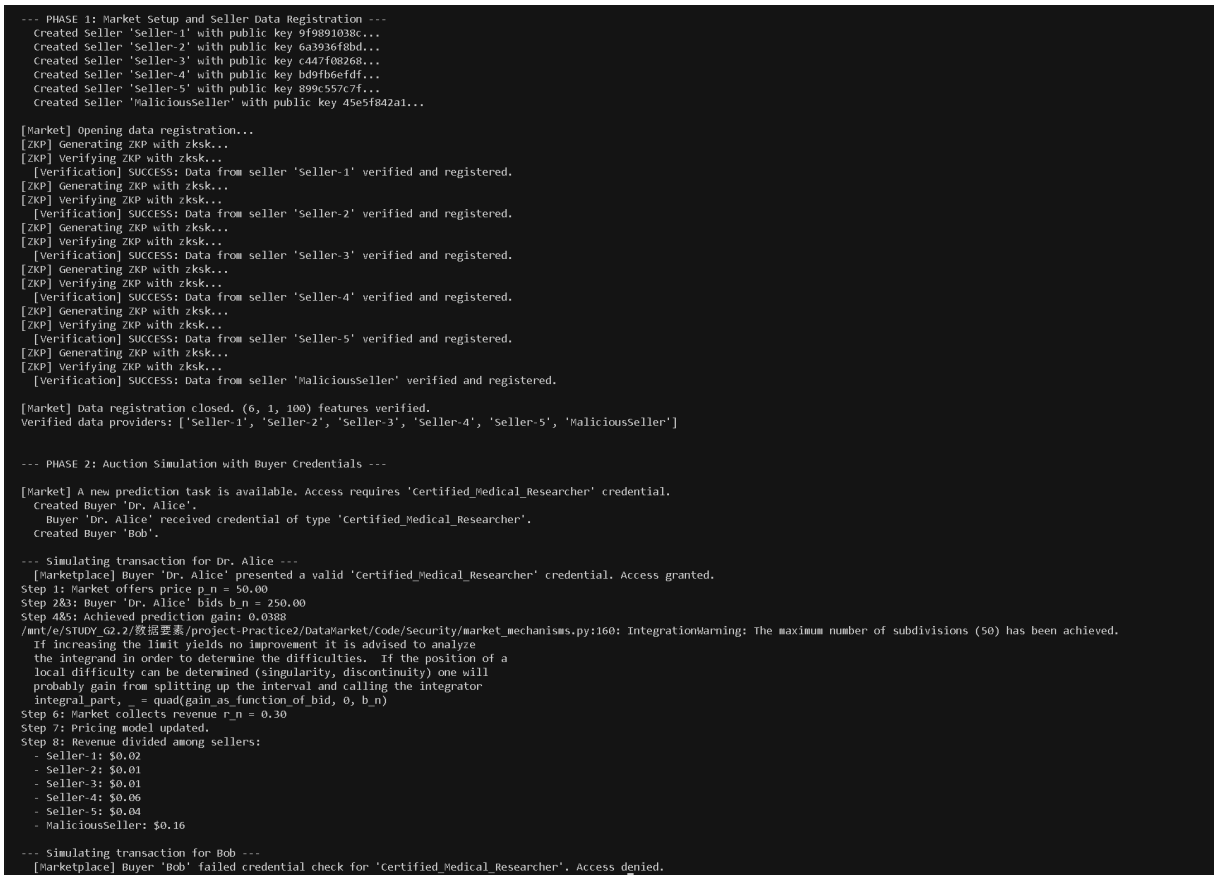
未引入 ZKP 验证的版本

```
Practice2/DataMarket/Code/Security$ python smain.py
--- PHASE 1: Market Setup and Seller Data Registration ---
  Created Seller 'Seller-1' with public key 12919f7c0d...
  Created Seller 'Seller-2' with public key 5cba4afcaa...
  Created Seller 'Seller-3' with public key d88f807070...
  Created Seller 'Seller-4' with public key 0b4a610c85...
  Created Seller 'Seller-5' with public key 9276425af8...
  Created Seller 'MaliciousSeller' with public key 615eb2e552...

[Market] Opening data registration...
    [Crypto] Generating mock ZKP for data hash c57f215e9d...
    [Crypto] Verifying mock ZKP for public key 12919f7c0d...
  [Verification] SUCCESS: Data from seller 'Seller-1' verified and registered.
    [Crypto] Generating mock ZKP for data hash eb4006ce2c...
    [Crypto] Verifying mock ZKP for public key 5cba4afcaa...
  [Verification] SUCCESS: Data from seller 'Seller-2' verified and registered.
    [Crypto] Generating mock ZKP for data hash 6c03c84d4f...
    [Crypto] Verifying mock ZKP for public key d88f807070...
  [Verification] SUCCESS: Data from seller 'Seller-3' verified and registered.
    [Crypto] Generating mock ZKP for data hash 804a040128...
    [Crypto] Verifying mock ZKP for public key 0b4a610c85...
  [Verification] SUCCESS: Data from seller 'Seller-4' verified and registered.
    [Crypto] Generating mock ZKP for data hash 023e864880...
    [Crypto] Verifying mock ZKP for public key 9276425af8...
  [Verification] SUCCESS: Data from seller 'Seller-5' verified and registered.
    [Crypto] Generating mock ZKP for data hash c28d6ee8a4...
    [Crypto] Verifying mock ZKP for public key 615eb2e552...
  [Verification] SUCCESS: Data from seller 'MaliciousSeller' verified and registered.

[Market] Data registration closed. (6, 1, 100) features verified.
Verified data providers: ['Seller-1', 'Seller-2', 'Seller-3', 'Seller-4', 'Seller-5', 'MaliciousSeller']


--- PHASE 2: Auction Simulation with Buyer Credentials ---

[Market] A new prediction task is available. Access requires 'Certified_Medical_Researcher' credential.
  Created Buyer 'Dr. Alice'.
    Buyer 'Dr. Alice' received credential of type 'Certified_Medical_Researcher'.
  Created Buyer 'Bob'.

--- Simulating transaction for Dr. Alice ---
  [Marketplace] Buyer 'Dr. Alice' presented a valid 'Certified_Medical_Researcher' credential. Access granted.
Step 1: Market offers price p_n = 50.00
Step 2&3: Buyer 'Dr. Alice' bids b_n = 250.00
Step 4&5: Achieved prediction gain: 0.0385
/mnt/e/STUDY_G2.2/数据要素/project-Practice2/DataMarket/Code/Security/market_mechanisms.py:119: IntegrationWarning: The maximum number of subdivisions (50) has been a
uity) one will
```

```
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  integral_part, _ = quad(gain_as_function_of_bid, 0, b_n)
Step 6: Market collects revenue r_n = 0.18
Step 7: Pricing model updated.
Step 8: Revenue divided among sellers:
  - Seller-1: $0.01
  - Seller-2: $0.00
  - Seller-3: $0.04
  - Seller-4: $0.01
  - Seller-5: $0.02
  - MaliciousSeller: $0.11

--- Simulating transaction for Bob ---
  [Marketplace] Buyer 'Bob' failed credential check for 'Certified_Medical_Researcher'. Access denied.
```

