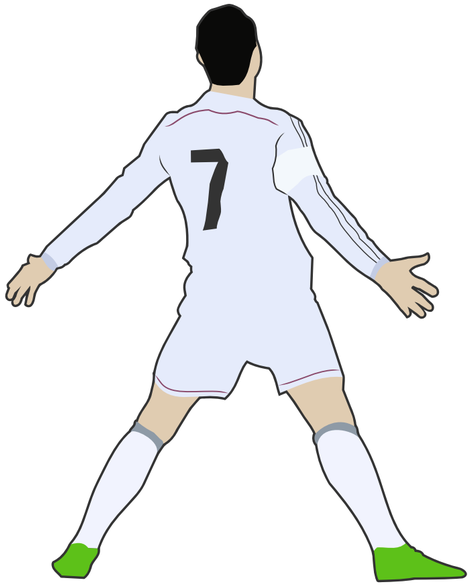
HOW CERTAIN OBJECTIVE FACTORS AFFECT PREMIER LEAGUE MATCH OUTCOME (2015-2025)

DATA1002 - Group 1

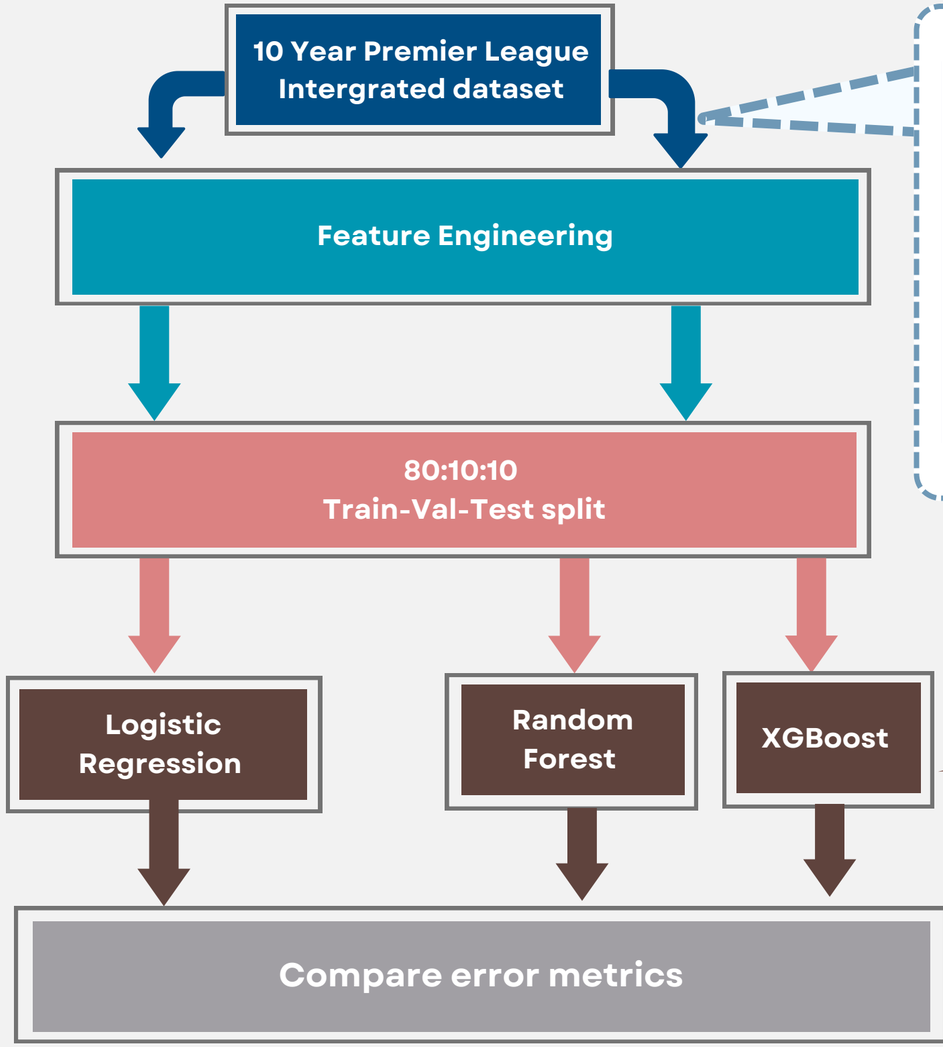
Presented by: Kai Truong , Charlie Tran , Binh Minh Tran

RESEARCH QUESTION & DATASET SUMMARY

How accurately can pre-match factors (Elo rating, xG difference, venue, odds) predict Premier League match results?



EXPERIMENTAL WORKFLOW



time_id	HomeTeam	Home_ELO	AwayTeam	Away_ELO	ELO_Difference	ELO_Predicted_Winner	xG_Predicted_Winner	HomeWin
8/8/2015	Bournemouth	1594.6	Aston Villa	1577.9	16.7	H	H	0
8/8/2015	Chelsea	1890.96	Swansea City	1670.61	220.35	H	A	0
8/8/2015	Everton	1706.56	Watford	1574.09	132.47	H	H	1
8/8/2015	Leicester City	1633.62	Sunderland	1605.13	28.49	H	H	1

Predict

Accuracy, F1 Score,
Precision, Recall

Context

Dataset:

PL_integrated_dataset_10years (Integrated 2015–2025)

- 3,774 matches, 63 variables (from Football-Data, Understat, ClubElo).

