Predicting Mispriced Polymarket Contracts Using Machine Learning

Logan Morof
CMSE 492 Final Project

November 30, 2025

Abstract

Prediction markets aggregate collective beliefs about the likelihood of future events. While theoretically efficient, real-world prediction markets frequently exhibit temporary mispricings due to liquidity constraints, behavioral biases, market segmentation, and asynchronous information flow. This project develops machine learning models to detect mispriced Polymarket snapshots using engineered time-series and liquidity features. Using logistic regression, random forest, and tuned XGBoost models, I evaluate predictive performance, interpret model behavior with SHAP, and assess the practical implications of threshold selection and backtesting. The optimized XGBoost model achieves strong performance with ROC-AUC of 0.94 and F1 around 0.62 despite severe class imbalance. Additional diagnostics including calibration curves, cost-sensitive threshold analysis, and optimistic backtesting provide insight into both the strengths and limitations of the approach.

1 Background and Motivation

Prediction markets such as Polymarket allow traders to buy and sell contracts whose prices represent implied probabilities of real-world outcomes. Ideally, market prices incorporate all available information and track the eventual resolution closely. However, due to fragmented liquidity, uneven information flow, delays in market response, and heterogeneous trader behavior, prediction markets often diverge meaningfully from their true final outcomes. This project seeks to determine whether machine learning can detect these deviations in advance.

A snapshot is classified as **mispriced** if the absolute difference between its last traded price and its final resolved price is at least 0.15. Although this threshold marks only about 3.7% of observations as mispriced, these events correspond to significant inefficiencies and potential trading opportunities. Detecting these events automatically would help traders monitor thousands of markets simultaneously and highlight markets where price signals diverge from fundamentals.

Machine learning is well suited to this task because it can synthesize noisy signals across price momentum, volatility, liquidity, order flow, and timing features. By training models on historical markets, we can identify non-linear patterns associated with future mispricing and evaluate how well different modeling strategies align with trader objectives.

2 Machine Learning Task and Objective

This project formulates mispricing detection as a binary classification task. For each snapshot, we compute:

$$y_{\text{misprice}} = \begin{cases} 1 & \text{if } |\text{final\_price} - \text{last\_price}| \geq 0.15 \\ 0 & \text{otherwise.} \end{cases}$$

Given feature vectors derived from price history, volatility, liquidity, and market timing, the goal is to estimate the probability that a snapshot will ultimately be mispriced. The minority class is rare and economically important, so metrics such as F1 score, precision, recall, and ROC-AUC are emphasized over accuracy. Supervised learning enables the integration of multiple weak financial signals to form a coherent mispricing probability.

The ultimate objective is to: (1) rank snapshots by mispricing risk, (2) interpret which features drive mispricing, and (3) evaluate the impact of threshold choices in a simulated trading context.

3 Data Description

The processed dataset contains 7,558 snapshots, each corresponding to a unique market-time pair. All features are numerical and represent:

- **Price and momentum:** rolling means, volatility, price range, deviation from trend.
- **Order flow:** buy/sell volume, net order imbalance, trade counts over multiple windows.
- **Liquidity:** cumulative depth and activity.
- **Timing:** market age, time to resolution, time since last trade.
- **Snapshot offsets:** hours before resolution (12, 6, 3, 1).

There are no missing values. The dataset is highly imbalanced: only about 3.7% of snapshots are mispriced. Figures 1 and 2 illustrate the imbalance.

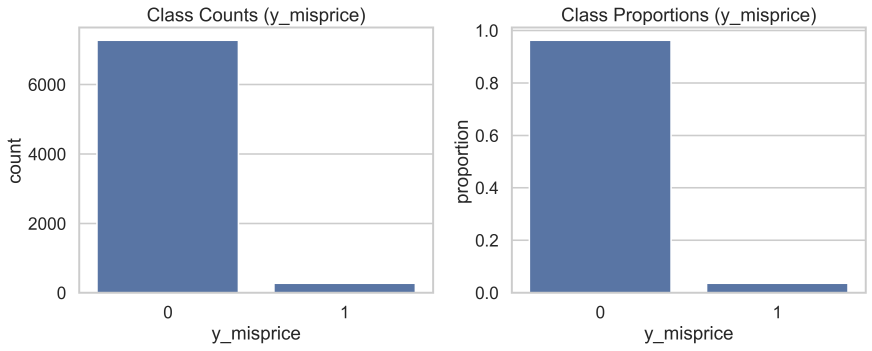

Figure 1: Class distribution (counts) of mispriced vs non-mispriced snapshots.

