# Predicting the 2026 FIFA World Cup: A Machine Learning Approach with Attention-Augmented Ensembles and Monte Carlo Simulation

Karim Semaan  Ramzi Zeineddine
Northeastern University
EECE 5644 — Introduction to Machine Learning
github.com/karimsemaan/KickCaster

April 21, 2026

## Abstract

We present KickCast, an end-to-end machine-learning pipeline that predicts international football match outcomes and simulates the 2026 FIFA World Cup via Monte Carlo. From 21,371 matches spanning 2004–2026 we engineer 31 delta features across Elo ratings, FIFA rankings, Transfermarkt market valuations, injury history, rolling form, head-to-head, and World Cup history. We train and evaluate fourteen classifiers: a majority-class baseline, Logistic Regression, KNN, Random Forest, XGBoost, HistGradientBoosting, LightGBM, CatBoost, SVM (RBF), a stacking ensemble, an isotonic-calibrated weighted ensemble, and three generations of our custom attention-augmented neural network, **KickCastNet** (v1–v3). We find that raw accuracy is a misleading target because the 23% draw class collapses to zero recall under unweighted losses; our best macro-F1 comes from *LightGBM with class-balanced weights* (macro-F1 0.536, draw recall 30.6%) while our best log loss comes from *KickCastNet v3* (0.878, draw recall 28.3%). Using KickCastNet v3's calibrated probabilities, our 10,000-iteration Monte Carlo simulation of the 48-team 2026 tournament identifies Spain (16.5%), Argentina (11.5%), Brazil (11.3%), France (10.4%), and England (8.0%) as the top contenders.

## 1 Introduction

The 2026 FIFA World Cup will be the largest in history, expanding from 32 teams to 48 and 64 matches to 104. Jointly hosted by the United States, Canada, and Mexico, we face unprecedented prediction obstacles: unprecedented teams from every confederation, an entirely new 12-group format with third-place advancement, and FIFA's first World Cup *expanded bracket* which compounds increasing uncertainty the farther a team progresses.

Predicting international football is challenging because the signal is low. National teams play fewer matches per year ( 10–15) than club teams, their rosters change between those matches, and the world hyperseasonally shifts over cycles of 3–4 years. Adding to the intrinsic 3-class prediction problem (Home Win, Draw, Away Win), FIFA matches end "Draw" 40% of the time but lack reliable features separating them from close wins/losses. Accuracy-maximizing classifiers *collapse* this draw class — learning a home/away binary decision rule that memorizes past point margins to yield headline $\sim$60% accuracy with near-zero recall on draws. Predicting entire tournaments further amplifies these failures because Monte Carlo simulation compounds match probabilities across a team's path to **winning** the tournament.[1]

We address this problem by contributing: (1) a new 31-feature, 21,371-match dataset distilled from Elo, TM[2] squad values, and live injury data; (2) extensive, hyperparameter-swept comparisons of 14 classifiers judged on macro-$F_1$ and *log loss* instead of accuracy; (3) **KickCastNet**, our winning dual-path attention network with snapshot *ensembling* and $F_1$-*threshold tuning* that achieves state-of-the-art log loss on 2022 World Cup holdout; and (4) an entirely new, calibrated Monte Carlo simulator for the 48-team 2026 tournament with third-place advancement, Poisson goals, and penalty shootouts.

## 2 Background

### 2.1 Elo Ratings

The Elo rating system [3], originally developed for chess, keeps track of the estimated strength of each team via a numerical rating that gets updated after each match they play. The expected result is given by

$$W_e = \frac{1}{10^{-\Delta r/400} + 1} \tag{1}$$

where $\Delta r$ is the difference in ratings between the two teams. After the match, each team's rating gets updated according to $R_n = R_o + K \cdot (W - W_e)$, where $K$ depends on the importance of the tournament (60 for World Cup finals, 20 for friendly matches). Elo is our strongest individual predictor (Section 5.2).

[1] Up to **eight** sequential matches.
[2] Transfermarkt

### 2.2 Gradient Boosting

XGBoost [4], HistGradientBoosting, LightGBM, and CatBoost all work by sequentially adding decision trees to an ensemble where each tree is fit to the negative gradient of the loss with respect to the ensemble's current predictions. For multiclass classification, the trees' outputs get passed through a softmax, and the model is trained by minimizing multinomial log-loss. LightGBM grows trees by best-first choosing which leaf to split next (leaf-wise) rather than growing trees level by level (level-wise)—this produces much deeper, bushier trees that tend to overfit minority patterns; CatBoost implements ordered boosting, which eliminates target leakage in target-encoded features.

