UNIVERSITY OF SUSSEX

AI Agent and Recommender for Football Tactics: Live Football Substitution Prediction Using Machine Learning and Real-Time Contextual Features

MSc Dissertation Report
Artificial Intelligence and Adaptive Systems
University of Sussex

Candidate Number: 291105

Supervisor: Professor Thomas Nowotny
Department of Informatics
Summer 2025

Word Count: 12,300

Statement of Originality and Intellectual Property Rights

This report is submitted as part of the requirement for the MSc in Artificial Intelligence and Adaptive Systems at the University of Sussex. It is the product of my own labour, except where explicitly stated otherwise. This work may not be copied, reproduced, or used for any purpose other than assessment without prior permission from the author and the University. All data sources and external materials are acknowledged in full.

Signed: Mohamed Elsaygh Date: 19 August 2025

ACKNOWLEDGEMENTS

This project would not have been possible without the help and support of many people. I am deeply grateful to my supervisor, **Professor Thomas Nowotny**, whose expertise in AI, machine learning, and adaptive systems shaped every stage of this work and whose steady guidance made the research both rigorous and enjoyable. I am especially indebted to my **external supervisor, Mr Nigel Jacklin**, appointed by the University at the start of my MSc, for his practical advice, constructive challenge, and encouragement throughout; and to **George Jacklin** for his generous assistance and reality-checking of early ideas. I would also like to thank **Professor Chris Johnson**, whose Adaptive Systems module allowed me to explore AI in football tactics through evolutionary algorithms; achieving a distinction (82%) in that work crystallised the idea for this master's project. As **a former professional footballer**, this project is more than an academic exercise—it is the foundation for **a future startup**. I am also grateful to **Hudl (StatsBomb)** for the open football dataset that underpinned this research, and to the wider open-source community whose shared code, data, and documentation accelerated development. Finally, heartfelt thanks to my family for their patience and unwavering support, and to friends and colleagues for feedback, motivation, and good humour along the way. Any remaining errors are my own.

Abstract

This dissertation presents the design, development, and evaluation of an **AI Agent and Recommender for Football Tactics**, with a specific focus on **live football substitution prediction using machine learning and real-time contextual features**. The model addresses the growing need in pro football for data-driven, context-aware tactical decision support systems that can be implemented in real-time during matches.

High-resolution event, lineup, and tracking data from the StatsBomb open dataset are utilized in the study, supplemented with engineered tactical features like player stamina estimates, scoreline dynamics, match phase context, positional encodings, and recent event windows. A number of supervised models are trained and compared, i.e., Logistic Regression, Support Vector Machines, Random Forests, Artificial Neural Networks, XGBoost, and a Stacked Ensemble. Advanced methods in evaluation such as threshold tuning, SHAP-based explanation, and error analysis are applied to ensure stable performance and interpretability.

The experimental result verifies that context-aware features significantly enhance substitution prediction accuracy. XGBoost and Stacked Ensemble models came out tops across a set of evaluation metrics, with SHAP analysis confirming that tactical context features — e.g., score margin, time left in match, and stamina of players — are among the best predictors. The findings also highlight potential overfitting risks in small, domain-specific datasets, particularly when a class imbalance exists, and highlight the value of meticulous validation methods such as Leave-One-Team-Out cross-validation.

The following are the contributions of this work: (1) a new, completely reproducible machine learning pipeline for real-time football substitution prediction, (2) an explainable feature set that captures both match dynamics and player context, and (3) an evaluation framework that trades off predictive accuracy and explainability. Besides substitution prediction, the system presented here opens the door to more general real-time tactical recommendation systems, easily integratable in coach workflows. The dissertation concludes with the discussion of limitations, practical deployment challenges, and potential future research avenues, such as integration with richer tracking information, computer vision systems, and offensive/defensive tactical labelling.

Contents

Acronyms 4

1 Introduction **5**

2 Background and Literature Review **8**
2.1 Problem Context and Prior Work 8
2.2 Conceptual Foundations 8
2.3 System Scope: Artefact Overview 9
2.4 Explainability in Substitution Prediction 9
2.5 Methodological Themes in the Literature 9
2.6 Research Gap, Novelty, and Contribution 10
2.7 Comparative Literature Summary 10
2.8 Scope and Threats to Validity 10
2.9 Summary 11

3 Design of Study: Rationale, Methodology, and Execution **12**
3.1 Research Design and Objectives 12
3.2 Data Sources and Scope 13
3.2.1 Exploratory Data Analysis (EDA) 13
3.3 System Artefact Overview 17
3.4 Preprocessing, Integration, and Leakage Control 18
3.5 Feature Engineering (Real-Time Computable) 19
3.6 Label Construction 21
3.7 Learning Objectives and Losses 24
3.8 Validation, Calibration, and Thresholding Protocol 26
3.9 Interpretability Protocol 27
3.10 Ablations, Baselines, and Robustness Slices 28
3.11 Execution and Reproducibility 28

4 Results **29**
4.1 Pipeline A: Baseline Substitution Prediction 29
4.2 Pipeline B: Should-Be-Subbed Recommendation 37
4.3 Pipeline C: Offensive vs Defensive Substitution Classification 57
4.4 Pipeline D: When *Not* to Sub (Risk-Averse Filter) 66
4.5 Evaluation Methodology and Experimental Setup 68
4.6 Consolidated Error Analysis 70

5 Discussion **71**
5.1 What the Results Mean 71
5.1.1 Core findings in context 71

5.1.2 Implications for practice . . . 71
5.2 Consequences and Risks . . . 72
5.2.1 Consequences for live decision-making . . . 72
5.3 On Unusual Perfect Scores (Accuracy = 1.00; Macro-F1 = 1.00) . . . 72
5.4 Metric Choice: Why Accuracy, Macro-F1, Precision/Recall, and PR-AUC 73
5.5 Limitations and Threats to Validity . . . 73
5.5.1 Data and labels . . . 73
5.5.2 Features and potential leakage . . . 73
5.5.3 Methodological choices . . . 73
5.6 Merits and Contributions . . . 74
5.7 What We Learned . . . 74
5.8 Recommendations for Follow-Up Work . . . 74
5.8.1 Short term (next iteration) . . . 74
5.8.2 Medium term . . . 74
5.8.3 Long term . . . 75
5.9 Ethical and Operational Considerations . . . 75
5.10 Synthesis and Closing Remarks . . . 75

6 Conclusions **76**
6.1 Answer to the Research Question . . . 76
6.2 What Was Built and Shown . . . 76
6.3 Sensible, Evidence-Based Conclusions . . . 76
6.4 Consistency With the Evidence . . . 77
6.5 Contribution to Knowledge and Practice . . . 77
6.6 Limitations (Brief Recap) . . . 77
6.7 Recommendations . . . 77
6.8 Closing Statement . . . 77

A Appendix **80**
A.1 Supplementary Exploratory Data Analysis (EDA) . . . 80
A.2 Model Diagnostics and Learning Curves . . . 89
A.3 System Diagrams and Feature Pipeline . . . 94
A.4 System Requirements . . . 95
A.5 Testing . . . 96
A.5.1 Test Objectives . . . 97
A.5.2 Test Methodology . . . 97
A.5.3 Pipeline-Specific Testing . . . 97
A.5.4 Shared Modules . . . 98
A.6 Evaluation of the System . . . 98
A.6.1 Evaluation Methodology . . . 98
A.6.2 Mathematical Definitions of Metrics . . . 98
A.6.3 Experimental Setup . . . 99
A.6.4 Results and Comparative Analysis . . . 99
A.6.5 Error Analysis . . . 99
A.6.6 Interpretability and Explainability . . . 99
A.6.7 Limitations Observed . . . 99
A.6.8 Outcomes and Insights . . . 99
A.7 Secondary Data Analysis Compliance Form . . . 100

Secondary Data Analysis Compliance Form **100**

ACRONYMS

Abbreviation	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the Curve (ROC or PR)
CV	Cross-Validation
EDA	Exploratory Data Analysis
F1	Harmonic mean of precision and recall
k-NN	k-Nearest Neighbours
LR	Logistic Regression
LOGO	Leave-One-Group-Out (cross-validation)
ML	Machine Learning
MLP	Multi-Layer Perceptron (ANN)
PCA	Principal Component Analysis
PR	Precision–Recall
PR-AUC	Area Under the Precision–Recall Curve
RF	Random Forest
ROC	Receiver Operating Characteristic
ROC-AUC	Area Under the ROC Curve
SB	StatsBomb (open data)
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
UI	User Interface
XGB	eXtreme Gradient Boosting (XGBoost)
xG	Expected Goals
xT	Expected Threat

Chapter 1

Introduction

Football has shifted in recent decades from intuition-led coaching to data-informed, in-game tactical decisions. Substitution is a highly influential intervention affecting fatigue, shape, and results [1, 2]. Optical tracking and event logs (e.g., StatsBomb, Metrica, Wyscout [3]) enable high-frequency data that can inform real-time substitutions. Yet substitutions are often reactive, time-pressured, and subjective rather than evidence-based.