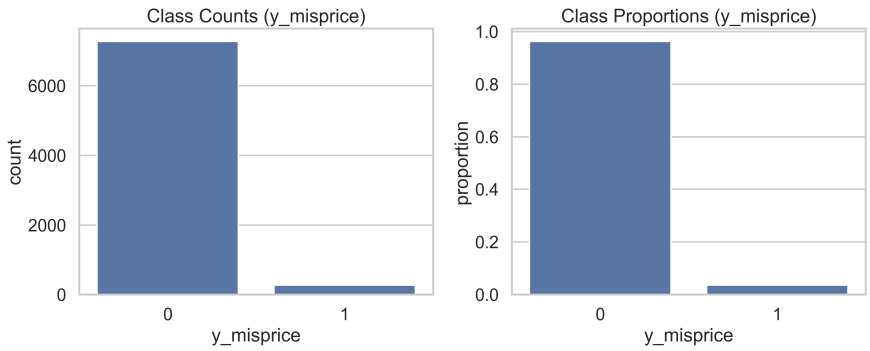

Figure 2: Class distribution (proportions). Only about 3.7% of snapshots are mispriced.

4 Preprocessing

The mispricing label was recomputed from the final and last prices to ensure consistency. The logistic regression, random forest, and XGBoost models were trained using all numerical features except the label and final price (to prevent leakage). Tree-based models do not require scaling; logistic regression uses internal regularization to mitigate large feature magnitude differences.

Stratified 80/20 splitting was used to preserve the minority class in the test set. No SMOTE or oversampling was applied to avoid altering the natural structure of financial features.

5 Models

Three models were compared:

- **Logistic Regression (baseline)** – linear decision boundary; class-weighted.
- **Random Forest** – non-linear, robust to noise, moderate interpretability.
- **XGBoost (optimized)** – gradient boosted trees with hyperparameter tuning; highest performance.

Hyperparameters for the optimized XGBoost model include:

- learning rate, max depth, subsample ratio
- number of boosting rounds
- regularization parameters (lambda, alpha)

RandomizedSearchCV was used to tune both the forest and XGBoost models.

6 Training Methodology

Training followed these steps:

1. Split the data using stratified sampling.
2. Train baseline models with default or lightly tuned hyperparameters.

3. Use RandomizedSearchCV to explore deeper hyperparameter spaces for RF and XGBoost.
4. Evaluate models with ROC-AUC, precision, recall, and F1.
5. Run threshold sweep to find optimal operating points.

The optimized XGBoost model produced the strongest results and was selected for interpretability and backtesting.

7 Metrics

Because of class imbalance, accuracy is not informative. Instead, we report ROC-AUC, F1 score, precision, and recall. The optimized XGBoost model achieved:

- ROC-AUC: 0.94
- Precision (tuned): 0.61–0.64
- Recall: 0.62
- F1: 0.62

Figures 3 and 4 show the ROC and PR curves.