Attention

此处的图片是引入 ZKP 验证的结果，时间问题来不及同步到前端了，仅在此处实现。



5.3.2 实验分析

按照图示步骤模拟了带上隐私计算功能的机器学习市场模型交易完整流程。

其中，生成公钥和 ZKP 的过程均在输出中有体现，对于 VC 过程，Alice 的数据模拟了 VC 通过并正常进行拍卖定价和收益分配的全流程，而 Bob 的数据则模拟了 VC 验证失败的情况，可以看到拍卖正常中止。

Warning

当然可以注意到，程序运行过程中有一个未解决的 integral_part 问题。

涉及的代码是基础流程代码，并非隐私计算相关代码。 此处的问题和基础版本 4.1.3.3.提到的报错相似，解决方法也相似，同样出于工作量原因考虑没有进行修复。

5.3 前端展示

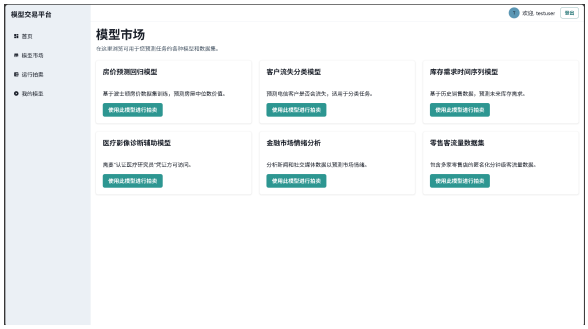
类别	技术/库	作用说明
核心框架与构建	React	构建用户界面，采用组件化开发
	Vite	提供快速开发和构建（含热更新）
UI 组件与样式	Chakra UI	构建统一风格的 UI 组件
	Emotion	提供 CSS-in-JS 样式支持
	Framer Motion	实现组件动画和过渡效果