ML offers a structured, data-driven substitution strategy. Predictive modelling, real-time inference, and XAI already support rapid, accountable decisions in healthcare [4] and finance. Live substitution decision support is underexplored [5, 6]; most work is post-match, leaving a gap in real-time, explainable, coach-facing systems.

Research question. *Can ML enhance live substitution decisions via tactical, explainable, context-aware recommendations?* We adopt a design-science approach [7]: define the problem, build the artefact, and evaluate rigorously. The artefact ingests near-real-time match data and returns calibrated, explainable outputs.

Why this is challenging. Key risks: (i) leakage (same-match mixing or late-minute proxies), (ii) class imbalance/rare niches, (iii) proxy labels, (iv) external validity across teams/leagues, (v) real-time constraints. We address them via LOGO validation, leakage-controlled features, calibration with PR-driven thresholds, and SHAP-based auditability [8].

System at a glance. The architecture has four supervised pipelines targeting distinct decisions and sharing loaders, feature engineering, validation, calibration, and explanation utilities. Pipeline A (`main.py`) predicts substitution occurrence using context/stamina/position features and a 15-min window, SMOTE, and multiple models (LR, RF, XGB, ANN, SVM, k-NN, stacked). Pipeline B (`build_should_be_subbed.py`) outputs a calibrated *should-sub* recommendation using leakage-controlled features, GridSearch, isotonic calibration, and PR-based thresholding.
Pipeline C (`offensive_defensive_sub_analysis.py`) evaluates match-wise generalisation via LOGO CV, SMOTE-per-fold, PR-AUC-based thresholding, and SHAP; it saves match-wise probabilities for coach alignment. Pipeline D (`when_not_to_sub.py`) detects *do-not-sub* contexts using reversed/regularised labels, a `ColumnTransformer`, tuned models (GridSearchCV), post-hoc calibration, and SHAP/permutation importance. All pipelines emit SHAP justifications.

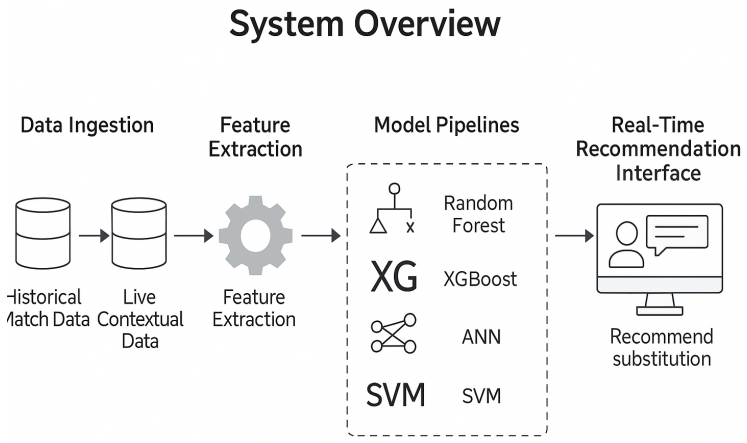


Figure 1.1: System architecture showing data ingestion, feature extraction, model pipelines, and real-time recommendation interface.

Outputs. (i) substitution probability/decision (A), (ii) calibrated *should-sub* score/decision (B), (iii) match-wise evaluation and error artefacts (C), and (iv) a context-aware *do-not-sub* alert (D). SHAP explanations attribute predictions to intelligible features (e.g., score, time, stamina) [4].

Table 1.1: Overview of the four model pipelines implemented in this study

Pipeline / Script	**Objective**	**Label**	**Key Techniques**
A / `main.py`	Substitution occurrence prediction	Binary (Sub / No Sub)	Feature engineering; SMOTE; LR/RF/SVM/k-NN/ANN/XGB/stacked; SHAP; confusion matrices/reports
B / `should_be_sub`	Should-be-subbed recommendation (tactical)	Binary (Should Sub / Hold)	Leakage-controlled features; GridSearchCV; isotonic calibration; PR-based thresholding; SHAP/feature importance
C / `off_def_sub`	Match-wise generalisation & error analysis	Offensive / Defensive (or aligned binary)	LOGO CV (match); SMOTE per fold; PR-AUC thresholding; SHAP; match-wise probability exports
D / `when_not_to_sub.py`	Do-Not-Sub detection	Binary (No Sub / Sub)	Reversed/regularised labels; `ColumnTransformer`; GridSearchCV; calibration; SHAP; permutation importance; misclassification reports

Motivation and interpretability. Substitutions can shift momentum [1]. Modern data provide sufficient temporal/contextual resolution for real-time modelling. To support adoption, all pipelines integrate SHAP for granular, auditable attributions [4].

Positioning and objectives. Prior substitution work spans statistics and ML [9] but often lacks (i) explicit interpretability, (ii) group-aware leakage checks, (iii) real-time deployability, and (iv) an integrated, multi-decision framework. We bridge these gaps with a modular, explainable, deployment-ready system for occurrence, should-sub scoring, coach-alignment, and do-not-sub detection. We evaluate multiple model families with Macro-F1/PR-AUC/ROC-AUC and reduced-feature variants for scalability/interpretability. Objectives: (1) build a real-time recommender from structured match data; (2) compare four pipelines under group-aware validation and calibrated thresholds; (3) deliver coach-friendly outputs that are predictively accurate and tactically interpretable.

Contributions. Contributions: (1) a real-time, modular artefact for occurrence, should-sub, intent, and do-not-sub; (2) decision-apt evaluation (LOGO, isotonic calibration, PR–F1 operating points, ablations/error slices); (3) end-to-end interpretability (SHAP/feature importance) with per-recommendation explanations; (4) evidence that compact models retain utility for low-latency deployment [7, 8].

Literature bridge. We build on three strands (Chapter 2): substitution timing/intent modelling, real-time analytics with group-aware evaluation, and explainable ML (SHAP) [4, 5, 9].

Chapter 2

Background and Literature Review

2.1 Problem Context and Prior Work

Football analytics has progressed from descriptive stats to learning from structured event and tracking data (StatsBomb, Opta, Wyscout) [10], enabling quantified pressing, possession value, and off-ball movement. Since 2010, xG, field tilt, and pass-value models have entered mainstream tactical analysis and recruitment [10, 11].

Substitutions as a decision problem. Substitutions influence fatigue, structure, and win probability [1, 2]. Decisions depend on static (role) and dynamic cues (stamina, game state, opposition). Compared with shooting/passing models, *live* substitution support is underdeveloped; most work is *post-hoc* or focuses on macro outcomes (e.g., win probability), offering indirect minute-by-minute guidance.

What has been done. Beal et al. [12] model timing (Bayesian game theory) without tactical intent. Liu & Ruano [13] analyse trends but not substitution logic. Related ML includes outcome prediction [14, 15], injury risk [16], and value-added player models [10]. Mohandas et al. [17] predict timing (RF, 9,000+ matches) without intent or practitioner-facing explanations. SHAP has matured for xG-style models [4, 18, 19] but is rarely embedded in *real-time* substitution recommenders. Grouped CV, calibration, and decision-theoretic thresholds are increasingly adopted [5, 8] but not yet standard in substitution support.

Design-science lens. We follow a design-science trajectory [7]: define the live, explainable substitution problem; build a four-pipeline artefact; evaluate on real matches; and reflect on mechanisms, limits, and generalisability—treating substitution support as a *constructive* contribution rather than retrospective analysis.

2.2 Conceptual Foundations

Substitution as a sequential, context-aware decision. Coaches repeatedly assess (score, minute, fatigue proxies, threat/control, discipline) under time pressure. Useful support needs: (i) context features (score, windows, role, momentum), (ii) temporal features (rolling windows), (iii) calibrated outputs for thresholds/risk trade-offs [8], and (iv) transparent explanations [4].

Grouped validation, leakage, and imbalance. Football data clusters by match/team. Random CV leaks across folds; Leave-One-Group-Out (LOGO) by `match_id` better matches deployment. Imbalance (many no-sub or rare defensive-intent cases) calls for Macro-F1,

SMOTE, calibration, and PR-based operating-point selection [8].

2.3 System Scope: Artefact Overview

A single high-level overview follows; implementation details are deferred to Methods. The artefact has four supervised pipelines sharing loading, feature engineering, validation, calibration, and explanation utilities:

- **Pipeline A (`main.py`)**: *Substitution occurrence* (Sub/No Sub). Uses engineered context/stamina/position features with a 15-min window; trains LR/RF/SVM/k-NN/ANN/XGB/stacked; SMOTE for imbalance; SHAP for explanations.
- **Pipeline B (`build_should_be_subbed.py`)**: *Should-be-subbed recommendation.* Leakage-controlled features; GridSearch; isotonic calibration; PR-curve thresholding for an operational decision.
- **Pipeline C (`offensive_defensive_sub_analysis.py`)**: *Match-wise generalisation and error analysis.* LOGO CV by match, SMOTE per fold, PR-AUC-based thresholding, SHAP for trees; saves match-wise probabilities for coach-alignment.
- **Pipeline D (`when_not_to_sub.py`)**: *Do-Not-Sub detection.* Reversed/regularised labels; `ColumnTransformer`; grid search; calibration; SHAP; permutation importance; misclassification exports.