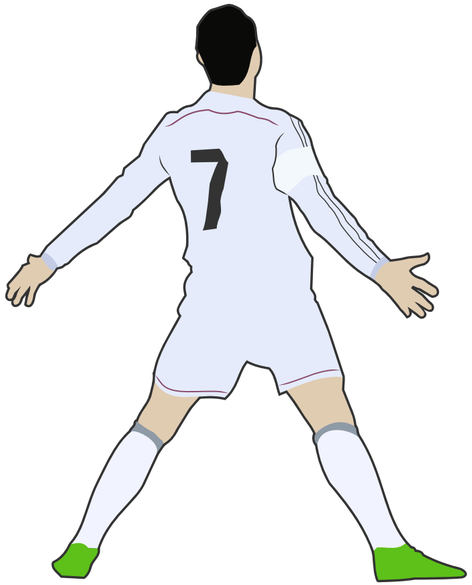
Target/Response Variable: HomeWin

Features used for modelling:

- ELO_Difference
- xG_Difference
- HomeTeam (venue)
- Odds_H, Odds_A, Odds_D
- win_pct_diff

DATA PREPARATION & ANALYSIS

A complete walk-through from cleaning, integration to the analysis of the dataset



Data Summary

SEASON SUMMARY

Season	Rows	Columns	Date_range	Missing_Values	Unique_Home_Teams	Unique_Away_Teams
2015-2016	380	65	01/03/2016 to 31/10/2015	3	20	20
2016-2017	380	65	01/01/2017 to 31/12/2016	0	20	20
2017-2018	380	65	01/01/2018 to 31/12/2017	0	20	20
2018-2019	380	62	01/01/2019 to 31/03/2019	0	20	20
2019-2020	380	106	01/01/2020 to 31/08/2019	0	20	20
2020-2021	380	106	01/01/2021 to 31/10/2020	0	20	20
2021-2022	380	106	01/01/2022 to 31/10/2021	13	20	20
2022-2023	380	106	01/01/2023 to 31/12/2022	4	20	20
2023-2024	380	106	01/01/2024 to 31/12/2023	1166	20	20
2024-2025	380	120	01/01/2025 to 31/08/2024	1440	20	20

A total of **3800** rows and **153** unique columns

Data Cleaning and Standardization

1. Cleans raw Premier League match data

2. Handles missing values and inconsistencies

3. Standardizes team names and date formats

Cleaning 2015-2016...

✓ Date column converted (DD/MM/YYYY format)

✓ Team names standardized

✓ Season column added

✓ Missing values: 3 → 3

✓ Data types standardized

Cleaning 2016-2017...

✓ Date column converted (flexible parsing)

✓ Team names standardized

✓ Season column added

✓ Data types standardized

Cleaning 2017-2018...

✓ Date column converted (DD/MM/YYYY format)

✓ Team names standardized

✓ Season column added

✓ Data types standardized

Data Integration and Merging

1. Merges match results with ELO ratings and xG data using **common** columns

```
Common columns across all seasons (39):
['AC', 'AF', 'AR', 'AS', 'AST', 'AY', 'AwayTeam', 'B365A', 'B365D', 'B365H', 'BWA', 'BWD', 'BWH', 'Date', 'Div', 'FTAG', 'FTHG', 'FTR', 'HC', 'HF', 'HR',
'HS', 'HST', 'HTAG', 'HTHG', 'HTR', 'HY', 'HomeTeam', 'PSA', 'PSCA', 'PSCD', 'PSCH', 'PSD', 'PSH', 'Referee', 'Season', 'WHA', 'WHD', 'WHH']
```

2. Creates engineered features for analysis

- **xG_difference** → captures shot quality
- **ELO_Difference** → captures strength
- **win_pct_diff (5-match trailing)** → reflects short-term form

Data Integration and Merging

3. Final data check-up and enhancement

Final dataset shape: (3774, 63)

Date range: 2015-08-08 to 2025-05-25

Total columns: 63

4. Integrates multiple data sources into unified dataset

PL_integrated_dataset_10years_

SAMPLE OF INTEGRATED DATASET

Date	Season	HomeTeam	AwayTeam	FTHG	FTAG
2015-08-08	2015 - 2016	Bournemouth	Aston Villa	0	1
2015-08-08	2015 - 2016	Chelsea	Swansea City	2	2
2015-08-08	2015 - 2016	Everton	Watford	2	2
2015-08-08	2015 - 2016	Leicester City	Sunderland	4	2
2015-08-08	2015 - 2016	Manchester United	Tottenham	1	0

Home_xG	Away_xG	Home_ELO	Away_ELO
0.876106	0.782253	1594.601929	1577.901123
0.643960	2.592030	1890.962891	1670.608276
0.604226	0.557892	1706.561157	1547.087280
2.568030	1.459460	1633.624390	1605.132446
0.627539	0.674600	1810.338257	1730.680176

Data Analysis

KEY INSIGHTS FROM BINARY ANALYSIS

- Home Advantage: Home teams win 57.9% of non-draw matches
- Correlation analysis shows predictive methods have moderate positive correlation with actual outcomes
- Confidence intervals indicate statistical significance of prediction accuracies
- All methods show bias towards home wins in binary predictions