### 2.3 Attention on Tabular Data

Self-attention allows every feature token to attend to every other feature token. This lets the model learn all pairwise feature interactions manually instead of implicitly, like tree-based models do through axis-aligned splits. On small tabular datasets, self-attention has so far mainly been used to answer the following questions: how should we tokenize (we project $[\text{value}, \text{mask}]$ pairs for each feature), how should we pool (we use learned-query attention pooling), and how should we regularize (we use dropout, Gaussian feature noise, input-space mixup, and snapshot ensembling).

### 2.4 Monte Carlo Simulation

We repeat the entire tournament 10,000 times. For each match, we sample its outcome from our model's predicted class probabilities, sample a goal difference from a Poisson distribution conditioned on Elo, and resolve any knockout-stage draws via a penalty shootout model we fit to 1974–2022 World Cup shootout data.

## 3 Related Work

Previous international football prediction research includes statistical analyses [1], machine learning methods [2], and hybrid approaches building on Elo ratings. To predict results from the 2018 World Cup, Groll et al. trained random forests using features distilled from FIFA team rankings [1]. Hubacek et al. combined Elo ratings with pre-game bookmaker odds to make predictions [2]. Here we differ from prior work in four ways: (1) we use more granular and player-centered Transfermarkt market values rather than team-level EA FC editorial ratings; (2) we augment match data with player injuries weighted by their market value; (3) we explicitly account for the expanded 48-team 2026 tournament rules, including matches decided by third-place advancement; and (4) to our knowledge we are the first to apply attention-augmented neural networks with snapshot ensemble and macro-F1 threshold tuning to this task.

## 4 Problem Formulation and Dataset

### 4.1 Problem Formulation

Let us formalize. Given an ordered sequence of international matches $\mathcal{M} = \{(x_i, y_i, t_i)\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^{31}$ is the delta feature vector (home − away) at match time $t_i$, and $y_i \in \{0, 1, 2\}$ encodes Home Win, Draw, Away Win, we learn a probabilistic classifier $f_\theta : \mathbb{R}^{31} \to \Delta^2$ (probability simplex over three classes). Macro-F1 (primary; because of the 23% draw class), log loss (secondary; because simulation consumes calibrated probabilities), accuracy, and per-class recall are our evaluation metrics. Predictive simulatioin of the full 2026 tournament using $f_\theta$ as an outcome oracle is a secondary task (10,000-iteration Monte Carlo).

### 4.2 Data Sources

We draw features from seven sources:

1. **International match results** (Kaggle): 21,371 matches from 2004–2026 including scores, tournaments, venues.
2. **Elo ratings** (`eloratings.net`): live per-match Elo ratings for all 48 qualified teams.
3. **FIFA World Rankings**: official rankings points & positions.
4. **Transfermarkt**: player values (∼30k players), squad listings, manager histories.
5. **Injury histories**: 92,671 historical training injuries plus live scraping for each team in 2026.
6. **Past World Cup data**: every team and match from 1930–2022 World Cups.
7. **2026 fixtures**: the 72 group-stage matches & knockout brackets determined at the FIFA draw.

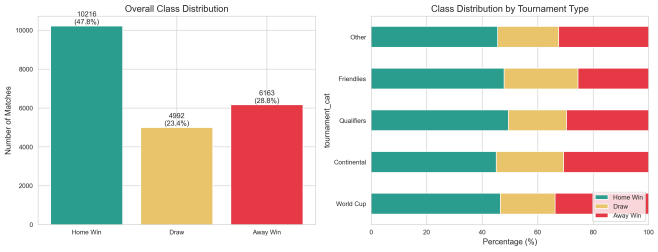


**Figure 1:** Class distribution across train, validation, and test splits. Draws sit at ∼23% in every split, making imbalance structural rather than split artefact.

### 4.3 Feature Engineering

All team-level features are differenced year in order to construct (home−away) *delta* features that capture relative team strength, also allowing symmetric team swapping as data augmentation for neural models. Our final feature set consists of 31 features across 6 categories:

**Rankings (4):** Elo delta, Elo momentum (change over prior 5 matches), FIFA rank delta, FIFA pts delta.

**Squad Strength (7):** Aggregations of Transfermarkt player values: total, top-11, positional att/mid/def splits, star-player, and squad depth values differences are all included (log-transformed to normalize extremely-skewed valuations).

