Upset Predictions: A Machine Learning System for UFC Fight Outcome Forecasting

Version 2.1 — March 2026

Upset MMA (upsetmma.app)

Abstract

This paper presents Upset Predictions, a machine learning system for predicting the outcomes of Ultimate Fighting Championship (UFC) mixed martial arts bouts. The system employs a validation-selected XGBoost primary scorer with 283 features selected from 383 candidates, a Glicko-2 fighter rating system, and Bayesian smoothing techniques in a unified prediction pipeline. The current production model (v20260306) achieves 64.8% test accuracy across 318 held-out fights spanning 26 cards, with a backtest mean accuracy of 61.7% across four rolling temporal folds. We detail the motivation behind each architectural choice, the feature engineering methodology informed by domain-specific MMA analytics research, and the measures taken to prevent data leakage and ensure train/serve parity. The model is served via a FastAPI endpoint and retrained weekly through an automated CI pipeline.

1. Introduction

Predicting the outcome of mixed martial arts bouts is a particularly challenging binary classification task. Unlike team sports with large game samples, UFC fighters compete only 2–3 times per year, making per-fighter sample sizes extremely small. Fights can end via knockout, submission, or decision, introducing multi-modal outcome distributions. Fighter skill evolves non-linearly with age, injuries, and training camp changes, and matchup dynamics (e.g., striker vs. grappler) create interaction effects that aggregate statistics alone cannot capture.

The benchmark for UFC fight prediction accuracy is established by Las Vegas sportsbook closing lines, which historically resolve at approximately 65–70% accuracy [20]. The best publicly reported ML systems achieve comparable accuracy in the 65–70% range using gradient-boosted tree ensembles and comprehensive fight statistics [18].

Upset Predictions addresses these challenges through:

- **Deep feature engineering** — 283 selected features from 383 candidates across 16 categories, informed by domain-specific MMA analytics research, particularly the work of Latshaw [14–17].
- **Glicko-2 ratings** — latent fighter skill modeling with uncertainty, providing a principled alternative to Elo for fighters with irregular competition schedules [8, 9].
- **Bayesian smoothing** — Beta-Binomial and Poisson-Gamma conjugate priors to handle debut fighters and low-sample-size statistics [7].
- **Ensemble learning** — XGBoost and LightGBM with Optuna-tuned hyperparameters, leveraging the documented complementarity of tree-boosting algorithms [5, 6].

• **Strict leakage prevention** — point-in-time feature computation, random corner assignment, and removal of features unavailable at serving time.

2. Data Sources and Collection

2.1 UFCStats.com

All fight data is sourced from UFCStats.com, the official statistics provider for the UFC. The dataset comprises four tables:

Table	Records	Description
ufcFighters	4,451	Fighter profiles: height, reach, weight, DOB, stance, career record
ufcFights	8,494	Historical fight results: winner, method, round, time, weight class
ufcFightStats	16,941	Per-round statistics: strikes (head/body/leg), takedowns, submissions, control time
ufcEvents	~700	Event metadata: date, location, name

Data is synchronized to a Convex database via automated scraping actions using Cheerio for HTML parsing. Scheduled cron jobs run every 12 hours to capture new events and results.

2.2 Training Window

The model trains on fights from January 2014 to present, a cutoff chosen to reflect the modern era of UFC judging criteria and the post-USADA testing regime (introduced July 2015). The data is split chronologically:

- **Training set**: 2014 – July 2024 (5,058 fights)
- **Validation set**: July 2024 – July 2025 (515 fights)
- **Test set**: July 2025 – present (318 fights)

This time-based split is critical for temporal data to prevent future information leakage that would inflate reported accuracy [4]. Random shuffled splits would allow the model to learn patterns from future fights when predicting past ones.

2.3 Data Excluded

Three data sources were explicitly removed after audit:

- **Full career record** (`recordWins`/`recordLosses`): These are current-day totals, not point-in-time values, constituting direct data leakage.
- **Fighter rankings**: Only a current snapshot exists with no historical ranking data. Using current rankings for past fights is leakage.
- **Betting odds**: While powerful predictors during training, they are unavailable at serving time when predictions are generated before odds are published, creating train/serve distribution skew.

