SweatyMarkets V1
Data : ManCity 2023-2024
Guillaume Leopold

I) Turning sports odds into a X% change in a market mechanism

Odds retrieved from a sports API are converted into win probabilities, which – combined with sports based volatility (team unpredictability) dynamically drive the price of a team token.

Step 1 – Data Acquisition and Preprocessing

For each match $m \in \mathcal{M}$, retrieve:

- Teams: i, j
- Final scores: $S_i^{(m)}, S_j^{(m)}$
- Bookmaker odds: $O_i^{(m)}, O_j^{(m)}, O_D^{(m)}$

From these, compute the **implied probabilities** $P_i^{(m)} \in [0, 1]$:

$$P_i^{(m)} = \frac{1/O_i^{(m)}}{1/O_i^{(m)} + 1/O_j^{(m)} + 1/O_D^{(m)}} \quad \text{(and similarly for } P_j^{(m)}, P_D^{(m)})$$

These probabilities reflect the market's expectations before the match and will be contrasted with actual outcomes.

Step 2 – Observed Outcomes and Surprise Term

Define the **outcome score** $A_i^{(m)} \in \{0, 0.5, 1\}$ for team i as:

$$A_i^{(m)} = \begin{cases} 1 & \text{if } S_i^{(m)} > S_j^{(m)} \\ 0.5 & \text{if } S_i^{(m)} = S_j^{(m)} \\ 0 & \text{if } S_i^{(m)} < S_j^{(m)} \end{cases}$$

The **surprise** or prediction error is then defined as:

$$\delta_i^{(m)} = A_i^{(m)} - P_i^{(m)}$$

Additionally, compute the **goal differential**:

$$D_i^{(m)} = S_i^{(m)} - S_j^{(m)} \in \mathbb{Z}$$

Step 3 – Volatility and Liquidity Estimation

We introduce **conditional volatility** using a GARCH(1,1) process:

Let $r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$ be the log return of the token price series for a team.

The **time-varying volatility** is estimated as:

$$\sigma_t^2 = \omega + \alpha \cdot r_{t-1}^2 + \beta \cdot \sigma_{t-1}^2 \quad \text{with } \omega, \alpha, \beta > 0$$

This allows for volatility clustering and autoregressive dynamics.

Liquidity $\lambda_i^{(m)}$ is modeled as the inverse of volatility:

$$\lambda_i^{(m)} = \frac{1}{\sigma_t}$$

This captures the intuitive idea that highly volatile tokens are less liquid and more sensitive to shocks.

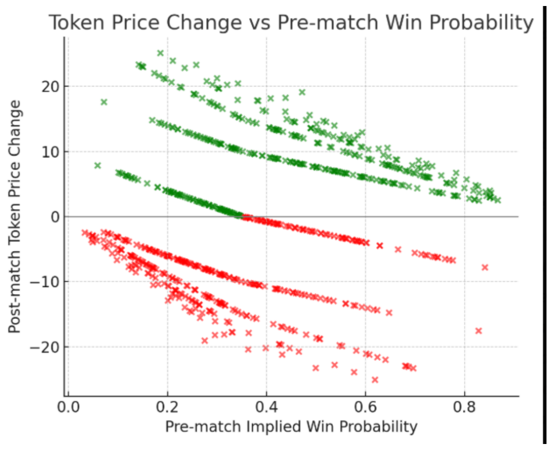
Step 4 – Raw Rating Update Function

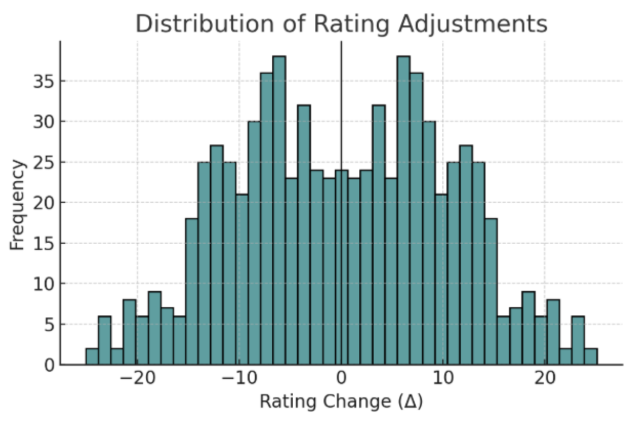
The **raw rating variation** $\Delta R_{\text{raw},i}^{(m)}$ incorporates:

- Surprise $\delta_i^{(m)}$
- Goal differential $D_i^{(m)}$
- Market liquidity and volatility

$$\Delta R_{\text{raw},i}^{(m)} = \lambda_c \cdot (1 - P_i^{(m)}) \cdot \left(1 + \lambda_i^{(m)} \cdot \sigma_t\right) \cdot \log(1 + \max(D_i^{(m)}, 0))$$

Where $\lambda_c > 0$ is a global calibration constant. Similar expressions (with asymmetric weights) are used for draws and losses.





II) Regulations : Zero-Sum Condition & Dynamic Loss Limit

Step 5 – Zero-Sum Normalization

To ensure that gains and losses across the market cancel out at each matchday:

$$\Delta R_i^{(m)} = \Delta R_{\text{raw},i}^{(m)} - \frac{1}{N}\sum_{j=1}^{N} \Delta R_{\text{raw},j}^{(m)} \quad \text{so that} \quad \sum_{i=1}^{N} \Delta R_i^{(m)} = 0$$

This guarantees market conservation of value across all tokens.

Step 6 – Dynamic Loss Limitation (CVaR-style)

To prevent unrealistically large losses, especially in illiquid conditions, we define a **dynamic floor** for each token:

$$L_{\min}^{(m)} = \max\left(-0.99, -\left(\theta + \phi \cdot \frac{\sigma_{\text{avg}}^{(m)}}{\lambda_c}\right)\right)$$

Where $\sigma_{\text{avg}}^{(m)}$ is the average volatility across tokens at matchday m, and $\theta, \phi > 0$ are fixed constants.

The final capped variation is:

$$\Delta R_i^{(m)} := \max\left(\Delta R_i^{(m)}, L_{\min}^{(m)}\right)$$

This ensures bounded negative updates and avoids token collapse.

A zero-sum condition ensures that gains and losses among participants offset to maintain internal balance. The dynamic loss limit – scaled to volatility – prevents extreme downside moves and stabilize token behavior during extreme events