AI Club DC Hackathon
REPORT

Members: Shloka Tomar, Nikitha S

Our Journey:

- Dropped 'Home_Score_AET', 'Away_Score_AET', 'Home_Penalties', 'Away_Penalties', because almost all the values in those features were null/NaN.

Created new features:

- Goal_diff and points_diff representing the difference in goals and points of the teams respectively.

- Created a column ‘advance’ which would predict which team moves forward based on the goal difference being greater than or equal to zero.

- Made features goals_for_5, goals_against_5, points_5, which represented the mean of the last 5 goals scores by the team and its opponent, and added .shift(1) to prevent it from including the current match to avoid data leakage.

- Made a feature ‘consistency’ that represented the standard deviation of the last 10 goal difference of the matches of each team (with .shift(1) included obviously) to represent the reliability/consistency of each team.

- Made features season_goal_diff and season_points_diff to capture how a team did every season.

- Made h2h_win_rate and h2h_goal_diff to represent the head to head record of 2 teams going against each other.

- Made a feature matches_played which contained the number of matches played by each time, representing their experience in a way.

- Then we dropped columns causing data leakage, i.e, the columns that were directly telling us the result of the match.
- ('Team_Score','Opponent_Score','goal_diff','advance', 'Team_Points', 'Opponent_Points','points_diff', 'Date', 'Time')

What we tried :)

1. The first model tried was XGBoost. The individual accuracy for predicting which of the 2 teams would win varied from 65-67 depending on the parameters we gave, but the end kaggle score always ended up being 34.375.

2. Next we tried LightGBM. The kaggle score ended up being the same.

3. Tried scraping xG parameter from fbref, but only after scraping we realized that the data was past the training cutoff. So then we tried making a proxy xG feature which was like xG but artificially created to train the model on. But this failed pretty badly and gave a score of around 17.

4. Tried elo ratings - failed because it made an extreme bias towards stronger teams and ended up always favouring them. We found this elo dataset on kaggle which had elo ratings of all the teams. We incorporated it and it had the same problem as elo ratings; and extreme bias towards stronger teams, so we ended up discarding it.

5. We included fifa world cup data because cwc winners have proven they can beat the best teams from every continent. We scraped that data from https://rsssf.org/tablesf/fifawcc.html, and used that to calculate the "prestige" of the team.

6. Something chatgpt suggested was using weighted model predictions based on the round we were predicting - we tried this too, favouring team "strength" over the model in later rounds, but this negatively impacted our score so again, we ditched it.

Our final model

Ensemble of Xgboost, lightgbm, with hyperparameter tuning (using optuna). It worked on a dataset made by combining full_dataset.csv, train.json, and the fifa worldcup data.

Classification ensemble predicting Champions League knockout match winners using XGBoost and LightGBM with equal weighting. (left out catboost because it was pretty slow and made almost no difference)

Data

Source: 106,876 historical football matches (full_dataset.csv, train.json)

Data cleaning summary:

Started with 106,876 matches, ended with 104,412 (97.7% retention, removed 2,464 rows).

What Was Cleaned:

Dropped 4 columns: Home_Score_AET, Away_Score_AET, Home_Penalties, Away_Penalties. These had 97-98% missing values (only exist for matches going to extra time/penalties) and weren't predictive for regular match outcomes.

Removed missing scores: 2,464 rows with null values in Team_Score, Opponent_Score, Team_Points, or Opponent_Points. Can't train a prediction model without knowing match outcomes.

Parsed dates: Converted date strings (format '31/08/2002') to datetime objects. Required for temporal train/test split and rolling window calculations. Dropped rows with unparseable dates.

Removed invalid scores: Filtered out negative scores (data entry errors) and scores ≥30 (unrealistic, likely data corruption). Real football scores fall in 0-20 range.

Removed duplicates: Dropped duplicate matches based on Date+Team+Opponent+Competition. Prevents same match from being counted multiple times which would bias statistics.

Cleaned train.json: Standardized inconsistent round names (round_16→round_of_16, quarterfinals→quarter_finals, semifinals→semi_finals) and removed whitespace from season keys. Ensures bracket progression logic works correctly.

Result:

High-quality dataset with no missing values in critical columns, valid score ranges, proper datetime objects, no duplicates, and standardized categorical values. Ready for feature engineering with minimal data loss (only 2.3% removed, all genuinely problematic records).

Features (27 total)

goals_for_5 - Average goals scored in last 5 matches
goals_against_5 - Average goals conceded in last 5 matches
points_5 - Average points earned in last 5 matches
win_rate_5 - Win percentage in last 5 matches
goal_diff_5 - Goal difference in last 5 matches
These capture momentum and current team state.

season_goal_diff - Average goal difference across the current season
season_points_avg - Average points per match this season
These capture overall season performance.

h2h_win_rate - Historical win rate against this specific opponent
h2h_goal_diff - Historical goal difference against this specific opponent
Past matchups reveal psychological edges, tactical mismatches, or playing style advantages.

matches_played - Total career matches
cl_matches - Champions League specific matches
These gauge team experience.

goals_std_5 - Standard deviation of goals scored (last 5 matches)
consistency_score - Normalized consistency metric 1/(1 + std)
Measures how consistent the team is.

knockout_matches - Total knockout matches played
semifinals_reached - Number of times reached semifinals
finals_reached - Number of times reached finals
big_game_exp - Combined score (finals × 3 + semis)

team_cwc_prestige - Team's CWC prestige score
opp_cwc_prestige - Opponent's CWC prestige score
cwc_prestige_diff - Difference between the two
Cwc winners have proven that they win under maximum pressure.
Basic categorical features:
Round - Which round (R16, QF, SF, Final)
Team - Team identity
Opponent - Opponent identity
Location - Home/Away/Neutral
Country - Where match is played
Competition - CL vs domestic league

All features use .shift(1) or expanding windows to prevent data leakage.

Models

XGBoost: Gradient boosting with categorical support (69.76% accuracy)
LightGBM: Leaf-wise tree growth (69.63% accuracy)
Ensemble: Arithmetic mean of probabilities (69.85% accuracy)

Optimization

Method: Optuna

Trials: 20 per model
Search space: learning_rate [0.01-0.3], max_depth [3-10], n_estimators [100-600], subsample [0.6-1.0], regularization parameters

Prediction Flow

1. Extract 27 features for matchup
2. Get probability predictions from both models
3. Average: p_ensemble = 0.5×p_xgb + 0.5×p_lgb
4. Predict teamA if p_ensemble > 0.5, else teamB
5. Follow bracket progression (R16→QF→SF→Final)

Performance

69.85% validation accuracy.

(P.S: Our initial submission had a score of 34.375, and that was also the score of our final submission. We tried a LOT of stuff in between that actually reduced our score, but found it impossible to surpass that score without some form of data leakage.)