3. Feature Engineering

The feature engineering pipeline generates 283 selected features (from 383 pre-pruning candidates) across 16 categories for each matchup. Every feature is computed **point-in-time**: only data available before the fight date is used.

3.1 Positional Bias Elimination

Before computing features, fighters are randomly assigned to "Fighter A" and "Fighter B" positions using a fixed random seed, replacing the original red/blue corner assignments. The target is derived directly from the `winnerFighterId` field:

```
target = 1 if fighterA == winnerFighterId else 0
```

UFC red corner assignment correlates with favoritism (the higher-ranked fighter typically receives the red corner). Randomization eliminates this confound. A critical bug discovered during audit revealed that 1,194 of 5,891 training labels (20.3%) were inverted due to an incorrect assumption. The fix derives the target from `winnerFighterId` directly, which is correct regardless of corner assignment.

3.2 Physical Attributes

Feature	Description
`height, reach, height_diff, reach_diff`	Height/reach in inches and differentials
`weight`	Weight in pounds
`stance, stance_mismatch`	Stance encoding (Orthodox/Southpaw/Switch), mismatch flag

Physical attributes are immutable or near-immutable and require no point-in-time filtering. Reach advantage is one of the strongest single physical predictors in MMA — a differential of 4+ inches confers a statistically significant striking advantage at distance [12].

3.3 Record and Experience (Point-in-Time)

Feature	Description
`ufc_wins, ufc_losses, ufc_total_fights`	UFC record before fight
`ufc_win_pct, current_streak`	Win percentage, win/loss streak
`form_last3, form_last5`	Win rate in last 3/5 fights
`is_debut, days_since_last_loss`	First UFC fight flag, days since last loss

Computed from `ufcFights` filtered by `eventDateMs < fight_date_ms`, ensuring no leakage of future results.

3.4 Striking Statistics (Point-in-Time)

Feature	Description
`slpm, sapm`	Significant strikes landed/absorbed per minute
`str_acc, str_def`	Striking accuracy and defense
`str_diff_pm`	Striking differential per minute

Defense stats (`sapm`, `str_def`) are computed using **opponent fight stats** via the `opponentId` field. UFCStats records strikes from the attacker's perspective, so computing defense correctly requires looking up what the opponent actually landed.

3.5 Grappling Statistics (Point-in-Time)

Feature	Description
`td_avg, td_acc, td_def`	Takedowns per fight, accuracy, defense
`sub_avg`	Submission attempts per fight

3.6 Recent Form (EWMA)

For each fighter, the 5 most recent fights are processed through an Exponentially Weighted Moving Average (EWMA) with $\alpha = 0.3$, plus simple averages, standard deviations, and trend features across 22 statistical categories.

```
EWMA_1 = x_1
EWMA_t = alpha * x_t + (1 - alpha) * EWMA_(t-1)
```

where $\alpha \in (0, 1)$ controls the decay rate. With $\alpha = 0.3$, the most recent fight receives 30% weight, the second 21%, the third 14.7%, and so on [10]. EWMA naturally discounts stale data while preserving long-term patterns. Trend features (`trend = last_value − EWMA`) capture whether a fighter is currently performing above or below their weighted average.

3.7 Win Method Profile

KO/TKO, submission, and decision win percentages computed from historical `resultMethod` filtered by date and winner. These encode a fighter's finishing ability and preferred path to victory.

3.8 Glicko-2 Rating Features

Feature	Description
`glicko_mu, glicko_phi, glicko_sigma`	Rating mean, deviation, volatility
`glicko_mu_diff, glicko_phi_diff`	Rating and uncertainty differentials
`glicko_win_prob_red`	Analytical win probability from Glicko-2

See Section 4 for the full Glicko-2 specification.

3.9 Momentum and Activity

Feature	Description
`mu_trend`	Glicko-2 μ change over last 2 fights
`days_since_last`	Days since last fight
`fights_last_year`	Number of fights in past 12 months

Ring rust (prolonged inactivity) is a well-documented phenomenon in combat sports. Fighters returning after long layoffs show statistically lower win rates [13].