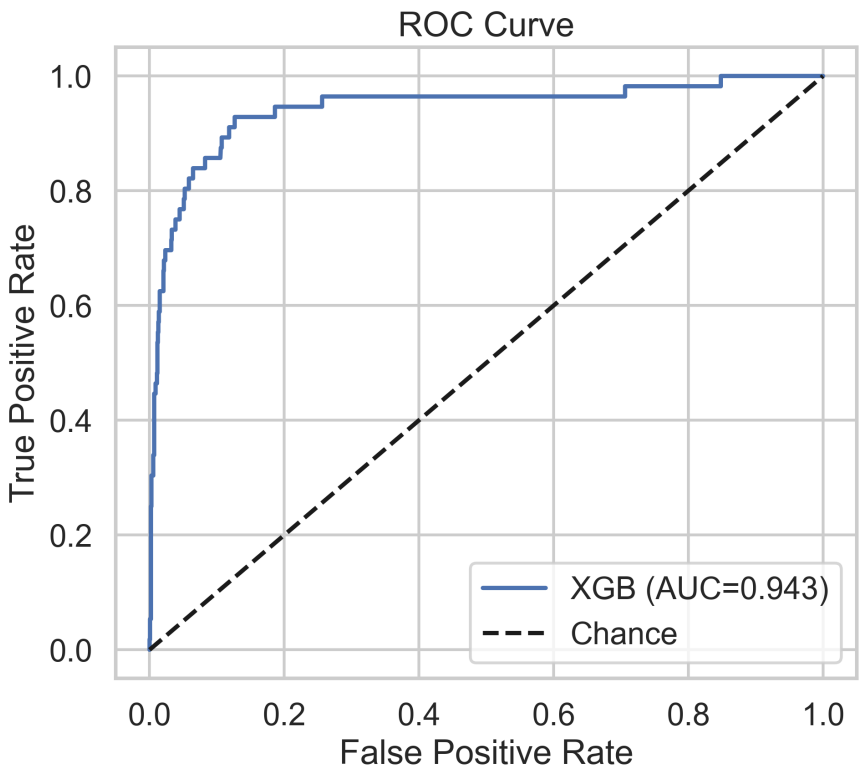

Figure 3: ROC curve for optimized XGBoost model.

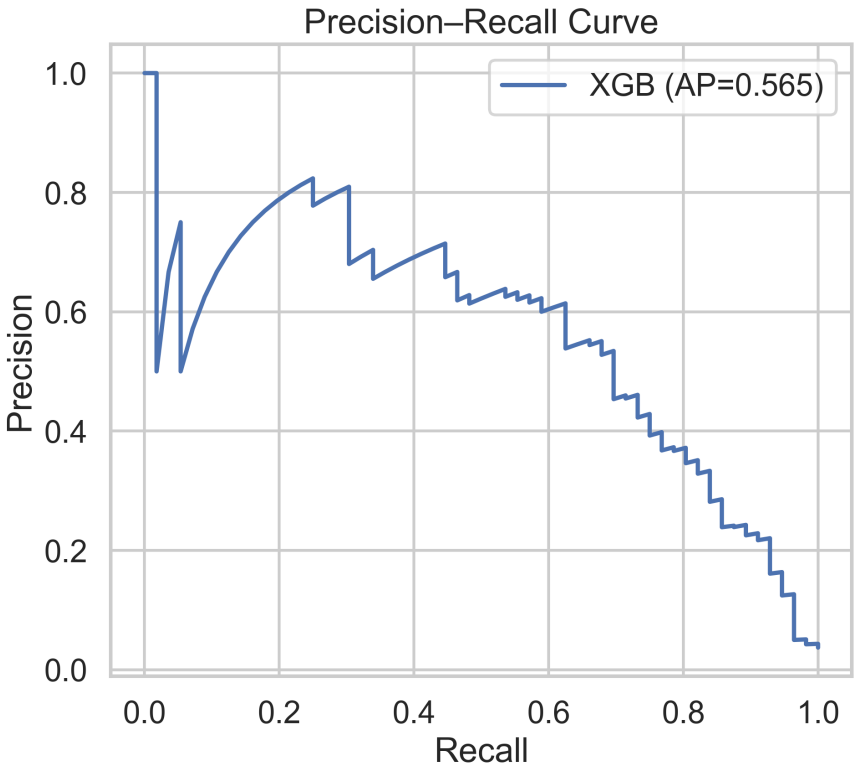

Figure 4: Precision–Recall curve.

8 Results and Model Comparison

The XGBoost model outperformed the alternatives across all metrics. The threshold sweep (Figure 5) shows that the best F1 occurs around 0.50, while higher precision can be achieved at thresholds above 0.70.

