CasinoHoldemAI Documentation

Marcin Kondrat (`@breftejk`)

July 9, 2025

1 Introduction

CasinoHoldemAI is a production-grade decision engine for Casino Hold'em, integrating rigorous Monte Carlo simulation with a gradient-boosted decision tree classifier (XGBoost). Key contributions include:

- A configurable Monte Carlo equity estimator with iteration count N.
- Rich feature extraction combining combinatorial and probabilistic metrics.
- A robust XGBoost training pipeline for optimistic CALL/FOLD classification.

2 Installation

1. Create a virtual environment:

```
python3 -m venv .venv
source .venv/bin/activate
```

2. Upgrade core tools:

```
pip install --upgrade pip setuptools wheel Cython
```

3. Install dependencies and package:

```
pip install numpy pandas eval7 scikit-learn xgboost joblib tqdm
pip install .
```

4. Verify:

```
casino-ai --help
```

3 Theoretical Foundations

3.1 Monte Carlo Simulation

Let W_i be the indicator of a win on trial i. Then:

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} W_i, \quad \mathrm{Var}(\hat{p}) = \frac{\hat{p}(1-\hat{p})}{N}.$$

By the Central Limit Theorem, for large N:

$$\hat{p} \sim \mathcal{N}\Big(p,\, \frac{p(1-p)}{N}\Big).$$

A $100(1-\alpha)\%$ confidence interval:

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}.$$

3.2 Feature Engineering

Define rank frequency vector $\mathbf{c} = [c_2, \ldots, c_{\mathrm{A}}]$. Pattern indicators:

$$\text{Pair} = \mathbb{I}\big(\max(\mathbf{c}) \geq 2\big), \quad \text{Trips} = \mathbb{I}\big(\max(\mathbf{c}) \geq 3\big).$$

Expected outs calculation:

$$E[\text{outs}] = \sum_k o_k \frac{\binom{o_k}{1}\binom{R-o_k}{T-1}}{\binom{R}{T}},$$

normalized to $[0, 1]$.

4 System Architecture

CLI: `argparse`-based command parsing.

Simulator: $\mathcal{O}(N)$ Monte Carlo per hand, parallel via `joblib.Parallel`.

FeatureExtractor: Computation of combinatorial/statistical features.

DataGenerator: Batch orchestration with progress via `tqdm`.

ModelTrainer: XGBoost training with train/validation split.

PokerAI: Real-time simulation + ML inference.

5 Machine Learning Details

5.1 XGBoost Ensemble

Model ensemble:

$$F(x) = \sum_{m=1}^{M} f_m(x), \quad \hat{y} = \sigma(F(x)) = \frac{1}{1+e^{-F(x)}}.$$

Objective:

$$\mathcal{L} = \sum_i \ell(y_i, \hat{y}_i) + \sum_m \Omega(f_m), \quad \Omega(f) = \gamma T + \tfrac{1}{2}\lambda\|w\|^2.$$

5.2 Hyperparameter Tuning

Grid search over:

$$\{\text{max\_depth}, \eta, \text{subsample}, \text{colsample\_bytree}, \lambda, \gamma\}$$

using 5-fold CV to minimize logloss.

5.3 Bias-Variance Tradeoff

Generalization error:

$$\text{Err} = \text{Bias}^2 + \text{Variance} + \text{Noise}.$$

Regularization and learning rate control complexity.

6 Data & Feature Engineering

6.1 Monte Carlo Outputs

- `win_rate`, `tie_rate`
- 95% CI: $\hat{p} \pm 1.96\sqrt{\hat{p}(1-\hat{p})/N}$

6.2 Hand Patterns

Indicator vector length 9: pair, two_pair, trips, straight, flush, full_house, quads, straight_flush, royal_flush.

6.3 Card Encoding

Ranks $\{2, \ldots, 9, T, J, Q, K, A\} \to \{2, \ldots, 14\}$; suits mapped to $\{1, \ldots, 4\}$.

7 API Reference

7.1 Generate Data (`gen`)

```
casino-ai gen --n N --out PATH [--iters I] [--workers W]
```

7.2 Train Model (`train`)

```
casino-ai train --in PATH --model PATH
```

7.3 Predict Decision (`pred`)

```
casino-ai pred --model PATH --cards C1,C2 --board B1,B2,B3 [--threshold T]
```

8 Disclaimer

This software is provided *as-is*, without any express or implied warranty. The author assumes no liability for decisions made using this tool.