3.10 Matchup-Specific and Style Features

Features including `physical_advantage` (combined height + reach differential), `style_clash` (absolute difference in grappler ratio), and `versatility_diff` capture interaction effects between fighter styles. A `versatility_score` based on Shannon entropy of the distance/clinch/ground distribution measures how evenly distributed a fighter's activity is across ranges.

3.11 Age Curves

Chronological age, a `past_prime` flag (age > 33), `fight_age` (years since first UFC fight), and `fight_age_decline` (fight age > 9.5 years). Research on athletic aging in MMA suggests peak performance between ages 28–33 [13].

3.12 Opponent Quality (Strength of Schedule)

`opp_avg_mu` (average Glicko-2 rating of past opponents) and `quality_adj_win_pct` (win rate weighted by opponent rating) adjust for schedule difficulty. A 10–0 record against unranked opponents is fundamentally different from 10–0 against former champions.

3.13 SAOE/SDOE (Over-Expected Metrics)

Adapted from Latshaw [17], Strike Accuracy Over Expected (SAOE) measures a fighter's accuracy relative to expected accuracy given *where* they throw strikes:

```
SAOE = observed_accuracy − expected_accuracy
```

```
expected = Σ(range_att × baseline_range_acc) / total_att
```

where baselines are computed from the prior year's weight-class averages. Ground strikes land at ~65% while distance strikes land at ~45% — SAOE strips out this style bias to reveal true precision. SDOE mirrors SAOE for the defensive side.

3.14 Advanced Rate Statistics

`knockdown_rate` (knockdowns per striking opportunity), `distance_time_pct` (proportion of fight spent standing), and `control_rate` (grappling dominance ratio) decompose fight dynamics into *where* and *how* a fight takes place [14–16].

3.15 Expected Rounds (xR)

A simplified approximation of Latshaw's JudgeAI round-scoring model [14, 15]:

```
round_score = 0.5 × (sig_str_diff / 15) + 0.3 × (ctrl_time_diff / 90) + 0.2 × (td_diff / 2)
```

```
round_win_prob = σ(3 × round_score)
```

```
xR_pct = mean(round_win_probs)
```

The 0.5/0.3/0.2 weighting reflects the UFC's judging criteria which prioritize effective striking, then grappling/control, then aggression.

3.16 Bayesian Smoothed Statistics

For fighters with limited history, raw statistics are unreliable. We apply Bayesian conjugate prior smoothing to regularize estimates toward division averages [7].

Beta-Binomial model for rates (e.g., striking accuracy):

```
smoothed_rate = (successes + prior_rate × pseudo_count) / (attempts + pseudo_count)
```

Poisson-Gamma model for counts (e.g., knockdowns per fight):

```
smoothed_count = (observed_total + prior_mean × pseudo_fights) / (n_fights + pseudo_fights)
```

Feature	Model	Pseudo Count
`bayes_str_acc`	Beta-Binomial	15
`bayes_td_acc`	Beta-Binomial	8
`bayes_slpm`	Poisson-Gamma	5 fights
`bayes_td_avg`	Poisson-Gamma	5 fights
`bayes_knockdowns`	Poisson-Gamma	5 fights

Feature	Model	Pseudo Count
bayes_ctrl_time	Poisson-Gamma	5 fights

Division priors are computed from the prior year's fights within the same weight class. Debut fighters (0 fights) receive division prior values directly rather than NaN, ensuring informative baseline estimates.

3.17 Feature Pruning

After initial feature computation (383 features), a quick XGBoost model computes feature importances. Features with importance below 0.001 are pruned, reducing the feature set to **283 selected features** in the current production artifact. This aggressive pruning reduces noise and helps prevent overfitting [5].

4. Fighter Rating System (Glicko-2)

4.1 Background

The Glicko-2 rating system [8] extends Elo with two additional parameters well-suited for combat sports:

- **Rating (μ)**: Estimated skill level. Initial value: 1500.
- **Rating Deviation (ϕ)**: Uncertainty in the estimate. Initial value: 350. High ϕ indicates an unreliable rating.
- **Volatility (σ)**: Expected fluctuation. Initial value: 0.06. High σ indicates inconsistent performance.