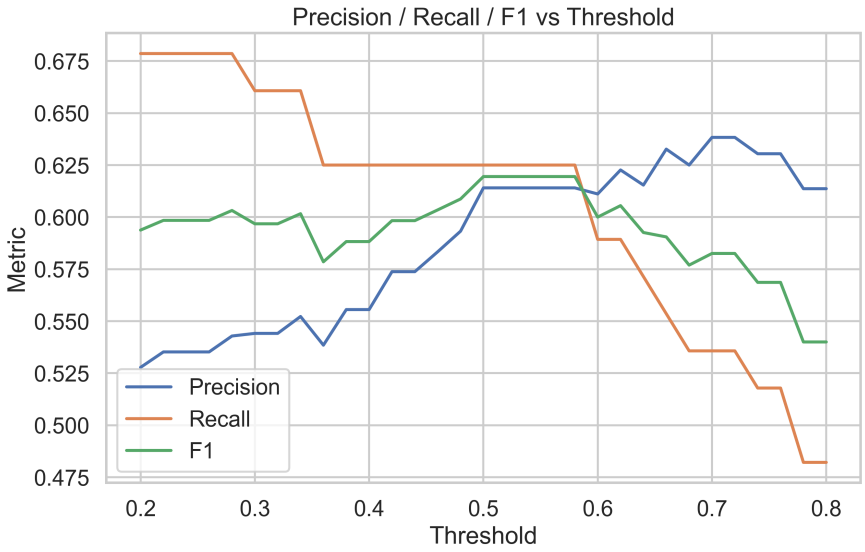

Figure 5: Precision, recall, and F1 as functions of the prediction threshold.

Confusion matrices at these thresholds illustrate trade-offs between false positives and false negatives.

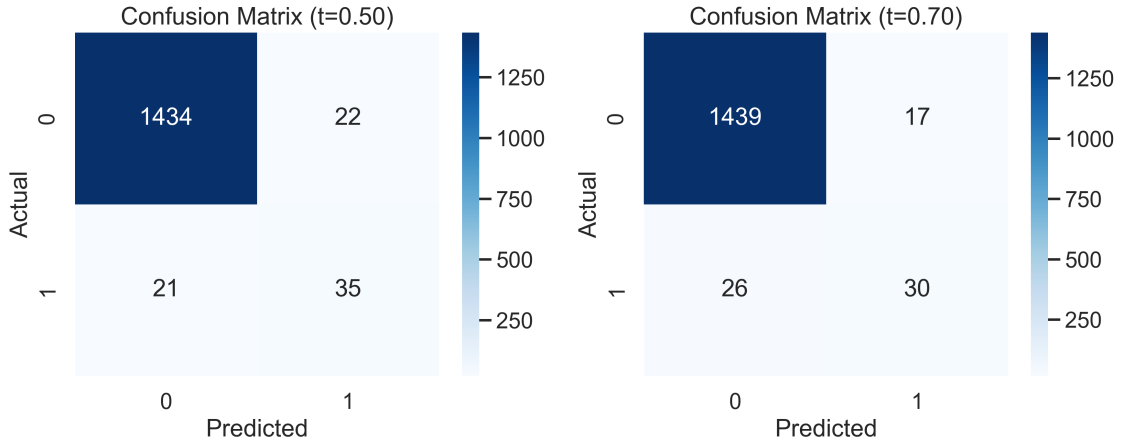

Figure 6: Confusion matrix at threshold 0.50.