Logistic Regression

Goal of this model

Predict whether the home team wins a Premier League match. The classification target is binary:

- HomeWin = 1 if HomePoints == 3, else 0

Feature Engineering

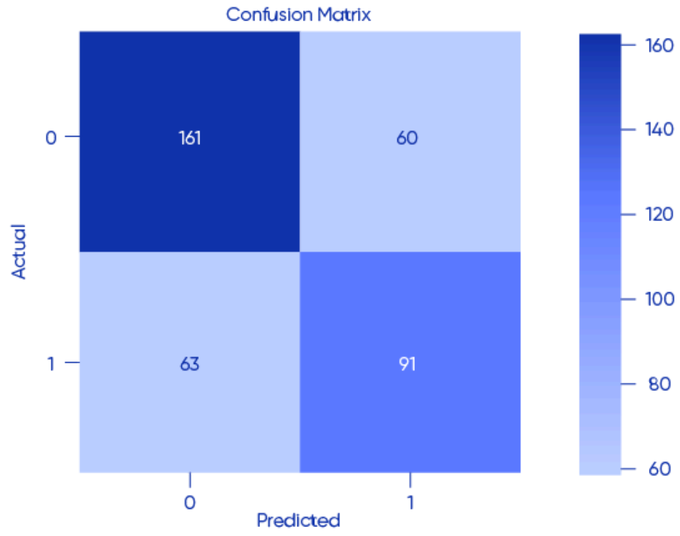
- **xG_difference** = avg xG (home, last 5) − avg xG (away, last 5)
- **ELO_Difference** = Home_ELO − Away_ELO
- **win_pct_diff** = home win% (last 5) − away win% (last 5)

```
        df.at[idx, 'away_team_avg_xG_last5'] = np.mean(last5_xg[away])

    if len(last5_results[home]) == 5:
        df.at[idx, 'home_team_win_rate_last5'] = sum(last5_results[home]) / 5 * 100
    if len(last5_results[away]) == 5:
        df.at[idx, 'away_team_win_rate_last5'] = sum(last5_results[away]) / 5 * 100

    # Update deques with current match stats
    last5_xg[home].append(row['Home_xG'])
    last5_xg[away].append(row['Away_xG'])

    last5_results[home].append(1 if row['HomePoints'] == 3 else 0)
    last5_results[away].append(1 if row['AwayPoints'] == 3 else 0)

# Feature 1: xG difference
df['xG_difference'] = df['home_team_avg_xG_last5'] - df['away_team_avg_xG_last5']

# Feature 2: win% difference
df['win_pct_diff'] = df['home_team_win_rate_last5'] - df['away_team_win_rate_last5']

# Feature 3: Elo rating difference
```

Split - 8/1/1 principle

- **Train** on the **8** past seasons (2015-2023)
- **Validate** on **1** season (2023-2024)
- **Test** on the **most recent** one (2024-2025)

```
# 8/1/1 split based on dates
train_df = df[df['Date'] < '2023-07-01']
val_df   = df[(df['Date'] >= '2023-07-01') & (df['Date'] < '2024-07-01')]
test_df  = df[df['Date'] >= '2024-07-01']

print(f"Train: {len(train_df)}, Validation: {len(val_df)}, Test: {len(test_df)}")
```



Feature	Coefficient
xG_difference	0.211
win_pct_diff	-0.096
ELO_Difference	0.773
Intercept	-0.234

Pr(HomeWin=1)= −0.234 + 0.211*xG_difference − 0.096*win_pct_diff + 0.773*ELO_Difference

Logistic Regression

Why Logistic Regression for this task?

- Easy to interprete
- Logistic regression outputs P(Home Win), so we can tune the threshold for precision/recall trade-offs.
- Regularized & robust
- This model is suitable as a baseline or interpretable benchmark for football match prediction.

Random Forest

Goal of this model

- Predict whether the home team will win (HomeWin = 1 vs 0).
- Use only information available before kickoff – NOT anything that happens after the match.

Feature Engineering step:

1. Rolling form features (last 5 matches)
We generate team form features so the model understands momentum:
For each team, before each match:

- points_last5: average points per game in last 5 matches
- goals_for_last5, goals_against_last5
- goal_diff_last5
- xG_last5: average expected goals created
- shot_accuracy_last5
- win_rate_last5 = points_last5 / 3

Then we attach those to matches as home vs away:

- **Home_points_last5, Away_points_last5, …**
- **Home_win_rate_last5, Away_win_rate_last5, etc.**

2. Other predictive features we feed into the model
From the merged Premier League dataset (3,774 matches, 2015–2025):

- **Team strength:**
  - Home_ELO, Away_ELO, ELO_Difference
- **Recent performance trend features:**
  - Home_points_last5, Away_points_last5
  - Home_goal_diff_last5, Away_goal_diff_last5
  - Home_xG_last5, Away_xG_last5
  - Home_shot_accuracy_last5, Away_shot_accuracy_last5
  - Home_win_rate_last5, Away_win_rate_last5
- **Market expectation:**
  - B365H, B365A (betting odds snapshot before the match)
- **Context:**
  - MatchWeek (early in the season vs later in the season)

```
candidate_features = [
    'ELO_Difference', 'Home_ELO', 'Away_ELO',
    'Home_points_last5', 'Away_points_last5',
    'Home_goals_for_last5', 'Away_goals_for_last5',
    'Home_goals_against_last5', 'Away_goals_against_last5',
    'Home_goal_diff_last5', 'Away_goal_diff_last5',
    'Home_xG_last5', 'Away_xG_last5',
    'Home_shot_accuracy_last5', 'Away_shot_accuracy_last5',
    'Home_win_rate_last5', 'Away_win_rate_last5',
    'B365H', 'B365A', 'MatchWeek'
]
```

