Fifa World Cup 2026 Winner Prediction

The Core Problem & My Approach

The Problem:

- Predictingfootball matches is notoriously difficult. Simple stats like "win rate" don't capture the whole story.
- How a team plays (their style) and their specific strengths/weaknesses against an opponent are just as important.

MyTwo-Phase Approach:

Phase1: Understand (Unsupervised Learning) First, can I use AI to discover if distinct "team play styles" even exist in the data?

Phase 2: Predict (Supervised Learning) Second, once I've established that team identity matters, can I build a model to predict match outcomes?

Phase 1: Understanding "Team DNA" with K-Means

Objective

Todiscovernatural groupings of teams based on their advanced performance statistics.

Model

K-Means Clustering (an unsupervised model, meaning I don't tell it the answers).

Dataset: final_team_data.csv

This dataset contained 12 key performance metrics for each team. I focused on variables like:

- possession
- xg (Expected Goals)
- sca_per90 (Shot-Creating Actions)
- progressive_passes
- gk_save_pct (Goalkeeper Save %)
- aerials_won_pct

Process:

1.**Scale Data:** StandardScaler so that all 12 features were treated equally by the model. 2.**Find 'k':** Using the "Elbow Method," I determined that k=3 was the optimal number of clusters. 3.**Run Model:** The K-Means algorithm then grouped every team into one of these three clusters based on their 12- dimensional "performance DNA."

Phase 1 Results: The Clusters

The K-Means model successfully found three distinct groups of teams. This confirmed my hypothesis: teams do have different, measurable "play styles." For example, one cluster might be "High-Possession, High-Attack" while another is "Defensive Counter-Attack." This discovery was the green light to move on to Phase 2, armed with the knowledge that team identity is a critical predictive factor.

Phase 2: Building the Predictor (Random Forest)

Objective

Topredict the specific outcome of a match (Home Win, Away Win, or Draw).

Model

Random Forest Classifier (a supervised model).

Why Random Forest?

- It's an ensemble of hundreds of individual Decision Trees.
- This "voting" system makes it highly accurate and prevents "overfitting," which is a common problem with single decision trees.

Dataset: fifaworldcup_with_win_rates.csv

This dataset is different; it contains historical match-by-match results.

Phase 2: The "Secret Sauce" – Feature Engineering

Key Features Used:

01

Win_Rate_Difference

(Home TeamWinRate - Away Team Win Rate). This is a simple but powerful measure of relative historical strength.

02

Is_Home_Advantage

Asimple binary (1or 0)flag to capture the effect of a team playing in their host country.

03

Knockout Stage

Abinaryflag, asthe pressure and tactics in a knockout match are different from a group stage.

04

Home Team Name & Away Team Name

This is the mostcritical feature. By **One-HotEncoding** the team names, I'm not just telling the model "Team A" vs.

"Team B." I'm allowing it to learn the specific, unique identity of "Brazil," "Germany," "Argentina," etc., and how they perform. This directly connects back to my K-Means findings.

Making the Model "Production-Ready" (The Job Files)

rf_classifier_v3.joblib

This is the **"Brain."** It's the fully trained Random Forest model (all 100 trees).

preprocessor_v3.joblib

This is the **"Recipe."** It's a ColumnTransformer that knows exactly how to convert new, raw input (like "Brazil" vs. "Germany") into the specific numeric format the "Brain" was trained on. This is essential.

label_encoder_v3.joblib

This is the **"Decoder Ring."** The model predicts numbers (0, 1, 2), and this file converts them back into human-readable text ("Away Win," "Draw," "Home Win").

team_win_rate_lookup.json

This is a **"DataHelper."** A simple dictionary to quickly look up any team's win rate to calculate the Win_Rate_Difference feature.

The Final Product: Final_v6.py Application

Key Features:

Data Visualisation Tab	Live Prediction & Odds	Full Tournament Simulation
Connects back to Phase 1.	The core of Phase 2 in action.	The application's "main event."

App Feature: Data Visualisation

This tab directly loads the final_team_data.csv (the K-Means dataset) and plots Expected Goals vs. Possession.
The purpose of this is to show the user the 'why'4to visually reinforce the concept from my K-Means analysis that teams have different styles. It builds trust in the model by showing the underlying data.

App Feature: Live Prediction & Match Odds

Here's what happens in the background when you click 'Predict':

1.The app gets the two team names (e.g., 'Argentina', 'France').

2.It uses the team_win_rate_lookup.json to calculate the Win_Rate_Difference.

3.It bundles this with the other features (like 'Knockout Stage') into an array.

4. It feeds this array into the preprocessor_v3.joblib (the "Recipe").

5.It takes the numeric output and feeds it into the rf_classifier_v3.joblib (the "Brain").

6.The model "votes" and outputs probabilities (e.g., [0.25, 0.35, 0.40]).

7. The app uses the label_encoder_v3.joblib (the "Decoder") to display these as:

- **Home Win: 40%**
- **Draw: 35%**
- **Away Win: 25%**

App Feature: Full Tournament Simulation

Here, the user selects their 16 teams, and the application programmatically simulates the entire tournament.
For every single match, from the Round of 16 to the Final, it runs the full prediction pipeline I just described. It determines the winner (based on the highest probability) and automatically advances them to the next round.
The 'Match Odds' tab then displays the predicted probabilities for every match that was simulated, and the 'Bracket' tab updates with the winners of each round, ultimately crowning a champion.

Conclusion & Key Takeaways

1. Phase 1 (K-Means)proved that teams have distinct, cluster-able "play styles."
2. Phase 2 (Random Forest) built a robust predictor by using Feature Engineering (especially One-Hot Encoded team names) to capture this team identity.
3. The "Job Files" created a reusable, production-ready pipeline.
4. The Final_v6.py Application successfully packaged this complex, two-model AI system into an interactive, functional, and user-friendly tool.