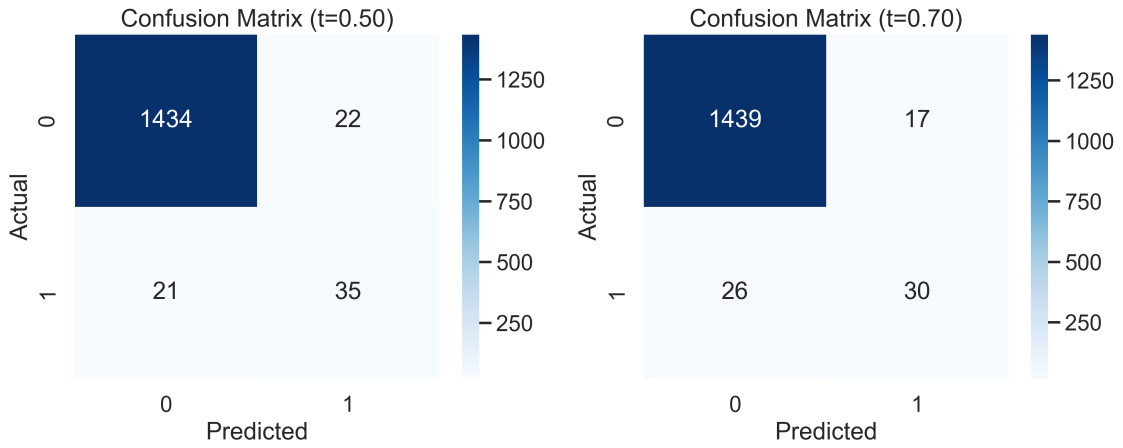

Figure 7: Confusion matrix at threshold 0.70.

9 Model Interpretation

Feature importance results (Figure 8) show that time-to-resolution, recent volatility, order flow ratios, and short-term trade counts are the strongest signals.

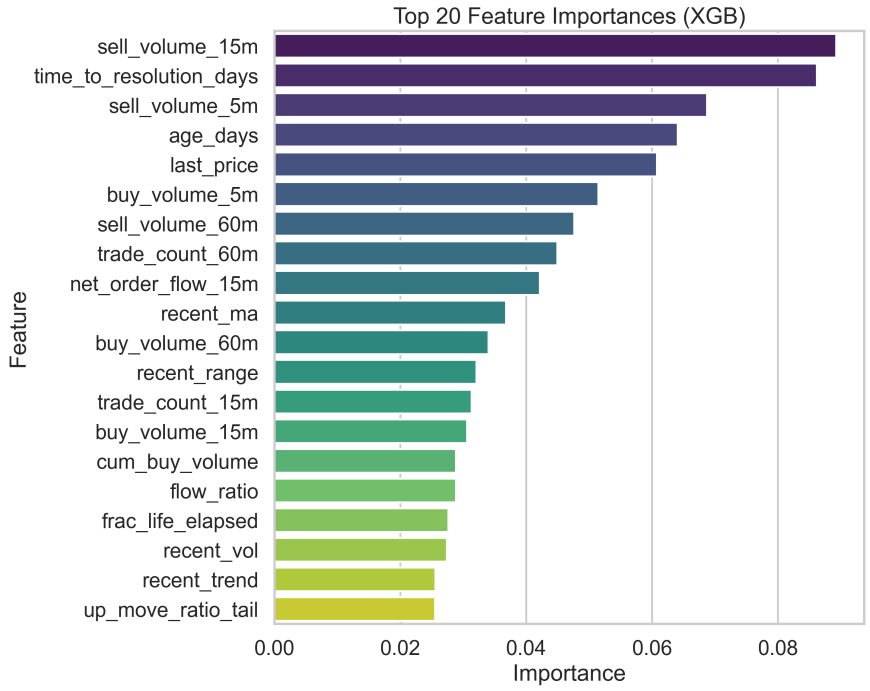

Figure 8: Top 20 feature importances from XGBoost.

SHAP values provide more detailed interpretability. The summary plot (Figure 9) highlights non-linear interactions, while the dependence plot reveals how specific features influence mispricing.