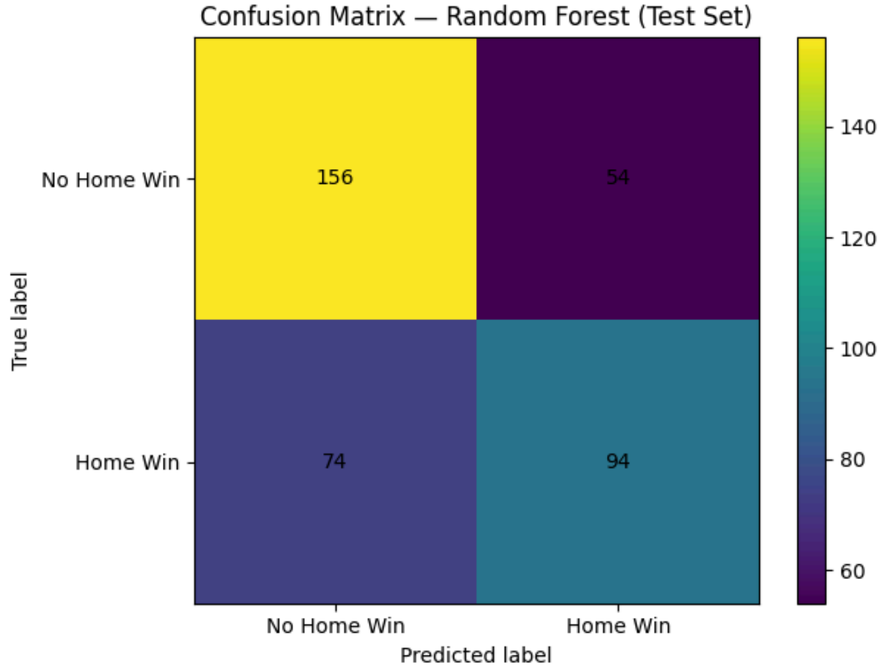
Why this matters?

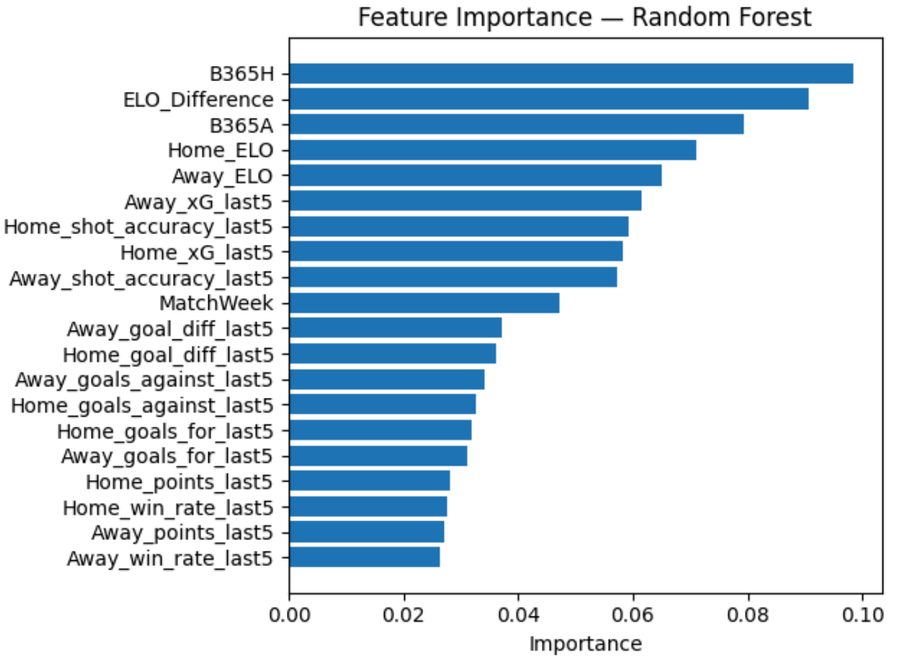
A team's true strength is not just season-long ability (Elo) – it's also recent form.
This lets the model learn patterns like "home team in hot form vs away team in a slump."

Random Forest

Why Random Forest for this task?

- Handles non-linear interactions (e.g. “high Elo + terrible recent form” is different from “high Elo + good recent form”).
- Naturally handles different feature scales (we don’t need standardisation).
- Can rank which features matter the most (feature importance).
- class_weight='balanced' helps when home wins and non-home-wins aren’t perfectly 50/50.





- **Top feature(s)** usually include `**ELO_Difference**` and `**xG_last5**`, indicating that relative team strength and expected performance correlate strongly with home-win outcomes.
- **Betting odds**, when included, often add incremental signal (markets embed a lot of prior information).

XGBoost

Model Overview:

- XGBoost is a regularized gradient-boosting method that builds many small decision trees sequentially to correct errors and minimize loss, delivering strong performance on tabular data.
- Loss function: Cross-Entropy Loss

Task & setup

- Binary target: Home win vs Away win (draws removed).
- Chronological split (8/1/1 seasons): train on past 8, validate on 1, test on most recent.