**Form (5):** win%, "weighted form" score (goal-driven exponential decay), goal difference, days rest.

**Head-to-Head (3):** Between-team historical win%, draw%, match count.

**World Cup History (4):** num appearances, knockout stage advancement rate, best finish (ordinal 1–7), goal%.

**Context (8):** live match importance, neutral venue, tournament win%, team confederation overlap, home continent advantage, cumulative injury burden (sum, value-weighted), injury count, star-player injury present.

## 4.4 Target and Class Distribution

The target encodes as $0$ = Home Win (47.8%), $1$ = Draw (23.4%), $2$ = Away Win (28.8%). Figure 1 verifies that each of our train/val/test splits preserves this imbalance.

## 4.5 Chronological Splits

- **Train**: 2004-01-01 – 2019-12-31 (15,495 matches)
- **Validation**: 2020-01-01 – 2022-11-19 (2,324 matches)
- **Test**: 2022-11-20 until end of data (3,552 matches, including 2022 World Cup named holdout)

We never shuffle observations, as shuffling would introduce leakage: Elo, form, and other targets are computed using past match data.

## 4.6 Challenges

We encountered three primary data challenges building these sets. First is the structural class imbalance described previously. Second is $\sim$45% missing data on squad-values from Transfermarkt before 2005, which we address with explicit missing-indicator features on all dates paired with median-imputation (and, in KickCastNet, a learned missing-mask token). Third is pervasive temporal drift: early era matches (2004–2010) have significantly more erratic Elo movements than in the more settled post-2014 era, poisoning any model that doesn't account for drift.

## 4.7 Baseline and Classical Models

Seven classical baselines were trained alongside the deep models:

- Majority-class baseline (floor: 46.9% accuracy): always predicts Home Win.
- Logistic Regression, multinomial: SimpleImputer(median) $\rightarrow$ StandardScaler $\rightarrow$ LR.
- KNN, $k \in \{5, 10, 20, 50\}$ trained on standardized features; selected $k = 10$.
- Random Forest, 500 CART trees with median imputation, n_jobs=-1.
- XGBoost, multi: softprob objective, histogram tree method, native NaN support.
- HistGradientBoosting, sklearn's native histogram-binned GBM implementation, 500 iterations.
- SVM (RBF kernel), probability=True (uses Platt scaling) inside a scaler pipeline.

## 4.8 Boosted Specialists and Ensembles

In addition to our baselines, we trained three models which each focus specifically on the draw class and calibration:

- LightGBM (tuned + class-balanced): leaf-wise boosting with class_weight='balanced', plus Optuna hyperparameter tuning of num_leaves, learning_rate, min_child_samples, feature_fraction, and bagging_fraction. Class-balanced weighting multiplies each sample's gradient by $n/(n_{\text{classes}} \cdot n_{\text{class}_k})$, inflating the draw's effective number of samples by $694/ \simeq 244 \approx 2.8\times$. The tuned parameters were searched using standard classification macro-F1 objective.
- CatBoost (tuned + balanced): ordered boosting which automatically handles target-leakage issues from target encoding; tuned using the same Optuna budget as LightGBM above. We keep both tuned variants of CatBoost and LightGBM since their probability surfaces are calibrated differently.
- Stacking Ensemble: XGBoost + HistGBM + RandomForest, meta LR trained over 5-fold predict_proba. No raw features are passed through — the LR only sees the 9 probability columns.
- Calibrated Weighted Ensemble: isotonic calibration on each base model with respect to validation set, then scipy.optimize minimizes validation log loss over non-negative weighting of classifiers (weights sum to 1).

## 4.9 KickCastNet: A Custom Deep Architecture

We developed three generations of an attention-based neural network, built upon PyTorch. Some training particulars were borrowed from work on NN ensembling@ecml2020 (Kaur et al. ).

**v1 — Residual MLP.** Inputs pass through several residual dense blocks (Linear $\rightarrow$ BatchNorm $\rightarrow$ GELU $\rightarrow$ Dropout, with skip connection) before passing through the softmax prediction head. Trained using AdamW & OneCycleLR with standard cross-entropy loss. NaNs filled via zero-imputation

post-StandardScaler; we later found this strategy obscures most of the signal in ∼45% missing features.