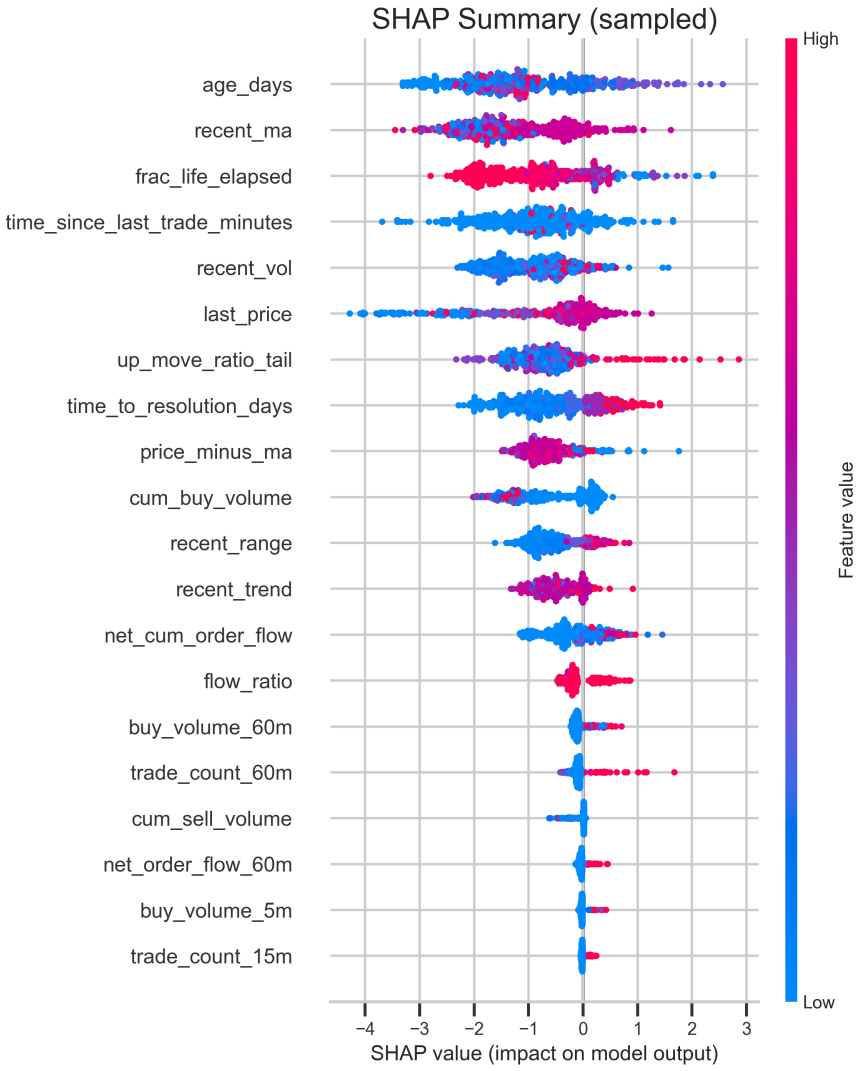

Figure 9: SHAP summary plot.

10 Backtesting Analysis

We evaluate three strategies:

- **Oracle:** trade all true mispriced snapshots.
- **Model @ F1 threshold (0.50):** balanced precision/recall.
- **Model @ high precision (0.70):** fewer high-confidence trades.

Figures 10 and 11 illustrate the distribution and total magnitude of optimistic PnL under each approach.

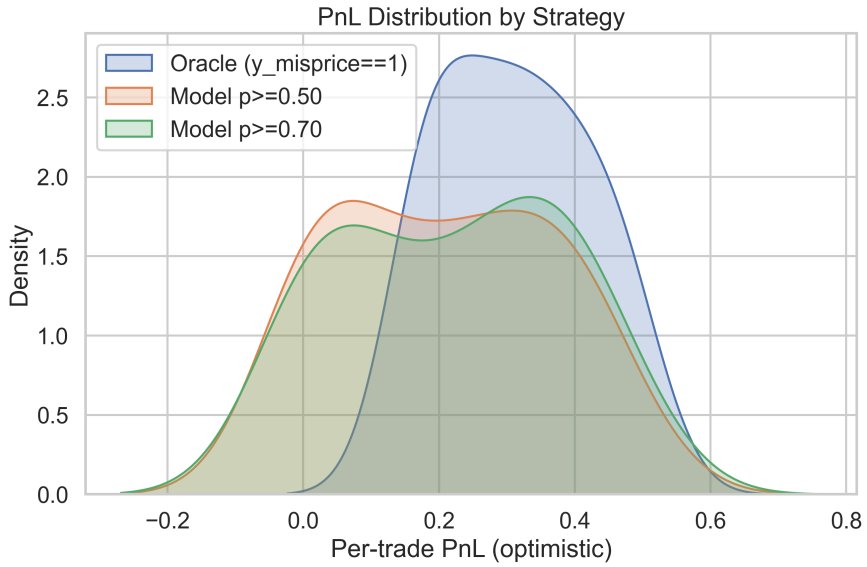

Figure 10: Distribution of per-trade optimistic PnL.