Standard Elo treats all ratings as equally certain, which is problematic when a fighter hasn't competed in 18 months. Glicko-2's ϕ parameter naturally encodes this: ϕ drifts upward during inactivity, widening the confidence interval and making the next fight result produce a larger rating update.

4.2 Mathematical Formulation

Scale conversion for numerical stability:

```
mu_2 = (mu - 1500) / 173.7178
phi_2 = phi / 173.7178
```

Key functions:

```
g(phi) = 1 / sqrt(1 + 3*phi^2 / pi^2)
E(mu, mu_opp, phi_opp) = 1 / (1 + exp(-g(phi_opp)(mu - mu_opp)))
```

Volatility update uses the Illinois algorithm (a regula falsi variant) for iterative root-finding, converging to tolerance e = 10^-6. System constant tau = 0.5.

4.3 MMA-Specific Adaptations

- **Inactivity decay**: For fighters inactive > 2 years, ϕ increases by 50 points per additional year, capped at 350.
- **Outcome encoding**: Win = 1.0, Loss = 0.0, Draw = 0.5, No Contest = skipped.

Win probability between fighters A and B:

```
P(A wins) = E(mu2_A, mu2_B, sqrt(phi2_A^2 + phi2_B^2))
```

This combines both fighters' uncertainties — a matchup between well-known fighters produces more confident predictions than one involving newcomers.

5. Model Architecture

5.1 XGBoost

XGBoost (eXtreme Gradient Boosting) [5] is a regularized gradient boosting framework selected for: (1) native missing value handling — learning optimal default directions during tree construction, critical for debut fighters with NaN statistics; (2) L1 and L2 regularization on the objective:

```
Obj = Sum l(y_i, y_hat_i) + Sum [gamma*T + 0.5*lambda*||w||^2 + alpha*||w||_1]
```

where T is the number of leaves, w are leaf weights, γ penalizes tree complexity, λ is L2 regularization, and α is L1 regularization; (3) sparsity-aware split-finding; and (4) binary logistic objective for direct probability output.

5.2 LightGBM

LightGBM [11] is selected as the second ensemble member for complementary strengths: leaf-wise tree growth (vs. XGBoost's level-wise), Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB). Despite similar theoretical foundations, their different tree growth strategies produce meaningfully different decision boundaries. Ensembling exploits this complementarity — errors made by one model are often corrected by the other [6].

6. Hyperparameter Optimization

Hyperparameters are tuned using Optuna [1], a define-by-run framework using the Tree-structured Parzen Estimator (TPE) sampler [2]. Each model is tuned with 100 trials. Optimization target: validation set log loss (minimized).

Parameter	Range	Scale
n_estimators	100 – 1000	Linear
max_depth	3 – 10	Linear
learning_rate	0.01 – 0.3	Log
subsample	0.6 – 1.0	Linear
colsample_bytree	0.5 – 1.0	Linear
min_child_weight	1 – 10	Linear
gamma	0 – 5	Linear
reg_alpha / reg_lambda	10^-8 – 10	Log

Table 1: XGBoost hyperparameter search space.

7. Ensemble Strategy

The final prediction is a weighted average of both models' predicted probabilities:

```
P_ensemble = w × P_xgb + (1 − w) × P_lgb
```

The optimal weight w is determined by grid search over [0.3, 0.4, 0.5, 0.6, 0.7], minimizing validation set log loss. The constrained range ensures neither model dominates, preserving the diversity benefit of ensembling. Simple weighted averaging is preferred over stacking because with a validation set of ~500 fights, a meta-learner would likely overfit [21].

Feature importance is aggregated as a weighted average of normalized importances from both models, used for SHAP-like feature explanations in the user-facing application.

8. Probability Calibration

Platt scaling [19] fits a logistic regression on ensemble output probabilities:

```
P_calibrated = σ(a × P_ensemble + b)
```

This was implemented but **disabled** after audit revealed the calibrator was fitted and evaluated on the same validation set — a methodological error causing overfitting. Raw ensemble probabilities are used instead. Calibration quality is still evaluated via reliability diagrams, mean calibration error, and Brier score decomposition [3].