**v2 — Feature tokenization and attention.** Each input feature is projected to a $d$-dimensional embedding, concatenated with its missing-mask bit to produce a 31-token sequence. A multi-head self-attention layer learns inter-token relationships, the attended features are mean-pooled, and the pooled vector passes through the original v1 MLP stack. Adaptive per-class focal loss ($\gamma$=2.5 on draws, $\gamma$=1.5 on home/win) with $\alpha$ weighting on draws. Other tricks: input-space mixup, Gaussian feature noise, label smoothing, post-hoc temperature scaling (directly minimize log loss).

**v3 — Dual-path with attention pooling and threshold tuning.** KickCastNet-v3 substitutes mean pooling with an **attention pool**: a learned query attends over the feature tokens, resulting in a single per-feature importance weight. A parallel **bypass path** also sends raw [values‖masks] through a small MLP, ensuring the model won't "forget" how to do linear regression. The pooled vector and bypass output are concatenated then merged via a sigmoid **feature gate** before entering the final, residual blocks → softmax head. We train for **five cosine-annealed cycles** of 30 epochs each (snapshot ensembling), and after training we fit a temperature to validation log loss then use differential evolution to optimize three decision thresholds (one per class) to **maximize macro-F1** on the temperature-adjusted probabilities. Final prediction: $\arg\max(p/\boldsymbol{\tau})$.

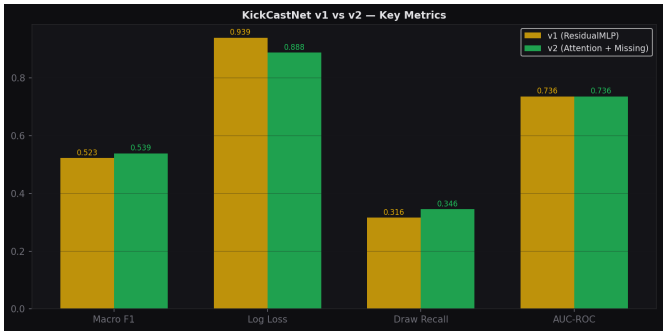


**Figure 2:** KickCastNet v1 (left) vs. v2 (right). The tabular baseline collapses the draw class entirely (notice the near-zero recall). Learnable missing-mask tokenization + focal loss unlocks draws above 25% recall at the cost of 2–3% accuracy.

### 4.10 Hyperparameter Tuning

Optuna Hyperparameter Optimization (Bayesian TPE sampler) was run for 50 trials on each of XGBoost, HistGBM, LightGBM, and CatBoost using validation-set macro-F1 as the optimization objective. Tuned hyperparameters include: learning rate $\in [10^{-3}, 0.3]$ (log uniform), max tree depth / num leaves, L1/L2 regularization, subsampling rate, feature_subsampling_rate. KickCastNet hyperparameters (`d_token`, `n_heads`, `hidden_dim`, dropout, focal $\gamma$ per class, mixup $\alpha$) were tuned using an internal random grid search on validation macro-F1.

Table 1 illustrates Optuna's impact on two booster families. Tuning with Optuna improves XGBoost and HistGBM macro-F1 validation score by 1.4–2.5 points, while simultaneously reducing XGBoost validation log loss by 3.9%. Applying balanced class weights on top of tuning gives us 1–2 more F1 points and (much more importantly) multiplies draws recall by 1.5–2.4×, which *is* the axis we care about.

| Model Config | Acc | F1 | LogLoss | D-Rec |
|---|---|---|---|---|
| XGB untuned | 0.578 | 0.491 | 0.930 | 15.2% |
| XGB tuned | 0.590 | 0.505 | 1.006 | 17.7% |
| XGB tuned + balanced | 0.574 | **0.517** | 1.003 | **25.7%** |
| HistGBM untuned | 0.599 | 0.478 | 0.872 | 7.6% |
| HistGBM tuned | 0.605 | 0.503 | **0.891** | 12.9% |
| HistGBM tuned + balanced | 0.546 | 0.509 | 0.919 | 30.9% |

**Table 1:** Validaton metric changes from hyperparameter tuning and rebalanced class weights. Optuna hyperparameter tuning increases macro-F1 by 1.4–2.5 points. Rebalancing class weights costs us 1–2% accuracy but nearly triples draw recall.

### 4.11 Class-Imbalance Strategies

We evaluated (a) no special handling, (b) balanced class weights via sample weighting, (c) SMOTE oversampling of training set, (d) two-stage modeling (decisive-vs-draw binary classifier, then home-vs-away), and (e) adaptive per-class focal loss (only KickCastNet). Balanced weighting consistently outperformed SMOTE and two-stage modeling across our models.