Features:

- Elo & diffs,
- Form (last-5), momentum (3–5), H2H, seasonality (month/phase + sin/cos), Betting odds + 2-class odds/logit, interactions.

Tuning & calibration:

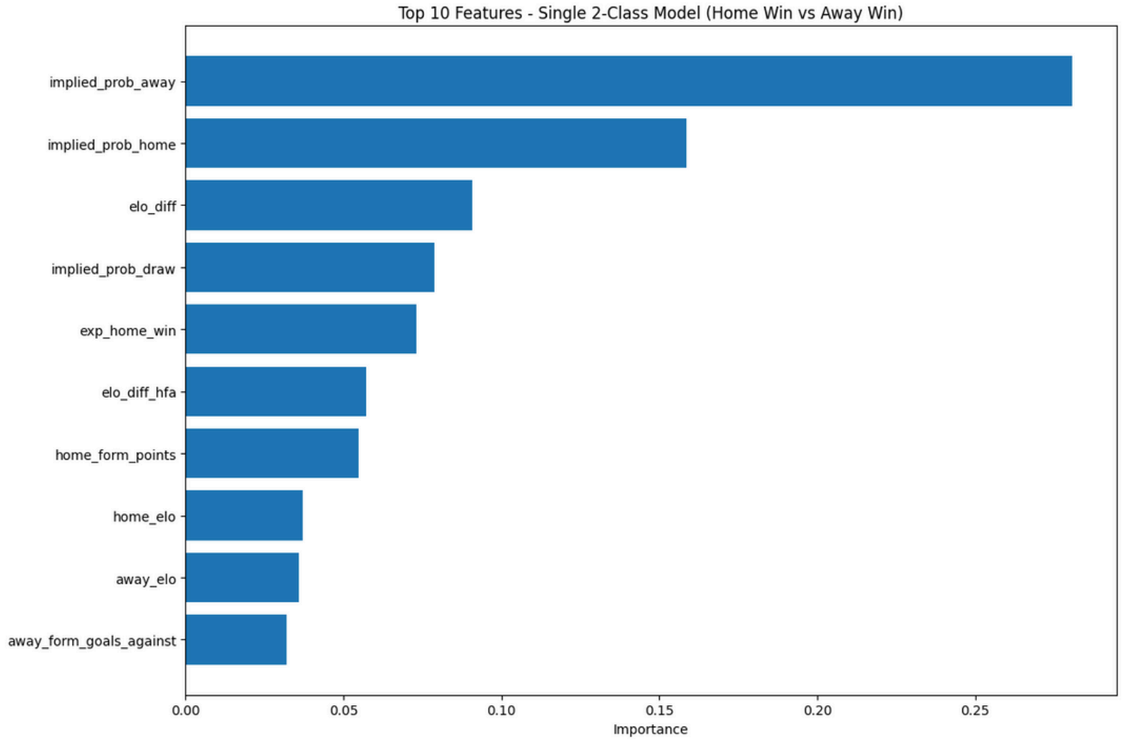
- Optuna search (depth, lr, trees, subsample, reg)
- Early stopping on validation,
- Platt (sigmoid) calibration for well-calibrated probabilities.

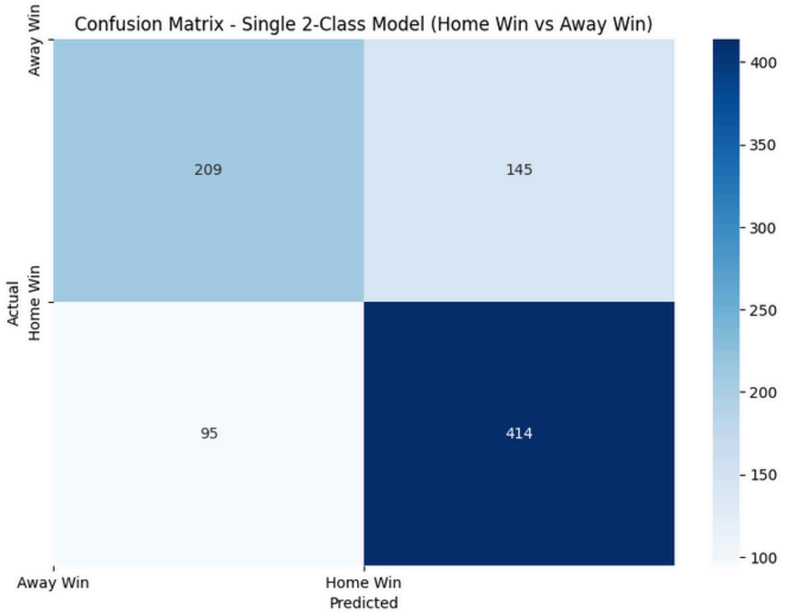
```python
def train_models(self, X_train, y_train, X_val=None, y_val=None, tune_hyperparams=True):
    """Train calibrated XGBoost (binary:logistic) with optional tuning."""
    Xtr = self.scaler.fit_transform(X_train)
    Xva = self.scaler.transform(X_val) if X_val is not None else None
    params = self.tune_hyperparameters(X_train, y_train) if tune_hyperparams else (self.best_params or get_default_params()

    Xf, yf, Xcal, ycal = self._create_calibration_set(Xtr, y_train)
    raw = xgb.XGBClassifier(objective='binary:logistic', eval_metric='logloss',
                            random_state=self.random_state, early_stopping_rounds=20,
                            verbosity=0, **params)

    if Xva is not None:
        raw.fit(Xf, yf, eval_set=[(Xva, y_val)], verbose=False)
    else:
        raw.fit(Xf, yf, verbose=False)

    self.model = CalibratedClassifierCV(raw, method='sigmoid', cv='prefit').fit(Xcal, ycal)
    self._evaluate_calibration(Xcal, ycal)

    self.best_params = params
    self.threshold_ = params.get('threshold', 0.5)
    self.hp_manager.save_threshold(self.threshold_)
    return self
```