9. Evaluation Methodology

9.1 Metrics

Metric	Formula	Interpretation
Accuracy	mean(■ == y)	Percentage of correct winner predictions
Log Loss	–mean(y·log(p) + (1–y)·log(1–p))	Penalizes confident wrong predictions
Brier Score	mean((p – y)²)	Mean squared probability error
Calibration Error	mean(|bin_true – bin_pred|)	Predicted vs. observed probability gap

Table 2: Evaluation metrics. Lower is better for all except Accuracy.

9.2 Benchmarks and Results

Method	Accuracy	Source
Random baseline	~50%	—
Always pick favorite (by record)	~55–58%	—
Academic ML models	~60–66%	Hitkul et al. [10], Zhan et al. [22]
Upset Predictions (current)	**64.8%**	This work
Vegas closing odds	~67–70%	Pinnacle [20]
MMA-AI (best public)	~70%	mma-ai.net [18]

Table 3: Comparison of UFC prediction methods.

Current production metrics (model v20260306, XGBoost primary):

- Test accuracy: 64.8% (95% CI: 60.1%–69.5%) on 318 fights across 26 cards
- Test log loss: 0.644
- Test Brier score: 0.226
- Test calibration error: 0.041
- Backtest mean accuracy: 61.7% across 4 rolling folds (mean log loss: 0.657)
- Training data: 5,058 (train), 515 (val), 318 (test), 5,891 (production refit)

With a test set of ~300 fights, the 95% confidence interval on accuracy is approximately ±5 percentage points using the normal approximation to the binomial.

9.3 Confidence Classification

- **High**: |P – 0.5| > 0.2 (probability < 30% or > 70%)

- **Medium**: $0.1 < |P - 0.5| \leq 0.2$
- **Low**: $|P - 0.5| \leq 0.1$

10. Training Pipeline and Data Integrity

10.1 Data Leakage Prevention

Five safeguards prevent data leakage:

- **Point-in-time features**: Every feature filtered by `eventDateMs < fight_date_ms`.
- **Random corner assignment**: Fighters randomly swapped to prevent positional bias.
- **Temporal split**: No random shuffling. The model never sees future data.
- **Removed leaky features**: Career totals, rankings, and odds audited and removed.
- **Label integrity**: 1,194 of 5,891 labels (20.3%) were corrected after discovering an inversion bug where `redFighterId` was incorrectly assumed to equal `winnerFighterId`.

10.2 Train/Serve Parity

The serving code mirrors training feature engineering exactly. Both pipelines use identical utility functions, EWMA parameters ($\alpha = 0.3$, $n = 5$), Bayesian pseudo-counts, stance/weight-class encodings, and defense stat computation via opponent lookups. A dedicated test suite validates that identical inputs produce identical feature vectors in both pipelines.

11. Serving Architecture

The model is served via a FastAPI application deployed as a Docker container. Three endpoints are exposed:

- `POST /predict` — Accepts fight matchups, returns win probabilities, confidence levels, top feature explanations, and method probabilities.
- `GET /health` — Health check with model version and fighter/rating counts.
- `GET /ratings/{fighter_id}` — Returns a fighter's current Glicko-2 rating.

On startup, model files, fighter data (4,451 fighters), Glicko-2 ratings, fight history, and feature importance are loaded into memory. For each prediction, 283 features are computed, both models generate probabilities, the ensemble combines them, and top-5 features are extracted for explainability.

11.1 Method Probability Estimation

Win method probabilities (KO, submission, decision) are estimated from historical method profiles, weighted by the model's win probability:

```
method_prob_k = winner.method_k × P_win + loser.method_k × (1 − P_win)
```

for each method $k \in \{\text{KO, SUB, DEC}\}$.

12. Continuous Retraining

A GitHub Actions workflow runs the full retraining pipeline weekly (Tuesdays at noon UTC): data export, Glicko-2 computation, feature engineering, Optuna tuning (100 trials), test suite execution, metric comparison, quality gate enforcement, and artifact upload. Each model receives a version string `v{YYYYMMDD_HHMMSS}` with comprehensive metadata. Models failing the quality gate are not deployed.