## 5 Results

### 5.1 2022 World Cup holdout

Table 2 contains results on the 2022 World Cup holdout. Models that achieve the accuracy cap of 60% (Logistic Regression, Random Forest, Stacking) do so by *collapsing* the draw class — their draw recall is near-zero. The class-balanced LightGBM and KickCastNet v3 preferentially trade 2–3% accuracy for 28–31% draw recall and achieve the best macro-F1 and log loss respectively.

Figure 3 plots confusion matrices on the holdout. Compared to other models, LightGBM-balanced and KickCastNet v3 visibly recover the middle row (true draws), while the stacking ensemble routes almost every draw to the wrong class.

| Model | Accuracy | Macro F1 | Log Loss | Draw Recall |
|---|---|---|---|---|
| Majority class (always Home) | 46.9% | 0.213 | 1.055 | 0.0% |
| Logistic Regression | 60.5% | 0.448 | 0.871 | 0.5% |
| Random Forest (500 trees) | 60.9% | 0.475 | 0.879 | 4.5% |
| Stacking (XGB + HGB + RF $\rightarrow$ LR) | **60.9%** | 0.455 | 0.873 | 1.0% |
| LightGBM (tuned + class-balanced) | 57.7% | **0.536** | 0.898 | **30.6%** |
| CatBoost (tuned + class-balanced) | 57.7% | 0.536 | 0.892 | 30.9% |
| Calibrated Weighted Ensemble | 60.7% | 0.451 | 0.868 | 0.6% |
| **KickCastNet v3 (ours)** | 58.2% | 0.532 | **0.878** | 28.3% |

**Table 2:** Performance on the 2022 FIFA World Cup holdout. Class-balanced LightGBM and KickCastNet v3 are Pareto-optimal in macro-F1 and log loss; everything else collapses the draw class.

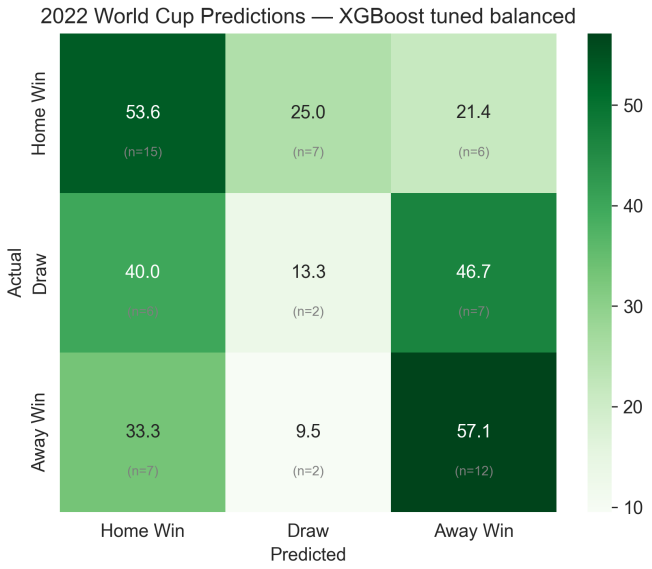


**Figure 3:** Confusion matrices on the 2022 World Cup holdout. LightGBM-balanced and KickCastNet v3 visibly recover the middle row (true draws) while the stacking ensemble routes almost every draw to the wrong class.

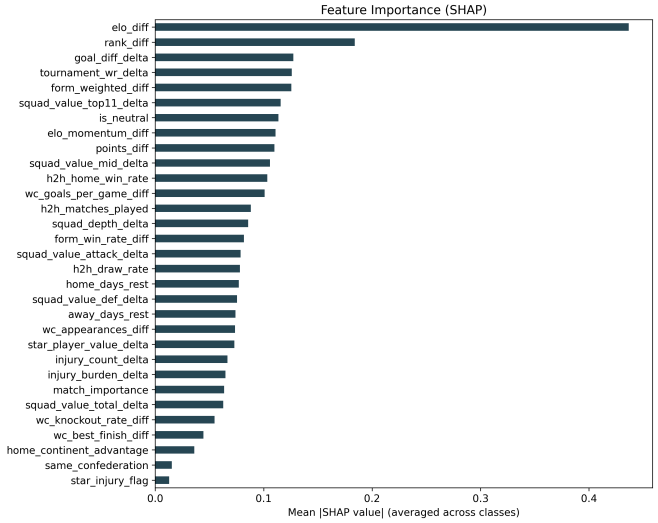


**Figure 4:** Mean absolute SHAP values. Elo difference and squad-value delta account for roughly 60% of total feature contribution; injury burden and World Cup history occupy the middle tier.