2.4 Explainability in Substitution Prediction

Why SHAP here. Tree ensembles (RF, XGB) often beat linear baselines on tabular, non-linear features but are opaque. SHAP provides faithful local attributions (local accuracy, missingness, consistency) and is efficient for trees [4, 19]. Here SHAP is operational: (i) *coach-facing* transparency (drivers like minute, score margin, stamina proxies), and (ii) *validation* of domain hypotheses (e.g., time/score rank high; leakage-prone features should not dominate under LOGO).

Limits of post-hoc explanations. SHAP is explanatory, not causal; it cannot answer "sub now vs. later?" counterfactuals. Being local/additive, strong interactions may be partially visible. These limits motivate simulation/causal tools for scenarios but do not preclude useful, auditable live explanations.

2.5 Methodological Themes in the Literature

Evaluation realism. Many studies use random splits despite match clustering [10, 17], inflating performance. Grouped CV (LOGO or leave-one-team-out, LOTO) better reflects deployment with unseen matches/teams. We adopt LOGO by match as the default.

Metrics under imbalance. Accuracy misleads under dominant negatives (e.g., no-sub). Macro-F1 balances classes; PR-AUC captures positive-class retrieval. Calibration plus an F1-maximising PR threshold turns scores into actionable decisions [8].

Human-in-the-loop and recent developments. Explanations matter in expert domains. SHAP-based xG interpretation exists [18], but end-to-end *live* substitution

recommenders with calibrated decisions and rationales are rare. Recent work stresses grouped/temporal CV, calibration, and slice robustness [5, 6], underpinning our design.

2.6 Research Gap, Novelty, and Contribution

Observed gaps. (i) Limited focus on *tactical intent* beyond timing; (ii) scarce *group-aware validation*; (iii) minimal *deployment realism* (latency, calibrated thresholds, slice robustness); (iv) weak *explanations* tied to coach decisions.

Novelty and engagement with recent developments. We integrate grouped CV, calibration/threshold optimisation [8], and SHAP-based validation [4] into a *live* setting, contributing a multi-pipeline design (occurrence, should-sub, intent alignment, do-not-sub). We also add reduced-feature variants (top-k tree importances) to balance latency and interpretability.

This study's contribution. A modular, explainable, deployment-oriented artefact that: (1) addresses occurrence, should-sub timing, intent/coach alignment, and do-not-sub; (2) uses real-time-computable features; (3) evaluates with LOGO, Macro-F1/PR-AUC, calibration, and threshold tuning; (4) embeds SHAP throughout. We also analyse *why* behaviour arises and flag construct-validity risks (e.g., proxy-driven perfect scores).

2.7 Comparative Literature Summary

Study	**Task**	**Data**	**Model**	**Limitations / Notes**
Beal et al. [12]	Substitution optimisation (timing)	EPL, 10K+ matches	Bayesian game theory	No tactical intent; not explainable/real-time
Cavus and Biecek [18]	xG interpretability	Event logs	XGBoost + SHAP	No substitution modelling
Mohandas et al. [17]	Substitution timing	9,000+ matches	Random Forest	No intent class; minimal explainability
Decroos et al. [10]	Interpretable player valuation	Belgian Pro League	VAEP + Logistic Regression	No substitution modelling; scoring only
This study	Live substitution support (occurrence, should-sub, intent/coach alignment, do-not-sub)	StatsBomb-format events (+ engineered context)	LR/RF/SVM/ANN/XGB/stacked; SHAP; LOGO CV	Real-time, calibrated, explainable, multi-task recommender

Table 2.1: Comparison of prior literature and this project's positioning. The final row reflects this dissertation's contributions.

2.8 Scope and Threats to Validity

Data coverage. Experiments use StatsBomb-format events; 360° frames are partial; no physiological tracking (stamina proxied from events). Broader coverage (more leagues; verified intent labels) is future work.

Domain shift. Substitution norms vary by league/coach. Generalisation is non-trivial; grouped CV mitigates but does not eliminate shift risk.

Construct validity. SHAP is not causal; perfect intent scores may reflect proxy labels or leakage. We treat such results as prompts for label audits, stronger regularisation, and expanded datasets.

Deployment realism. Offline evaluation omits latency, UI, and feedback loops. Real-time-computable features and calibrated thresholds narrow the gap, but live trials are pending.

2.9 Summary

Substitution modelling lags other analytics: intent is rarely modelled, match-level clustering is often ignored, and deployable explainability is scarce. This design-science work bridges those gaps with a modular artefact using real-time-computable features, calibrated decisions, and SHAP explanations, validated under grouped CV. The Methods chapter details datasets, features, objectives, calibration, grouped CV, and the statistical protocol.

Chapter 3

Design of Study: Rationale, Methodology, and Execution

This chapter outlines the research design, methodology, and execution for developing an AI-powered artefact for real-time, explainable substitution support in football. Adopting a *design-science* paradigm [7], we: (i) articulate the problem of real-time tactical decision-making; (ii) design a modular four-pipeline system; (iii) evaluate it on real-world football data with deployment-realistic validation; and (iv) reflect on generalizable insights and validity threats. The focus is on *why* the system works in context, supported by mathematical rigor and alignment with the provided code scripts.

3.1 Research Design and Objectives

Problem Relevance. Football coaches face time-critical substitution decisions under uncertainty, with data clustered by match and team, and non-stationarity across competitions. The system must be *context-aware* (e.g., score margin, player fatigue), *calibrated* for reliable probabilities, *fast* for real-time use, and *explainable* for coach trust [20].

Artefact. We develop a four-pipeline system (A–D) that ingests StatsBomb-format event and lineup data, outputting calibrated, explained substitution recommendations for: occurrence (sub vs. no sub), timing, tactical intent, and do-not-sub safeguards. The pipelines are implemented in scripts like `main.py`, `build_should_be_subbed.py`, `offensive_defensive_sub_analysis.py`, and `when_not_to_sub.py`.

Evaluation. We use grouped cross-validation (Leave-One-Group-Out, LOGO) to prevent match-level leakage, calibrate probabilities via isotonic methods, select thresholds on precision-recall curves, quantify uncertainty with bootstrap confidence intervals, and compare models using non-parametric tests. Explainability via SHAP [19] is integral, ensuring tactical interpretability.

Contribution. A modular, real-time, calibrated, and explainable decision-support system for football tactics, validated on clustered data, with novel contextual features (e.g., 15-minute event windows, fatigue proxies).

Justification of Design Process. The design-science approach ensures problem-driven development, iterative refinement, and rigorous evaluation, aligning with real-world deployment needs [7]. The modular pipeline structure facilitates maintenance and extensibility, while the focus on real-time features and explainability addresses coach-facing requirements.

3.2 Data Sources and Scope

We use StatsBomb open data (`https://github.com/statsbomb/open-data`), comprising JSON files for competitions, matches, lineups, and event logs (passes, shots, duels, substitutions) with timestamps and coordinates. A subset includes 360° freeze-frames, proxied by event-based spatial features (e.g., carry distances) when absent. Features are restricted to those computable in near real-time (e.g., score margin, recent event counts, stamina proxies) to ensure deployability, as implemented in `data_loader.py` and `feature_engineering.py`.

3.2.1 Exploratory Data Analysis (EDA)

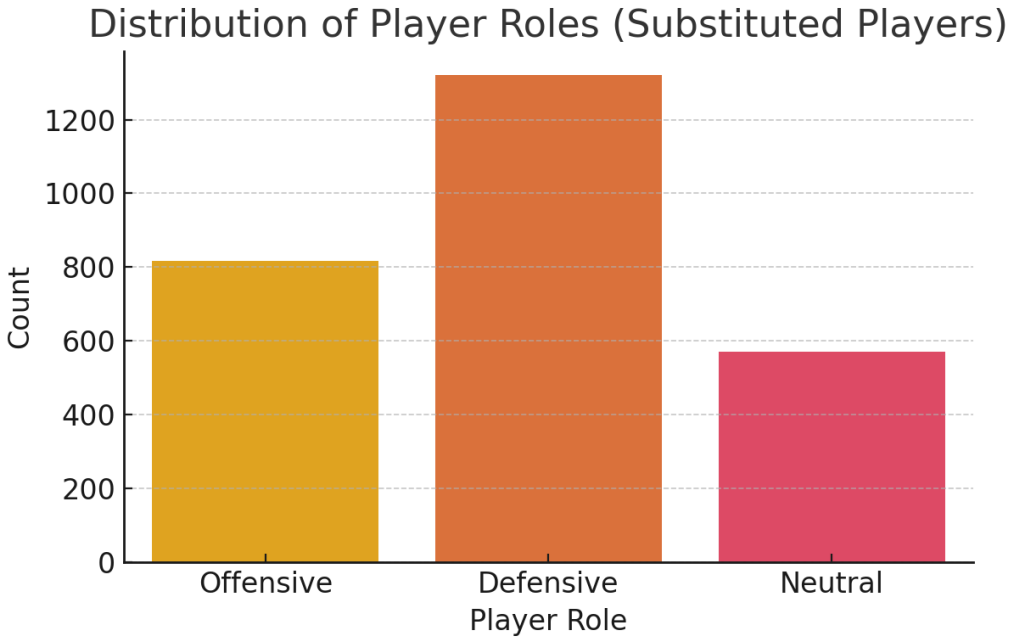


Figure 3.1: Distribution of player roles among *substituted* players.