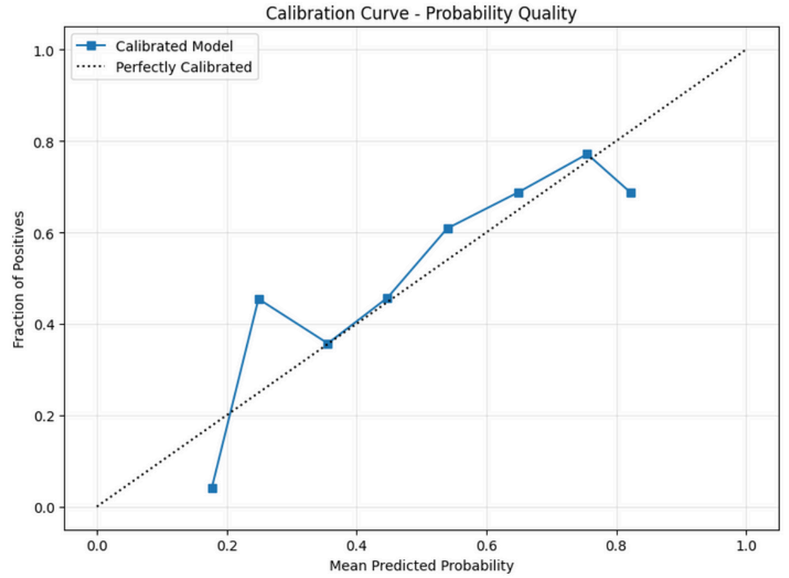
XGBoost

Model choice justification:

- Captures non-linear interactions (Elo × form × odds) that LR misses.
- Boosting often edges bagging (RF) on tabular, engineered features when tuned.
- Regularization + early stop tame overfit under season-wise drift.
- Calibrated probabilities useful for decision support (thresholding, risk).



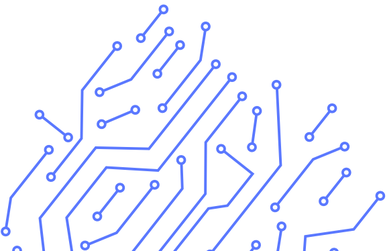




Model Evaluation & Comparison

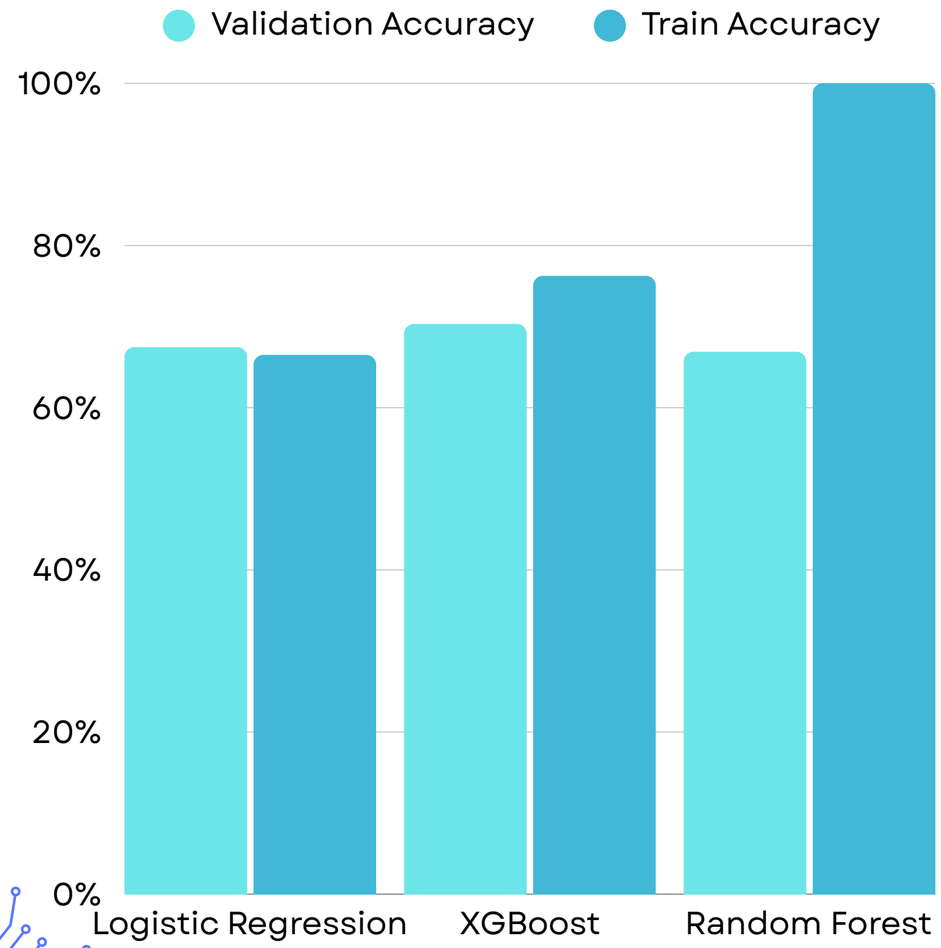
Performance Metric	Equation
Precision	$\frac{TP}{TP + FP}$
Recall	$\frac{TP}{TP + FN}$
Accuracy	$\frac{TP + TN}{TP + FN + TN + FP}$
F1-score	$\frac{2.(precision.recall)}{precision + recall}$

```python
def metrics(y_true, y_pred, y_proba=None):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    acc  = (tp+tn)/(tp+tn+fp+fn)
    prec = tp/(tp+fp) if (tp+fp) else 0.0
    rec  = tp/(tp+fn) if (tp+fn) else 0.0
    spec = tn/(tn+fp) if (tn+fp) else 0.0
    f1   = 2*prec*rec/(prec+rec) if (prec+rec) else 0.0
    out = {"TN":tn,"FP":fp,"FN":fn,"TP":tp,
           "accuracy":acc,"precision":prec,"recall":rec,"specificity":spec,"f1":f1}
    if y_proba is not None: out["auc"] = roc_auc_score(y_true, y_proba)
    return out
```