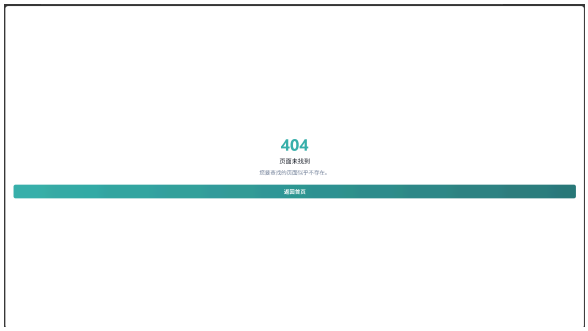
	React Icons	集成常用图标库
路由管理	React Router	控制页面跳转和访问权限
状态管理	Context API	管理全局状态（如用户认证）
	React Hooks	管理组件状态和副作用逻辑
API 通信	Axios	向后端发送 HTTP 请求
用户认证	JWT	实现登录认证和身份校验
	Cookie (HttpOnly)	存储 Token，提高安全性
	jwt-decode	解码 JWT，提取用户信息
开发环境	Node.js & npm	支持 Vite 和后端运行
模拟后端	Express.js	提供 mock API 接口
	CORS 中间件	允许前后端跨域通信

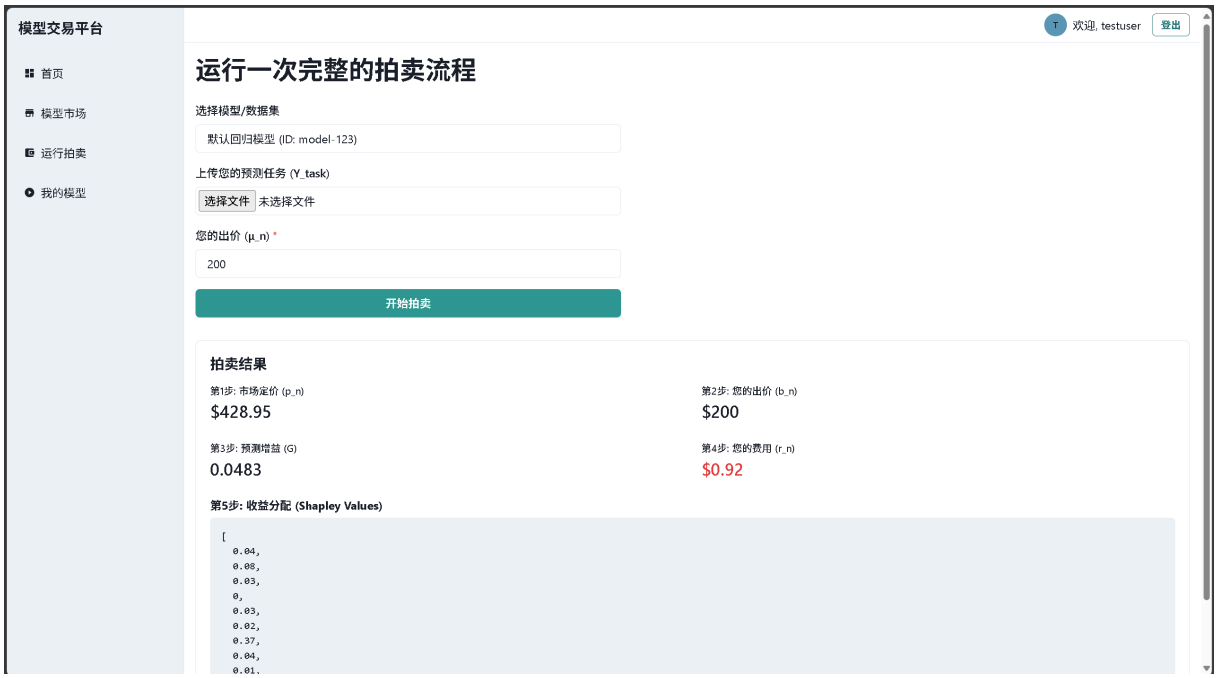
此处针对基础流程和密码学计算分别做了两个 API 接口连接到后端。

工作量问题导致没有完成的内容：切换算法的接口没有做、数据库没有连接。

主要功能页面：首页、登录（包含 JWT 和 cookies 鉴权，路由检测到未登录会自动跳转）、模拟拍卖页面（核心功能，接入本地模拟程序 api）、模型市场界面（除了 router 没做很多内容，只是生产环境下的前端应该会有这样一个页面）、default 页面（404 页面）







caption: 前端展示的核心页面

密码学功能展示：上传模型并进行 ZKP 验证，接入 ZKP 验证的后端 API，前端展示了生成的公钥和 ZKP。