We observe that defensive roles dominate raw substitution counts, partly reflecting roster composition

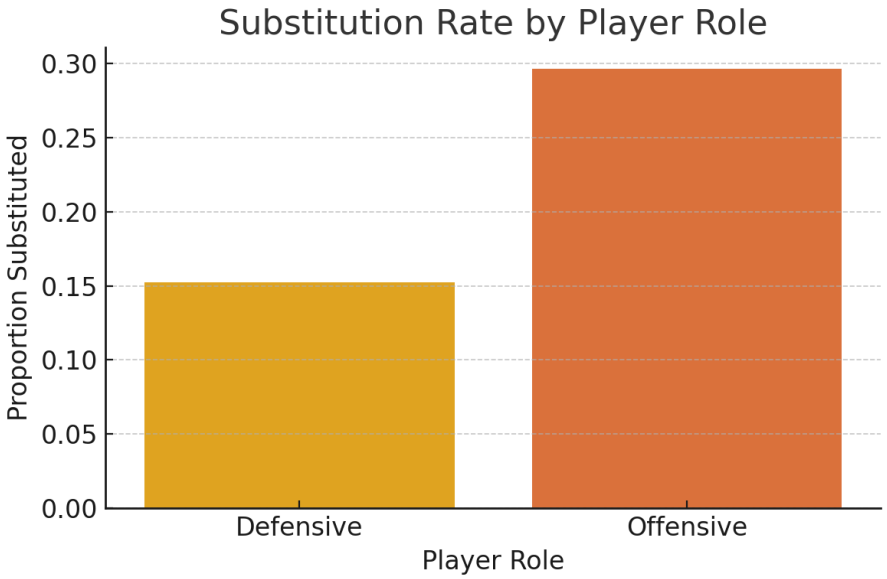


Figure 3.2: Substitution *rate* by role (share of players in the role who were substituted).

We observe that attackers are substituted at a substantially higher rate than defensive players, consistent with late-game tactical refresh

Fig. 3.1 shows raw counts *of subbed players by role—these reflect roster/appearance mix, so more defenders can yield larger totals.*
Fig. 3.2 shows the rate *(probability) of substitution within each role, i.e., $P(\text{subbed} \mid \text{role})$; hence attackers can have a higher rate even if defenders dominate counts.*

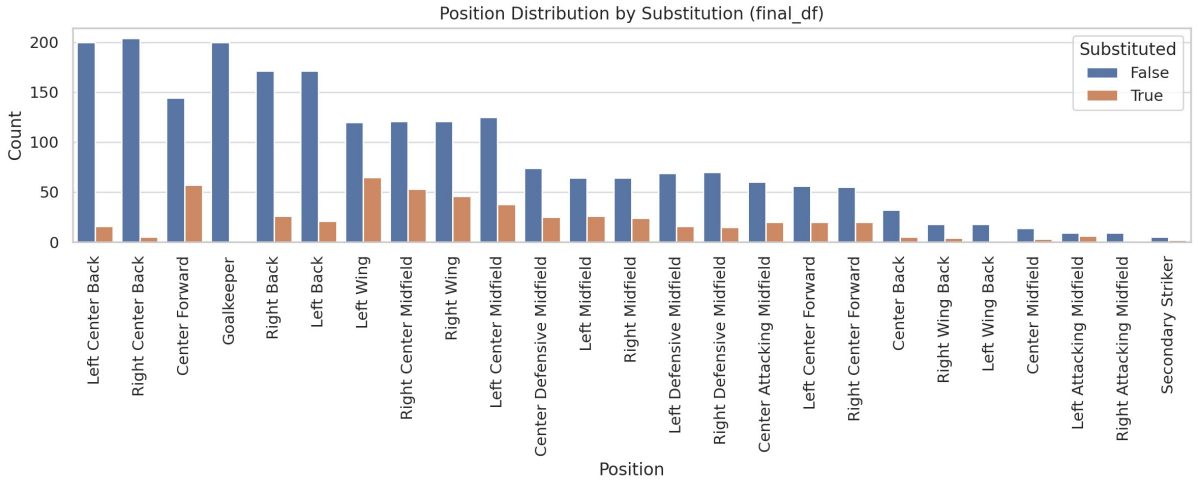


Figure 3.3: Counts per position, split by substitution outcome (final dataset).

We observe that forwards and wide roles are replaced most frequently, whereas goalkeepers and centre-backs are rarely substituted mid-match

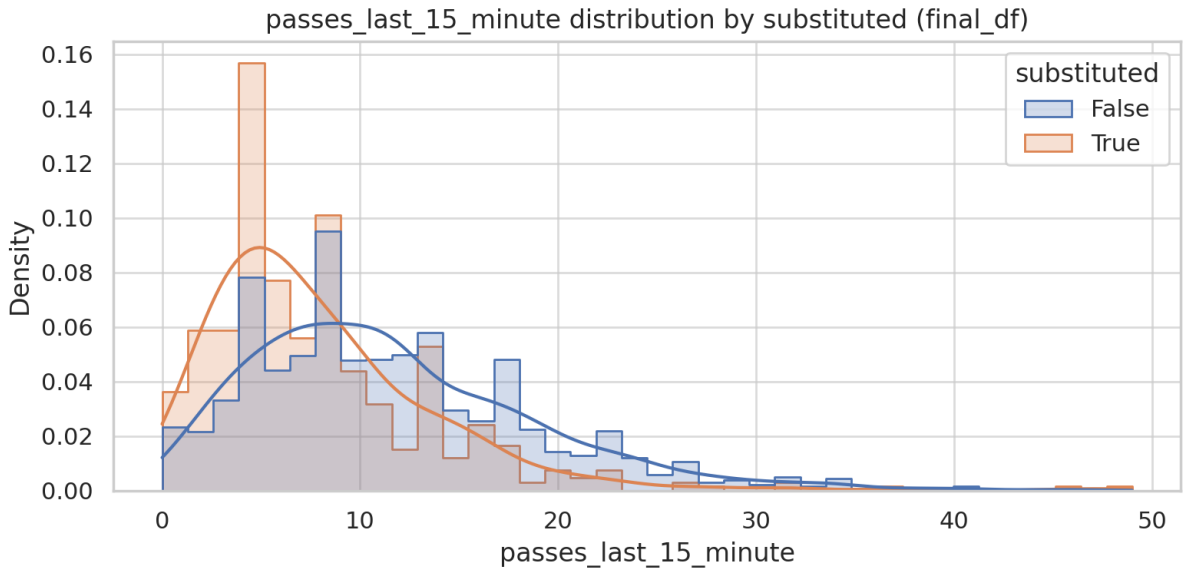


Figure 3.4: Recent passing involvement by actual substitution outcome (final dataset).

We observe that substituted players skew toward lower recent pass counts, indicating that reduced on-ball involvement often precedes exits

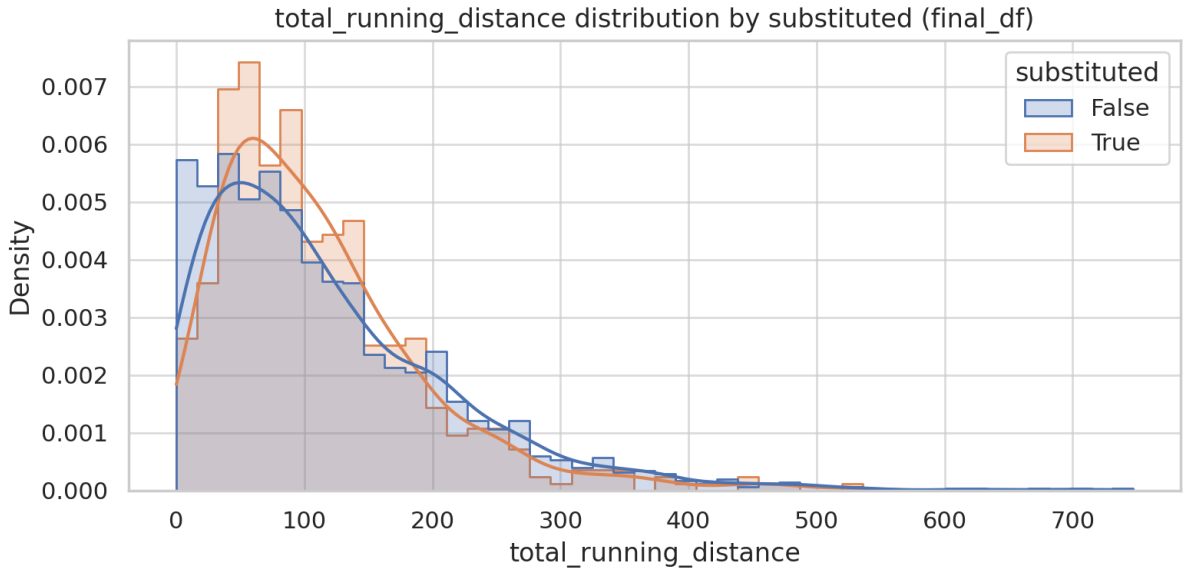


Figure 3.5: Total running distance by actual substitution outcome (final dataset).