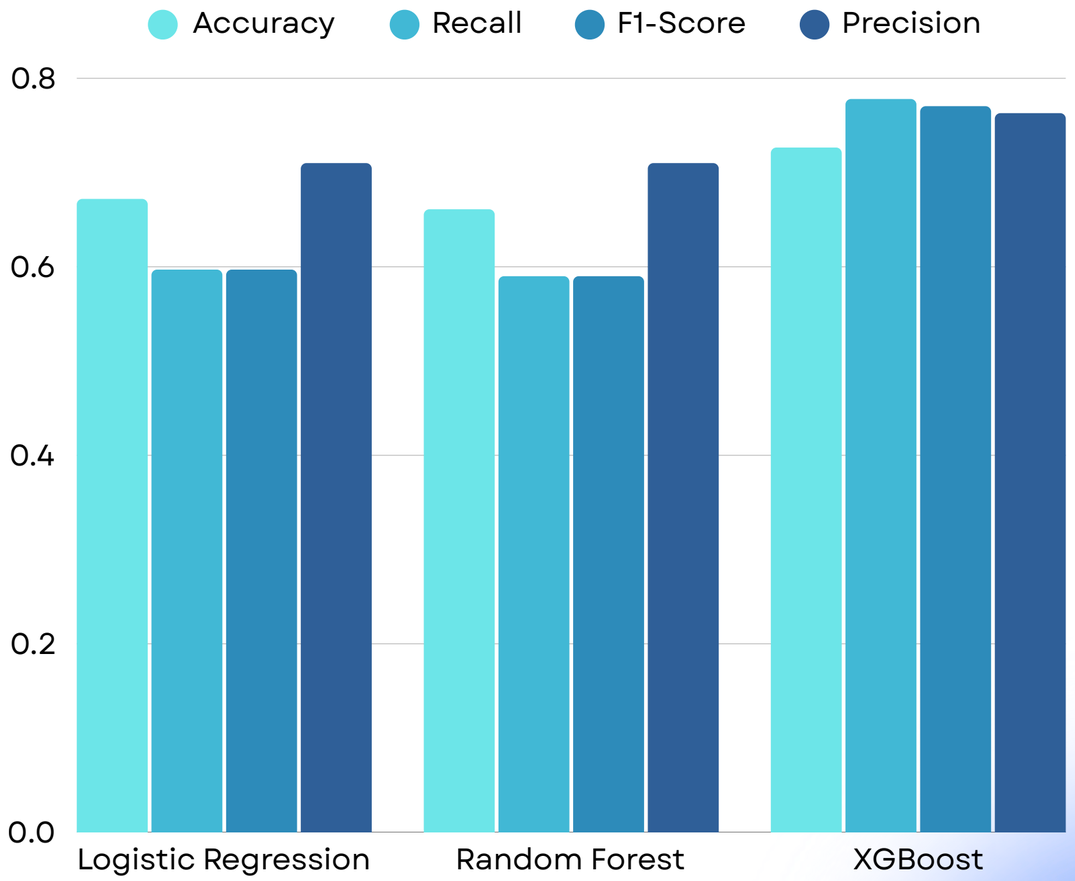


Model Evaluation & Comparison

Result on Training - Validation set



Result on Test set



Model Evaluation & Comparison

Training–validation set:

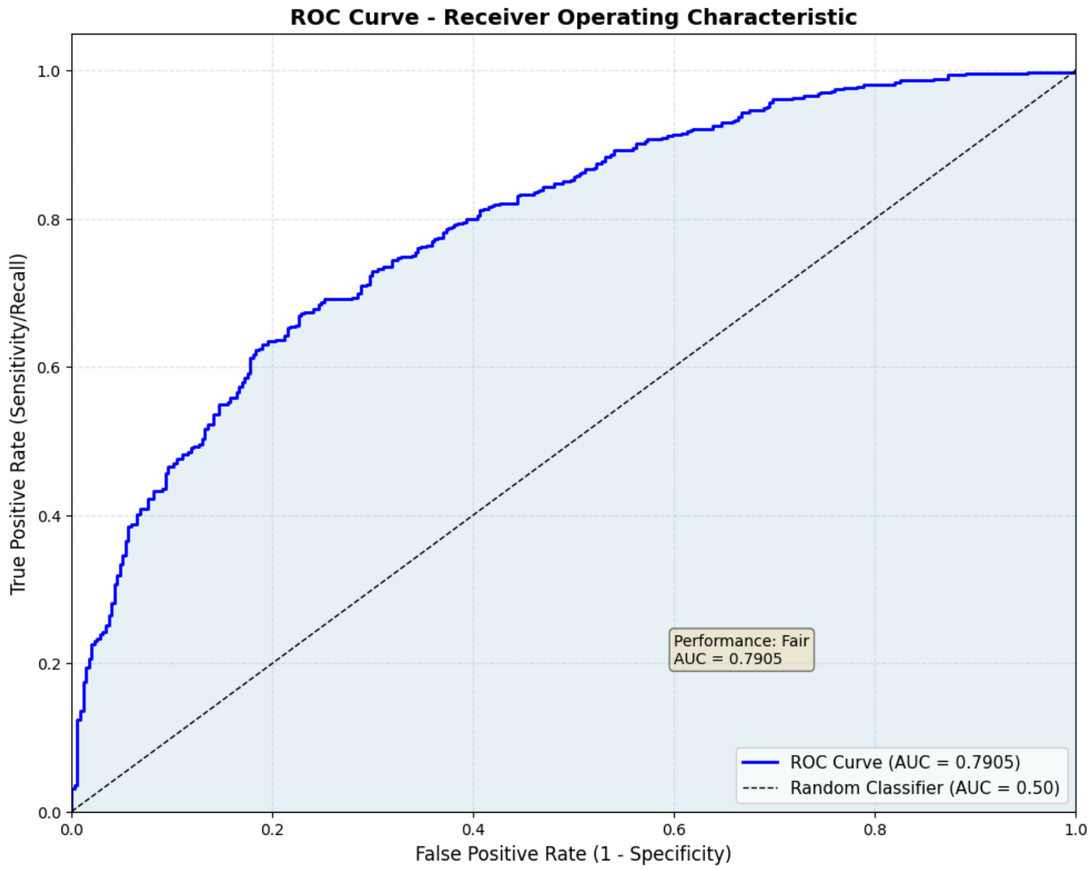
- Logistic Regression: Train ≈ Val (~67%) → good generalisation, low variance; capacity-limited.
- XGBoost: Train ~75% vs Val ~70% → small gap ⇒ mild overfit but acceptable.
- Random Forest: Train ~100% vs Val ~67% → severe overfit (deep trees memorising).

Test set:

- XGBoost leads on Accuracy ≈0.73, Recall ≈0.78, F1 ≈0.77, Precision ≈0.76 → best overall.
- Logistic Regression ~0.67 Acc, ~0.60 Recall/F1, ~0.70 Precision → stable baseline, misses positives vs XGB.
- Random Forest ~0.66–0.67 Acc, low Recall/F1 (~0.59), Precision ~0.70 → overfit hurts generalisation.

Takeaways:

- Choose XGBoost as primary model.
- Keep LR as interpretable baseline.
- RF needs stronger regularisation (limit depth/trees, increase min_samples_leaf, use oob/early stopping) or drop.



Conclusion & Reflection

Summary of findings:

- Elo + xG were the best predictors

Limitations:

- Key drivers: betting odds (home/away), Elo diff, plus recent form; calibration makes probabilities actionable (thresholding/risk).
- Draws excluded; season drift, add player/injury info, monitor recalibration.

Conclusion & Reflection

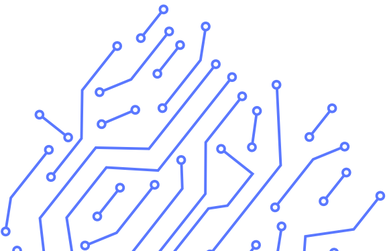
Model	Strengths	Weaknesses	Trade-offs / Best use
Logistic Regression	Simple, fast, interpretable; naturally calibrated probsa	Linear; lower ceiling (~0.67 Accuracy)	Baseline, insights, threshold tuning with trustable probs
Random Forest	Non-linear, robust to scaling; feature importance	Tends to overfit without depth/leaf control; probs often uncalibrated	Quick non-linear benchmark; needs strong regularisation
XGBoost	Top accuracy/recall; captures interactions; regularisation + early stop	More complex; needs tuning and calibration	Primary model for performance; use calibrated probs for decisions

Best Model

Conclusion & Reflection

Future work:

- 3-class model (home/draw/away) with softmax; per-class calibration.
- Richer signals: player availability/line-ups & injuries, manager changes, congestion; upgrade ratings to time-decayed Elo/Glicko.
- Robustness: stress-test on promotion/relegation teams, partial season, and rare events.



APPLICATIONS IN REAL LIFE

SPORTS ANALYSIS

BETTING

PERFORMANCE ANALYTICS

THANK YOU