13. Limitations and Future Work

13.1 Known Limitations

• **Small test set**: ~300 fights yield ±5% confidence intervals. Statistical significance of improvements is difficult to establish.

• **No historical rankings**: Only a current snapshot exists. Time-stamped rankings would provide a powerful feature.

• **No camp/coaching data**: Training camp quality, coaching changes, and weight cuts are not captured.

• **No injury data**: Fighters frequently compete with undisclosed injuries.

• **Train/serve feature distribution gap**: Serving uses all available data (correct for live predictions), while training uses point-in-time features.

13.2 Future Improvements

• Platt scaling on a held-out calibration subset carved from training data.

• Historical rankings pipeline for point-in-time ranking features.

• SHAP values [17] for per-prediction explanations (replacing feature importance).

• Neural network ensemble member for non-tree-based interaction patterns.

• Fight-specific context features: altitude, travel distance, event prestige.

• Odds integration if timing constraints can be resolved.

14. References

[1] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *KDD '19*, pp. 2623–2631.

[2] Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *NeurIPS 2011*, pp. 2546–2554.

[3] Brier, G. W. (1950). Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1), 1–3.

[4] Cerqueira, V., Torgo, L., & Mozeti■, I. (2020). Evaluating Time Series Forecasting Models. *Machine Learning*, 109, 1997–2028.

[5] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD '16*, pp. 785–794.

[6] Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *MCS 2000*, LNCS 1857, pp. 1–15.

[7] Gelman, A., Carlin, J. B., Stern, H. S., et al. (2013). *Bayesian Data Analysis* (3rd ed.). Chapman and Hall/CRC.

[8] Glickman, M. E. (2001). Dynamic Paired Comparison Models with Stochastic Variances. *Journal of Applied Statistics*, 28(6), 673–689.

[9] Glickman, M. E. (2012). Example of the Glicko-2 System. Technical report, Boston University.

[10] Hunter, J. S. (1986). The Exponentially Weighted Moving Average. *Journal of Quality Technology*, 18(4), 203–210.

[11] Ke, G., Meng, Q., Finley, T., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *NeurIPS 2017*, pp. 3149–3157.

[12] Kirk, C. (2018). Does Stature or Wingspan Length Have a Positive Effect on Competitor Rankings in MMA? *Sport Sciences for Health*, 14, 495–501.

[13] Lenetsky, S., Harris, N., & Brughelli, M. (2015). Assessment and Contributors of Punching Forces in Combat Sports Athletes. *Strength and Conditioning Journal*, 37(2), 1–7.

[14] Latshaw, N. (2021). Introducing Expected Rounds (xR). *Literal Fight Nerd*.

[15] Latshaw, N. (2021). Introducing JudgeAI. *Literal Fight Nerd*.

[16] Latshaw, N. (2022). Visualizing Fighter Styles. *Literal Fight Nerd*.

[17] Latshaw, N. (2025). Introducing Strike Accuracy Over Expected (SAOE). *Literal Fight Nerd*.

[18] MMA-AI (2024–2026). AI-Powered UFC Fight Predictions. mma-ai.net. Reported accuracy: ~70%.

[19] Platt, J. C. (1999). Probabilistic Outputs for Support Vector Machines. *Advances in Large Margin Classifiers*, pp. 61–74. MIT Press.

[20] Pinnacle Sports (2024). How Often Does the Favourite Win in UFC? Historical favorite win rate: 67–70%.

[21] Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, 5(2), 241–259.

[22] Zhan, J. et al. (2024). Data-Driven MMA Outcome Prediction Enhanced by Fighting Style Clustering. *MLISE 2024*. Accuracy: 65.52%.

[23] Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *NeurIPS 2017*, pp. 4765–4774.

[24] Hitkul, Jain, S., & Sharan, A. (2018). A Comparative Study of ML Algorithms for Prior Prediction of UFC Fights. *ICHSA 2018*, pp. 67–76.

Upset Predictions is developed by Upset MMA (upsetmma.app). Data sourced from UFCStats.com. Glicko-2 implementation based on Glickman [8, 9].