We observe that the distributions overlap, with substituted players exhibiting a slightly higher typical workload across many positions

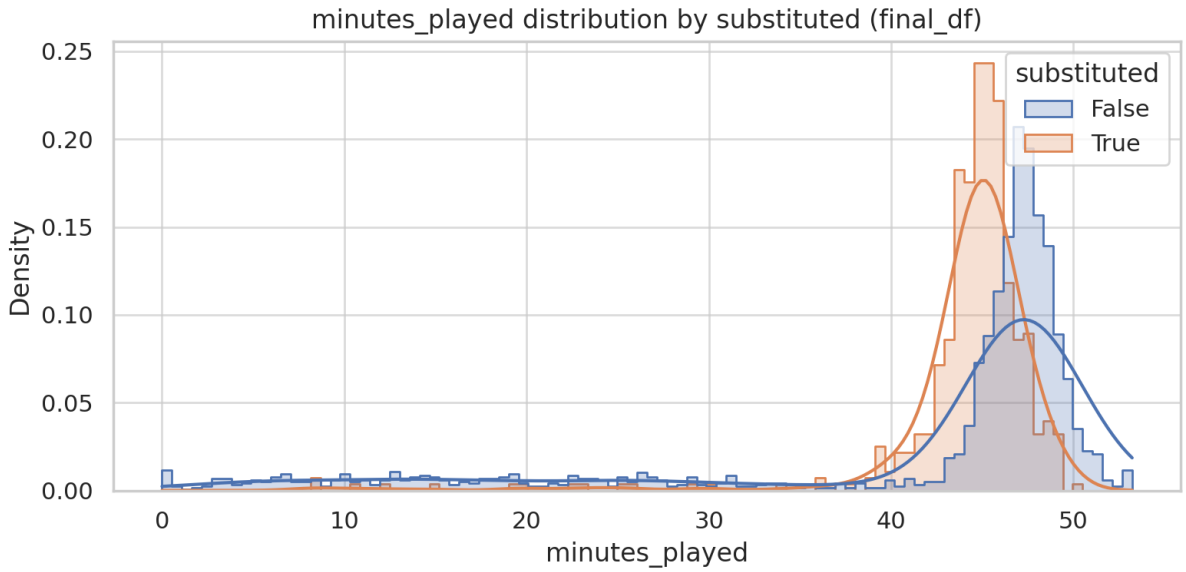


Figure 3.6: Minutes played distribution by actual substitution outcome (final dataset).

We observe that substitutions cluster around the 45–48 window (half-time / early second half), whereas non-subbed players accumulate minutes closer to full time

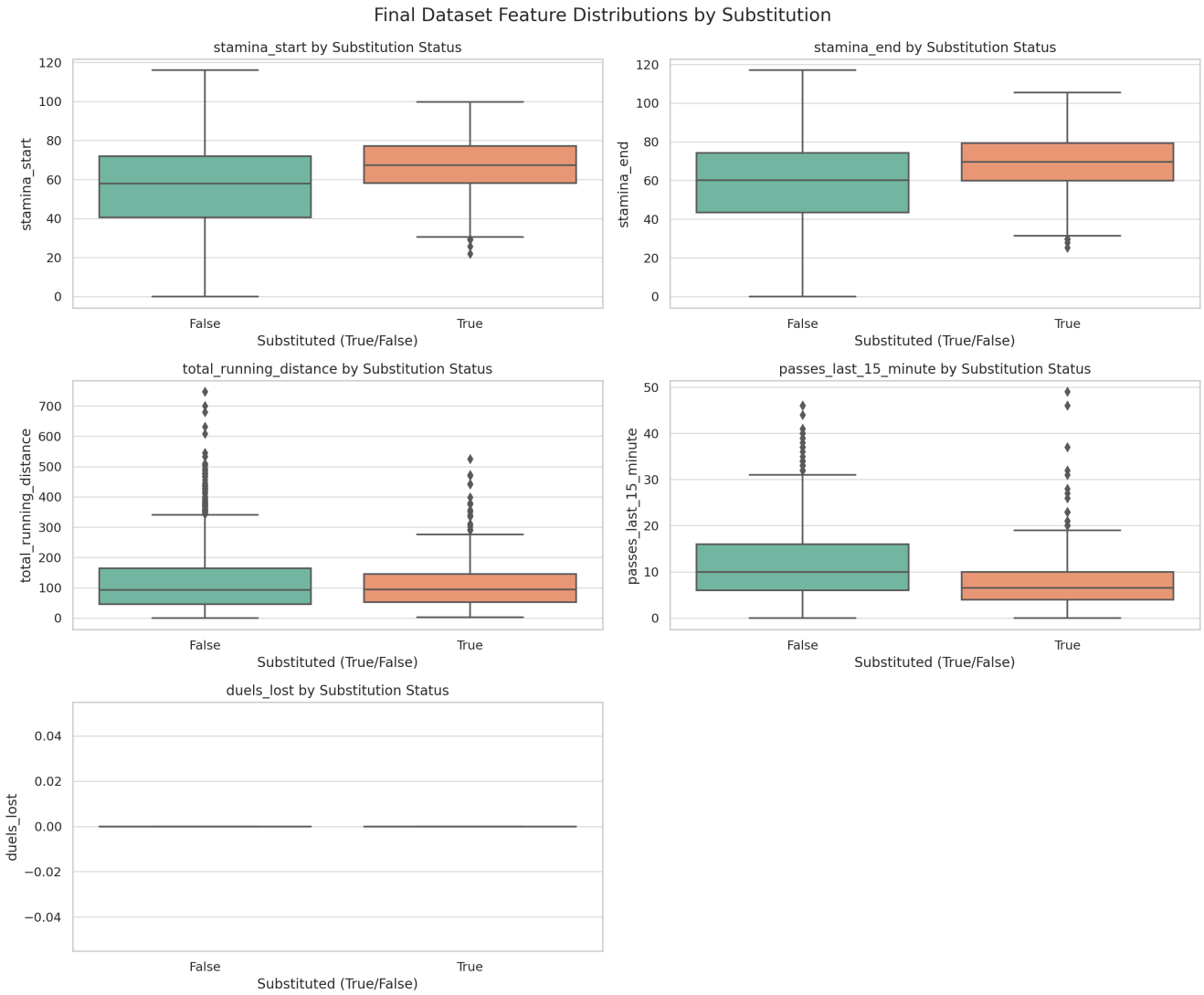


Figure 3.7: Final dataset feature distributions by substitution status (boxplots for stamina, distance, recent passes, duels lost).

We observe that subbed players exhibit lower recent passes and, because they exit earlier, higher residual stamina; consequently, minute and other context features are crucial

3.3 System Artefact Overview

The system comprises four pipelines, summarized below and detailed in subsequent sections:

- **Pipeline A (`main.py`)**: Predicts substitution occurrence (sub/no sub) using multiple models (Logistic Regression, Random Forest, SVM, k-NN, ANN, XGBoost, Stacking). Features include 15-minute rolling windows and context (e.g., score margin), with SMOTE for imbalance and SHAP for explainability.
- **Pipeline B (`build_should_be_subbed.py`)**: Recommends when players should be substituted, using leakage-controlled features, grid search, isotonic calibration, and PR-curve threshold selection. Compares full vs. reduced feature sets.

- **Pipeline C (`offensive_defensive_sub_analysis.py`)**: Analyzes match-wise generalization and tactical intent (offensive/defensive) using LOGO CV, SMOTE per fold, PR-AUC, and SHAP for tree models.
- **Pipeline D (`when_not_to_sub.py`)**: Identifies contexts where substitutions should be avoided, using regularized labels, `ColumnTransformer`, grid search, calibration, and permutation importance.

Shared modules (`data_loader.py`, `feature_engineering.py`, `explainability.py`, `visualizations.py`) ensure consistency and reproducibility.