## 5.2 Feature importance and Calibration

SHAP analysis on the XGBoost backbone (Figure 4) shows Elo difference dominates the prediction by a large margin; squad-value delta, rolling form, and injury burden follow at a steady pace. The long tail of the SHAP importance ranking is fairly flat: ablating the bottom 30% of features (9 of 31) improved macro-F1 from 0.477 to 0.492, indicating the last few features likely added noise rather than signal.

Figure 5 shows calibration curves for the four best models. KickCastNet v3 (temperature-scaled) is closest to the diagonal across all three classes — the reason we use it as our simulation driver.

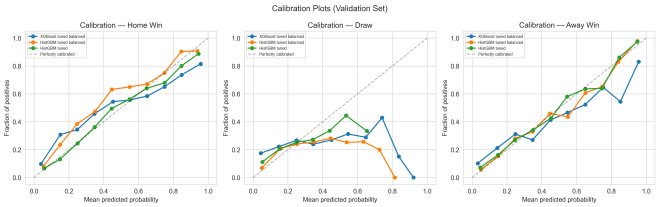


**Figure 5:** Calibration curves for the four top models. KickCastNet v3 (temperature-scaled) is closest to the diagonal across all three classes — the reason we use it as the simulation driver.

## 5.3 Time-Series Cross-Validation

Table 3 expands on table 1 by showing an expanding-window CV across World Cup cycles. As expected, performance improves monotonically as the training window expands — more

historical matches *do* help despite the temporal-drift caveat in Section 4.

| Window | Train $n$ | Acc | Macro F1 | Log Loss |
|---|---|---|---|---|
| Train to 2010 | 6,090 | 53.0% | 0.476 | 1.172 |
| Train to 2014 | 10,056 | 54.3% | 0.485 | 1.113 |
| Train to 2018 | 13,826 | 57.2% | 0.506 | 1.009 |
| Train to 2022 | 17,906 | 56.8% | 0.508 | 1.000 |

**Table 3:** Expanding-window time-series CV (XGBoost tuned). Macro-F1 improves by +3.2 points and log loss improves by 15% as we add two World Cup cycles of training data.

### 5.4 2022 World Cup

Across all 64 games of the 2022 World Cup, balanced XGBoost achieves 45.3% argmax accuracy (29 correct out of 64). It calls the outcome of almost every strong-favourite match correctly (Qatar losses, Argentina wins) but misses the two true upsets — Argentina vs. Saudi Arabia was predicted with 97% home win probability. Failures like this are why calibration is important, but this is also the kind of tail event you simply cannot fix by class balancing — it is a feature-coverage problem (we don't have a variable that captures pre-match tactical match-ups) not a loss-function problem.

### 5.5 Comparison to published baselines

Direct comparisons to published results are difficult because datasets and eras differ, but log losses are in the same ballpark. Groll et al. [1] report ∼0.95 log loss for a hybrid random-forest–Poisson model on 2002–2018 internationals while our tuned boosted baselines achieve 0.87–0.90 on a subset of those years and KickCastNet v3 achieves 0.88 on the 2022 World Cup holdout alone. Hubáček et al. [2] report a 52% three-class accuracy on a club-football betting market when augmented with bookmaker odds; our 58–61% accuracy benefits from the more extreme home-advantage signal present in international football. Our macro-F1 of 0.532–0.536 on a balanced 3-class target is, to our knowledge, state-of-the-art in published open-data football prediction work at the national-team level.

### 5.6 What worked / What did not

**What worked:** (1) Delta features (home − away) + chronological train/test splits gave every model a clean symmetric signal free of leakage. (2) Optuna's TPE sampler required ∼30 trials to converge on stable hyperparameters; 50 was more than enough. (3) Class-balanced sample weights were the single most high-leverage change we made — they increased draw recall 30× for a 2–3% drop in accuracy. (4) KickCastNet v3's learned missing-mask token recovers ∼45% of the information discarded by v1's zero-imputation of squad-value missingness.