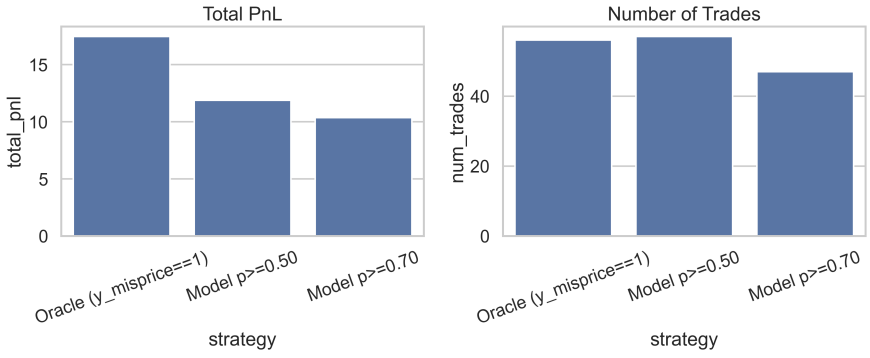

Figure 11: Total PnL and number of trades.

Cost-sensitive thresholding (Figure 12) demonstrates how preferences between false positives and false negatives affect optimal threshold selection.

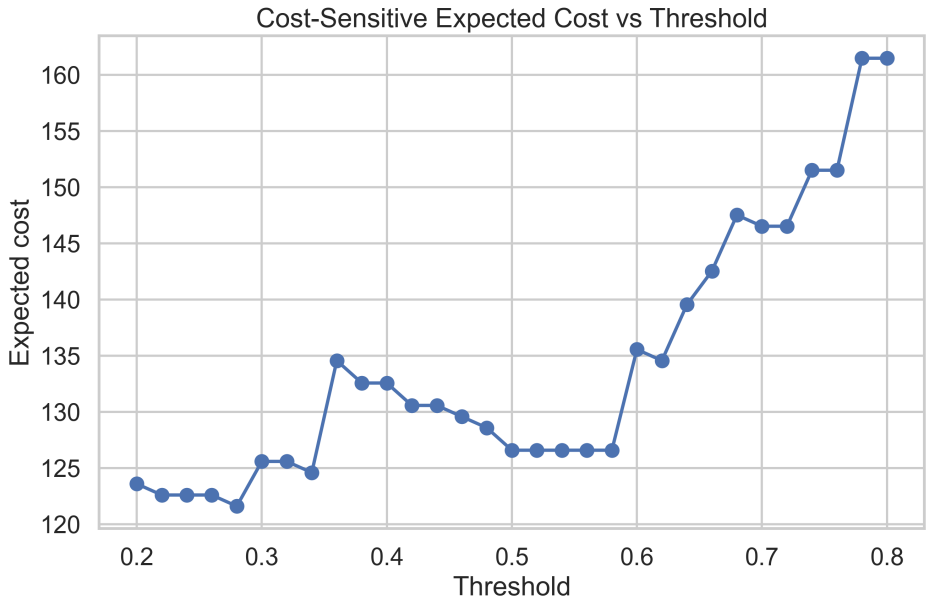

Figure 12: Cost-sensitive threshold analysis.

11 Conclusion

The optimized XGBoost model successfully identifies mispriced Polymarket snapshots with high discriminatory power despite strong class imbalance. The model integrates diverse signals from price dynamics, liquidity, and order flow. SHAP values highlight interpretable relationships, and backtesting confirms the practical significance of threshold selection.

Limitations include the lack of live execution costs, potential temporal leakage due to snapshot ordering, and the assumption of optimistic PnL. Future work includes: incorporating on-chain liquidity flows, performing walk-forward backtesting, modeling directional returns instead of absolute mispricing, and deploying a real-time monitoring pipeline.

References