3.4 Preprocessing, Integration, and Leakage Control

Raw JSON events and lineups are parsed into Pandas DataFrames (`data_loader.py`, `load_events.py`), keyed by `match_id` and `player_id`. Events are aggregated into a matrix:

$$\mathbf{D} \in \mathbb{R}^{n \times d}, \quad e_i = (t_i, p_i, \tau_i, \mathbf{x}_i)$$

Symbol Explanations: - $\mathbf{D}$: The aggregated event matrix. - $\mathbb{R}^{n \times d}$: The set of real-valued matrices with n rows (events) and d columns (features). - n: Number of events. - d: Number of features per event. - e_i: The i-th event row. - t_i: Timestamp of the i-th event. - p_i: Player ID for the i-th event. - τ_i: Event type for the i-th event. - $\mathbf{x}_i$: Attribute vector (e.g., coordinates) for the i-th event.
Equation Explanation: This equation represents the structured aggregation of raw event data into a matrix $\mathbf{D}$ where each row e_i captures an event's timestamp, player, type, and attributes, enabling efficient tabular processing in machine learning pipelines.

where n is events, d features, t_i timestamp, p_i player ID, τ_i event type, and $\mathbf{x}_i$ attributes (e.g., coordinates). Missing values are imputed for continuous features or omitted for sparse spatial data. Categorical features (e.g., position) are one-hot encoded:

$$\mathbf{p}_i = [\mathbb{I}(pos_i = \text{GK}), \mathbb{I}(pos_i = \text{DEF}), \ldots, \mathbb{I}(pos_i = \text{FWD})]^T$$

Symbol Explanations: - $\mathbf{p}_i$: One-hot encoded position vector for the i-th player. - $\mathbb{I}(\cdot)$: Indicator function (1 if true, 0 otherwise). - pos_i: Position category of the i-th player. - GK, DEF, ..., FWD: Position categories (e.g., Goalkeeper, Defender, Forward). - ^{T}: Transpose operator, making it a column vector.
Equation Explanation: This encodes categorical player positions into a binary vector using one-hot encoding, where only the matching category is 1 and others are 0, facilitating numerical input for models while preserving mutual exclusivity.

High-cardinality fields are reduced to avoid sparsity, as the dataset contained many different positions (e.g. GK, LB, ST, etc).

Leakage Control. Features like 15-minute windows (`compute_15_min_context`) are computed with strict temporal cutoffs to avoid future information. In Pipeline B, leakage-prone features (e.g., `passes_last_15_minute`) are dropped. SMOTE, scalers, and calibrators are fit only on training folds, as in `build_should_be_subbed.py` and `when_not_to_sub.py`.

Standardization. Numerical features are scaled:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \quad \mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}, \quad \sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_{ij} - \mu_j)^2}$$

Symbol Explanations: - $\tilde{x}_{ij}$: Standardized value for the i-th row and j-th feature. - x_{ij}: Original value for the i-th row and j-th feature. - μ_j: Mean of the j-th feature across m

samples. - σ_j: Standard deviation of the j-th feature across m samples. - m: Number of samples (rows) in the dataset. - $\sum$: Summation operator.
Equation Explanation: This standardizes numerical features by subtracting the mean μ_j and dividing by the standard deviation σ_j for each column j, ensuring zero mean and unit variance to improve model convergence and comparability. Implemented in `main.py` and `when_not_to_sub.py` using `StandardScaler`.

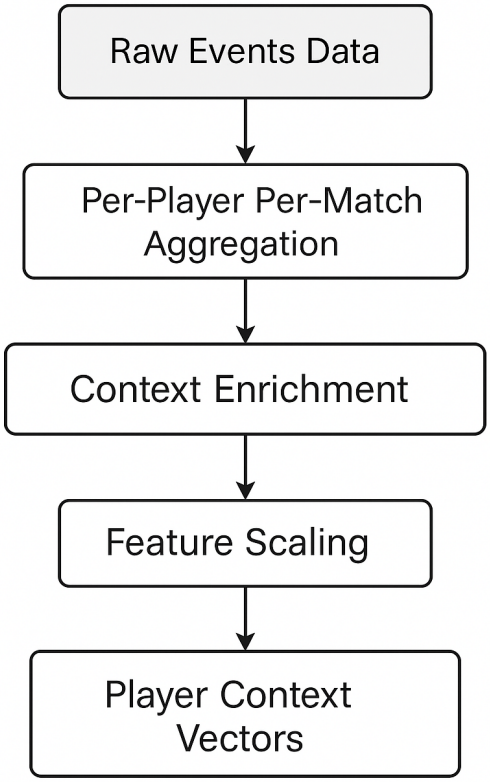


Figure 3.8: From raw events to player context vectors: (i) per-player per-match aggregation, (ii) context enrichment (time/score/role, rolling windows), (iii) scaling/encoding, yielding deployment-ready feature vectors.

The diagram clarifies the leak-safe data path: all transforms prior to modelling are real-time computable and are applied *inside* cross-validation folds to prevent any look-ahead.

3.5 Feature Engineering (Real-Time Computable)

Features are grouped into context, temporal, role, event windows, momentum, fatigue, and discipline, as shown in Table 3.1. All are computable in real-time from event streams, as in `feature_engineering.py` and `substitution_predictor.py`.

Table 3.1: Illustrative, live-computable feature schema

Group	Examples	Type	Derived from
Match context	score_margin, team_losing, home/away	numeric/binary	score updates, metadata
Temporal phase	minute, phase dummies (45–60, 60–75, 75–90)	numeric/binary	event clock
Role/position	position_name (one-hot)	categorical	lineups
Event windows	passes_last_15, duels_won_last_15, fouls_drawn_last_15	numeric	sliding window
Momentum	xThreat proxies start/end, delta_possession	numeric	windowed aggregates
Fatigue proxies	total_running_distance, stamina_start/end	numeric	carries/movement
Discipline	yellow_risk, fouls_committed, cards	numeric/binary	event aggregates

- **Event Counts.** Per player p, match m, event type τ:

$$f_{p,m}(\tau) = \sum_{i:p_i=p,m_i=m} \mathbb{I}(\tau_i = \tau)$$

Forms vector $\mathbf{f}_{p,m} = [f_{p,m}(\text{Pass}), f_{p,m}(\text{Duel}), \dots]^T$ (`extract_player_match_features`).
Symbol Explanations: - $f_{p,m}(\tau)$: Count of events of type τ for player p in match m. - p: Player identifier. - m: Match identifier. - τ: Event type (e.g., Pass, Duel). - $\sum_{i:p_i=p,m_i=m}$: Summation over all events i where player and match match p and m. - $\mathbb{I}(\tau_i = \tau)$: Indicator function (1 if event type τ_i equals τ, 0 otherwise). - $\mathbf{f}_{p,m}$: Vector of event counts for player p in match m. - ^{T}: Transpose operator, making it a column vector.
Equation Explanation: This counts the number of events of a specific type τ (e.g., Pass) for a player p in match m, aggregated into a vector $\mathbf{f}_{p,m}$ for feature extraction.

- **Running Distance.** For carry event e_i:

$$d_i \approx |x_{end,i} - x_{start,i}|, \quad D_{p,m} = \sum_{i:p_i=p,m_i=m,\tau_i=\text{Carry}} d_i$$

Proxies fatigue (`extract_stamina_features`).
Symbol Explanations: - d_i: Approximate distance of the i-th carry event. - $x_{end,i}$, $x_{start,i}$: End and start coordinates of the i-th carry event. - $|\cdot|$: Absolute difference (distance approximation). - $D_{p,m}$: Total running distance for player p in match m. - $\sum_{i:p_i=p,m_i=m,\tau_i=\text{Carry}}$: Summation over all carry events for player p in match m.
Equation Explanation: This approximates the distance d_i of each carry event as the straight-line difference between start and end coordinates, summing to $D_{p,m}$ to estimate fatigue.

- **Score Margin.** For event e_i:

$$s_{margin,i} = s_{team,i} - s_{opp,i}, \quad l_{team,i} = \mathbb{I}(s_{margin,i} < 0)$$

Captures tactical context (`add_match_context_features`).
Symbol Explanations: - $s_{margin,i}$: Score margin for the i-th event. - $s_{team,i}$: Team score at the i-th event. - $s_{opp,i}$: Opponent score at the i-th event. - $l_{team,i}$: Binary indicator (1 if team is losing, 0 otherwise). - $\mathbb{I}(s_{margin,i} < 0)$: Indicator function (1 if score margin is negative, 0 otherwise).

Equation Explanation: This computes the score margin $s_{margin,i}$ as the difference between team and opponent scores, with $l_{team,i}$ indicating if the team is losing for tactical analysis.

- **Timing Windows.** Binary indicators:

$$w_{45-60}(t) = \mathbb{I}(45 \leq t < 60), \quad w_{60-75}(t) = \mathbb{I}(60 \leq t < 75), \quad w_{75+}(t) = \mathbb{I}(t \geq 75)$$

For substitution timing (`add_timing_features`).
Symbol Explanations: - $w_{45-60}(t)$, $w_{60-75}(t)$, $w_{75+}(t)$: Binary indicators for time windows. - t: Timestamp of an event. - $\mathbb{I}(\cdot)$: Indicator function (1 if condition is true, 0 otherwise).
Equation Explanation: These define binary flags for specific time intervals (45-60, 60-75, 75+) based on event timestamp t, aiding substitution timing analysis.