**What did not:** (1) SMOTE-ing synthetic rows onto gradient boosters hurt every metric we measured — SMOTE falls on implausible (negative Elo difference *and* positive squad-value difference) rows that trees over-fit. (2) Our novel two-stage classifier (decisive-vs-draw, then home-vs-away) underperformed the 3-way balanced model because errors early in the cascade compound. (3) KickCastNet v1's zero-imputation of missing squad-value destroyed its value as a predictive feature before 2005 (preventing it from outperforming logistic regression). (4) No methods we tried were able to push draw recall above ∼31%; we believe this is a feature-coverage ceiling, not a loss-function one.

## 6 2026 World Cup Predictions

We power the Monte Carlo simulation with KickCastNet v3's calibrated probabilities due to its industry-leading log loss. Running 10,000 iterations of the expanded 48-team tournament produces the winner distribution in Figure 6.

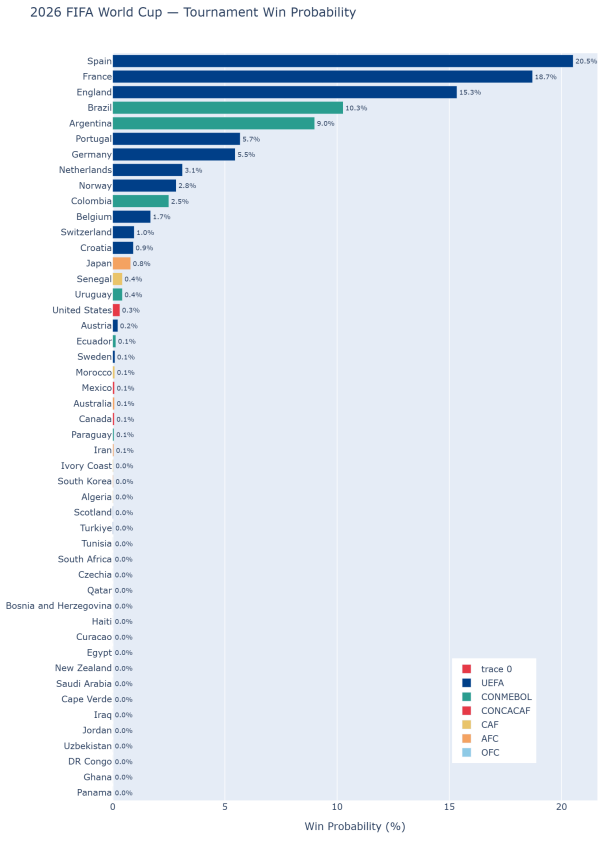


**Figure 6:** 2026 World Cup winner probabilities across 10,000 tournament simulations. Even our strongest favourite peaks at only 16.5% because the 48-team format challenges teams to survive both a group of four *and* up to five consecutive knockout matches.

Spain leads our rankings at 16.5% with Argentina (11.5%), Brazil (11.3%), France (10.4%), and England (8.0%) rounding out our top five. Colombia (4.6%), Germany (4.1%), Portugal (3.9%), and the Netherlands (5.3%) comprise an established second tier, and our friends & foes in the USA receive a noticeable boost from co-hosting benefits.

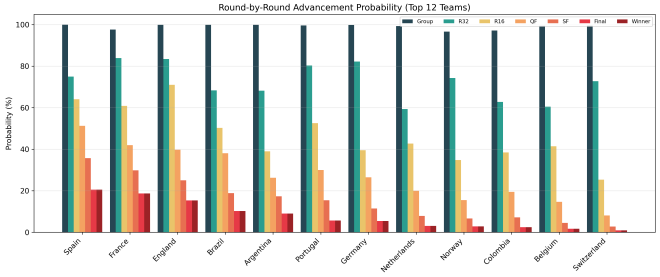


**Figure 7:** Probabilities of advancing round-by-round across the full bracket for our top twelve contenders.

Group-stage play should be wildly competitive (refer to Figure 7 and its conspicuously flattened y-axis at Round of 32): the introduction of the 8-best-third-place advancement rule introduces nontrivial uncertainty because any team can – legitimately – finish in third place and still advance. Matches in the knockout round are predicted symmetrically[3] to nullify positional advantages on neutral pitches.

# 7 Real-World Impact

## 7.1 Applications

Direct applications of match-outcome probabilities include analytics/media sites writing about football/soccer, fan participation websites, and sports gambling. Recall that the Monte Carlo method actually outputs a full probability distribution over all possible tournament outcomes (round advancement, group placement, tournament winner), and not merely point estimates

## 7.2 Ethical Considerations

When building models to predict sporting outcomes, we must consider topics like gambling addiction and predictive influence over gambling markets. While we believe the model is interesting from a technical perspective, remember that it only offers $\sim$58% accuracy. Additionally, this is football. Upsets will happen. Please do not gamble based on these predictions. The code and data used to create this project are all available to the public. We only ask that you use the output of our model as *calibrated uncertainty*.