- **15-Minute Context.** Passes in window $[t_{last} - 15\,\text{min}, t_{last}]$:

$$c_{passes,p,m} = \sum_{i:p_i=p,m_i=m,\tau_i=\text{Pass}} \mathbb{I}(t_{last} - 15\,\text{min} \leq t_i \leq t_{last})$$

Dynamic tactical feature (`compute_15_min_context`).
Symbol Explanations: - $c_{passes,p,m}$: Count of passes by player p in match m within the 15-minute window. - t_{last}: Latest timestamp considered. - $\sum_{i:p_i=p,m_i=m,\tau_i=\text{Pass}}$: Summation over all pass events for player p in match m. - $\mathbb{I}(t_{last} - 15\,\text{min} \leq t_i \leq t_{last})$: Indicator for events within the 15-minute window.
Equation Explanation: This counts passes by player p in match m within a 15-minute window ending at t_{last}, providing a dynamic tactical context.

- **Performance Deltas.** Sum of metrics:

$$\delta_{p,m,j} = \sum_{i:p_i=p,m_i=m} x_{i,j}$$

For numeric metrics like duels lost (`extract_performance_deltas`).
Symbol Explanations: - $\delta_{p,m,j}$: Sum of metric j for player p in match m. - $x_{i,j}$: Value of metric j for the i-th event. - $\sum_{i:p_i=p,m_i=m}$: Summation over all events for player p in match m.
Equation Explanation: This sums the values of a numeric metric j (e.g., duels lost) across all events for player p in match m, capturing performance changes.

Justification. These features are novel in combining real-time, sliding-window aggregates with tactical constructs (e.g., xThreat, fatigue), unlike static metrics in prior work [21]. They balance computational efficiency and tactical relevance, critical for live deployment.

3.6 Label Construction

Labels are defined for each pipeline, grounded in tactical criteria or observed events.

- **Pipeline A (Occurrence, `label_builder.py`).** Binary label for substitution:

$$y_{p,m} = \mathbb{I}\left(\exists e_i : \tau_i = \text{Substitution} \wedge p_i = p \wedge m_i = m\right)$$

Merged with non-substituted players (`extract_substitution_labels_from_events`).
Symbol Explanations: - $y_{p,m}$: Binary label (1 if substituted, 0 otherwise) for player p in match m. - $\mathbb{I}(\cdot)$: Indicator function (1 if condition is true, 0 otherwise). - $\exists e_i$: Existence of at least one event e_i. - τ_i: Event type of the i-th event. - p_i: Player ID of the i-th event. - m_i: Match ID of the i-th event. - p, m: Specific player and match identifiers.
Equation Explanation: This sets $y_{p,m}$ to 1 if there exists at least one substitution event for player p in match m, otherwise 0, labeling substitution occurrences.

- **Pipeline B (Should-Sub, `label_builder_substitution.py`).** Rule-based label:

$$F_{p,m} = (S_{start,p,m} - S_{end,p,m}) + M_{p,m}, \quad E_{p,m} = D_{lost,p,m} + P_{bad,p,m}, \quad \Delta T_{p,m} = T_{start,p,m} - T_{end,p,m}$$

$$y_{p,m} = \mathbb{I}\left(F_{p,m} \geq \theta_F \vee E_{p,m} \geq \theta_E \vee \Delta T_{p,m} \geq \theta_T\right)$$

where $\theta_F = 70$, $\theta_E = 5$, $\theta_T = 0.1$ (defaults, tunable via domain expertise). Implemented in `define_should_be_subbed`.

Symbol Explanations: - $F_{p,m}$: Fatigue score for player p in match m. - $S_{start,p,m}$, $S_{end,p,m}$: Start and end stamina levels for player p in match m. - $M_{p,m}$: Additional match-specific metric for player p in match m. - $E_{p,m}$: Error score for player p in match m. - $D_{lost,p,m}$: Number of duels lost by player p in match m. - $P_{bad,p,m}$: Number of bad passes by player p in match m. - $\Delta T_{p,m}$: Time difference (e.g., minutes played) for player p in match m. - $T_{start,p,m}$, $T_{end,p,m}$: Start and end times for player p in match m. - θ_F, θ_E, θ_T: Thresholds for fatigue, error, and time (70, 5, 0.1 respectively). - $\vee$: Logical OR operator.

Equation Explanation: This computes fatigue $F_{p,m}$, error $E_{p,m}$, and time difference $\Delta T_{p,m}$, setting $y_{p,m}$ to 1 if any exceeds its threshold, indicating a player should be substituted.

- **Pipeline B (Alternative, `label_builder.py`).** Simplified rule:

$$y_{p,m} = \mathbb{I}\left(D_{lost,p,m} > 3 \vee c_{passes,p,m} < 5\right)$$

Used for robustness (`build_should_be_subbed_labels`).

Symbol Explanations: - $y_{p,m}$: Binary label (1 if should be subbed, 0 otherwise) for player p in match m. - $D_{lost,p,m}$: Number of duels lost by player p in match m. - $c_{passes,p,m}$: Count of passes by player p in match m within a window. - $\mathbb{I}(\cdot)$: Indicator function (1 if condition is true, 0 otherwise). - $\vee$: Logical OR operator.

Equation Explanation: This sets $y_{p,m}$ to 1 if a player loses more than 3 duels or makes fewer than 5 passes, providing a simpler substitution criterion for robustness testing.

- **Pipeline C (Intent).** Labels offensive/defensive based on position changes and context (heuristics in `offensive_defensive_sub_analysis.py`), planned for expert relabeling.

- **Pipeline D (Do-Not-Sub).** Reversed labels from Pipeline B, regularized to prioritize retention (`when_not_to_sub.py`).

Justification. Rule-based labels decouple predictions from coach decisions, enabling strategy-driven recommendations, unlike purely observational labels [22].

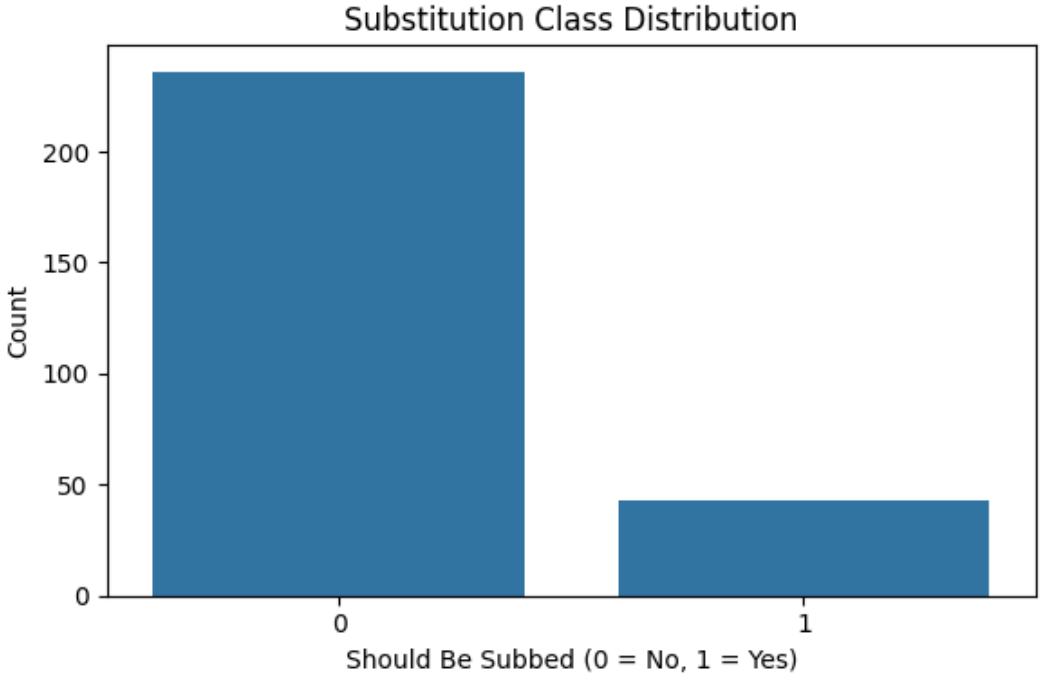


Figure 3.9: Class balance for the *should_be_subbed* label (0 = hold, 1 = sub).

We observe that the class distribution is imbalanced with a minority positive class, which motivates reporting Macro-F1 and PR-AUC in preference to Accuracy and applying SMOTE *on training folds only* to stabilise thresholded performance.