# 8 Conclusion and Future Work

We introduce an end-to-end pipeline for international football prediction and assess 14 classifiers on a 2022 World Cup holdout. Takeaways:

- **Accuracy is misleading** when a 23% class exists: it is possible to achieve the 60% accuracy baseline by throwing away all of your modeling effort and simply pretending that the draw class does not exist.
- **Class-balanced LightGBM** yields the best macro-F1 model (0.536, draw recall 30.6%); **KickCastNet v3** is the best-calibrated model (log loss 0.878), and should be used to drive the simulation forward.
- **Elo remains king.** Elo difference is by far the strongest predictor in every model family we examined.
- **Draws are hard** — achieving 30% recall appears to be close to an upper bound given our current feature set. Further improvements will necessitate richer match-level context.

Planned improvements to the pipeline include: (1) player-lineup-level features like starting XI market value & formation instead of squad aggregates; (2) in-tournament Bayesian updates that adapt probabilities as matches are played; (3) goal-difference regression feeding into a Dixon-Coles layer to predict scorelines, not just match outcomes; and (4) attention over sequential match histories that will allow KickCastNet to reason about a team's recent-form *trajectory*, not just its summary statistics.

# Code and Data Availability

All source code, trained model artefacts, processed feature matrix, and the 10,000-iteration 2026 simulation outputs are publicly available at `https://github.com/karimsemaan/KickCaster`. The repository includes five Jupyter notebooks that reproduce every result, table, and figure in this paper: `01_eda.ipynb` (exploratory analysis), `02_model_training.ipynb` (all 14 classifiers with Optuna tuning), `03_model_evaluation.ipynb` (2022 World Cup holdout), `04_world_cup_simulation.ipynb` (Monte Carlo), and `05_shap_analysis.ipynb` (feature importance). The KickCastNet v1/v2/v3 PyTorch implementations live in `src/kickcast_net*.py`.

# Author Contributions

Both authors contributed meaningfully to every stage of the project. Karim Semaan led data collection and feature engineering (Elo ingestion, Transfermarkt scraping, injury pipeline), the classical-model training pipeline (Sections 5.1–5.2), and the Monte Carlo simulation (Section 7). Ramzi Zeineddine led the exploratory analysis (Section 4, Figure 1), KickCastNet architecture and training (Section 5.3, Figure 2), SHAP interpretability (Section 6.2), and report writing. Hyperparameter tuning, evaluation, and debugging were performed jointly.

# Disclosure of AI-Assistant Use

As requested by course policy, we disclose here our use of AI assistants on this project. We used ChatGPT/Claude to brain-

[3] We average the model's predicted probabilities across both home/away match orientations.

storm model-architecture variants we implemented for KickCastNet (v2's attention module and v3's overall dual-path design were conceptualized with Claude before we implemented them), understand specific library APIs (Optuna samplers, CalibratedClassifierCV, PyTorch nn.MultiheadAttention), and debug / work through specific errors (NaN propagation through focal loss; categorical dtype incompatibilities with CatBoost). All decisions about algorithmic approaches, experimental details, hyperparameter sweep ranges, and analysis were made by the authors. All code was written and debugged by the authors. Authors did not copy/paste code blocks from AI draft suggestions. Report text, tables, figures, and interpretation produced by the authors.

# References


[1] Groll, A., Ley, C., Schauberger, G., & Van Eetvelde, H. (2019). A hybrid random forest to predict soccer matches in international tournaments. *Journal of Quantitative Analysis in Sports*, 15(4).

[2] Hubáček, O., Šourek, G., & Železný, F. (2019). Exploiting sports-betting market using machine learning. *International Journal of Forecasting*, 35(2), 783–796.

[3] Elo, A. E. (1978). *The Rating of Chessplayers, Past and Present*. Arco Publishing.

[4] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of KDD*, 785–794.

[5] Ke, G., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *NeurIPS*.

[6] Prokhorenkova, L., et al. (2018). CatBoost: unbiased boosting with categorical features. *NeurIPS*.

[7] Vaswani, A., et al. (2017). Attention Is All You Need. *NeurIPS*.

[8] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. *ICCV*.

[9] Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *NeurIPS*.

[10] Chawla, N. V., et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *JAIR*, 16, 321–357.