Feature–Label Relationships (Should-Sub Dataset)

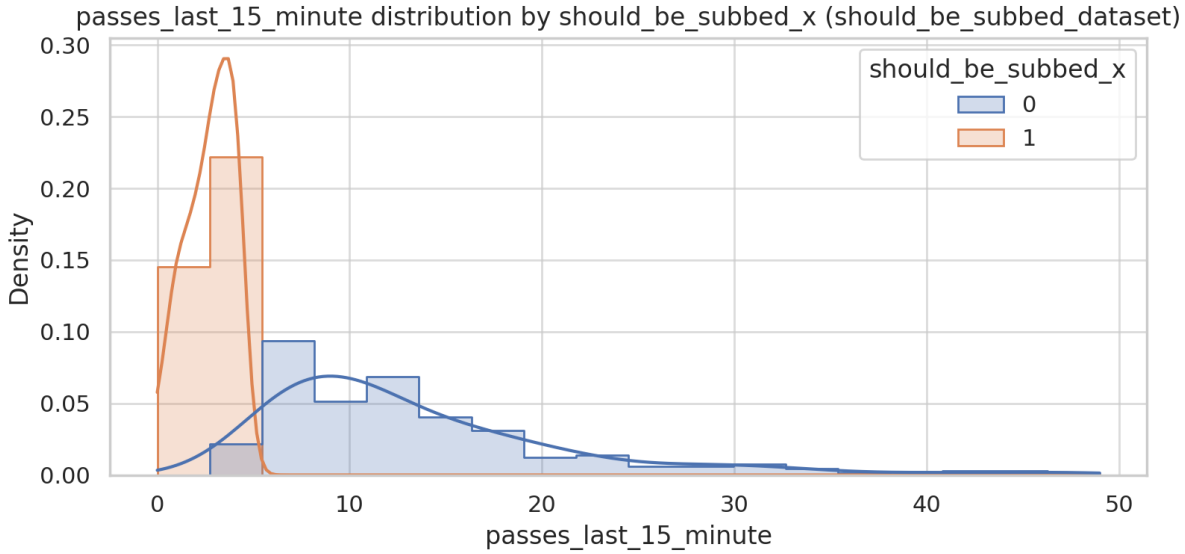


Figure 3.10: Distribution of `passes_last_15_minute` by `should_be_subbed_x` label.

We observe that the ***`should_sub`*** *class concentrates at low recent-passes values, providing a strong positive signal for substitution recommendations.*

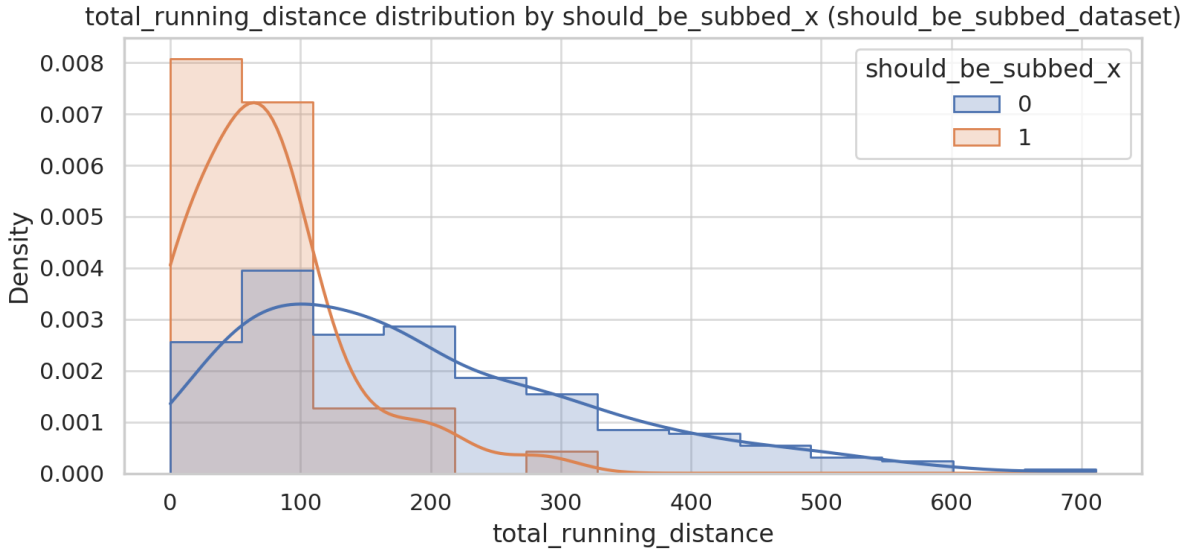


Figure 3.11: Distribution of `total_running_distance` by `should_be_subbed_x` label.

We observe that ***`should_sub`*** *observations tend to have lower cumulative distance, pointing to under-involvement rather than extreme fatigue.*

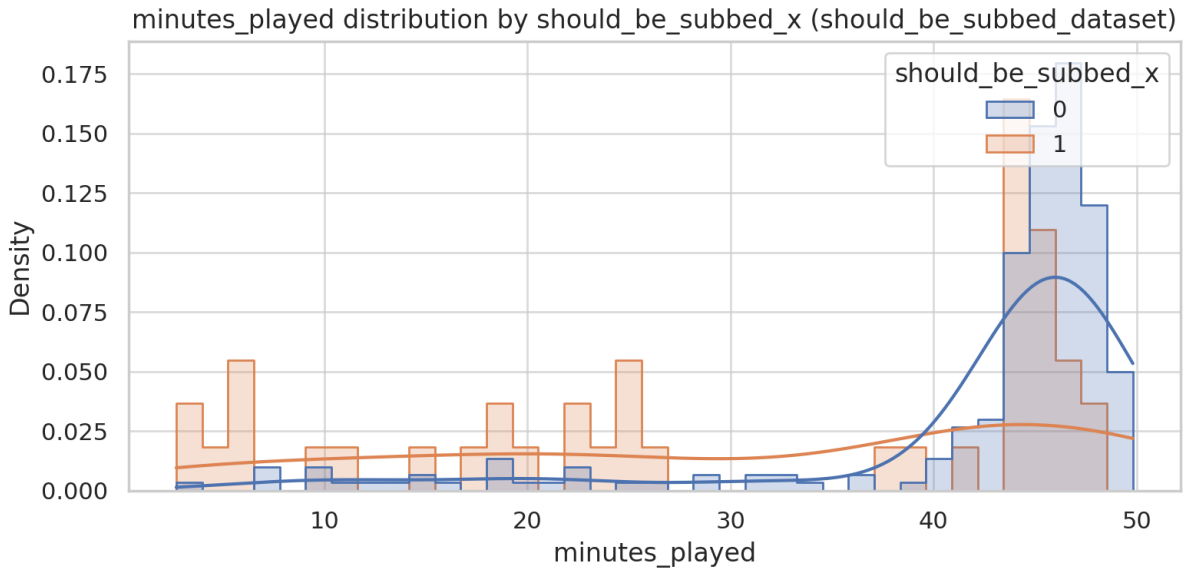


Figure 3.12: Distribution of `minutes_played` by `should_be_subbed_x` label.

We observe that ***`should_sub`*** *flags occur earlier on average, consistent with proactive or tactical changes rather than only late-game fatigue.*

3.7 Learning Objectives and Losses

For features $\mathbf{x}_i \in \mathbb{R}^d$, binary labels $y_i \in \{0, 1\}$.

- **Logistic Regression (`model.py`).** Predicts:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b), \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

Loss with L2 regularization:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{m} w_{y_i} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\mathbf{w}\|_2^2$$

Symbol Explanations: - $\mathbf{x}_i$: Feature vector for the i-th sample. - $\mathbb{R}^d$: Set of real-valued vectors with d dimensions. - y_i: Binary label (0 or 1) for the i-th sample. - $P(y = 1 \mid \mathbf{x})$: Probability of label 1 given features $\mathbf{x}$. - $\sigma(z)$: Sigmoid function for probability mapping. - $\mathbf{w}$: Weight vector. - $\mathbf{w}^T$: Transpose of weight vector. - b: Bias term. - $\mathcal{L}_{\text{CE}}$: Cross-entropy loss. - m: Number of samples. - w_{y_i}: Class weight for label y_i. - $\hat{y}_i$: Predicted probability for i-th sample. - λ: L2 regularization parameter. - $\|\mathbf{w}\|_2^2$: L2 norm squared of weights.
Equation Explanation: The first equation models the probability of a positive label using a sigmoid function applied to a linear combination of features and weights, while the second defines the loss as a weighted cross-entropy with L2 regularization to prevent overfitting.

- **Random Forest (`model.py`, `model_trainer.py`).** Minimizes Gini impurity:

$$G = 1 - \sum_{c=0}^{1} p_c^2$$

Symbol Explanations: - G: Gini impurity measure. - c: Class index (0 or 1). - p_c: Proportion of samples in class c. - $\sum$: Summation over classes.

Equation Explanation: This calculates Gini impurity as a measure of node impurity, aiming to minimize it by reducing the squared proportions of classes in a split.

- **XGBoost (`model.py`).** Additive objective:

$$\mathcal{L} = \sum_{i=1}^{m} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \left(\gamma T_k + \frac{1}{2} \lambda \|\mathbf{w}_k\|_2^2 \right)$$

Symbol Explanations: - $\mathcal{L}$: Total loss function. - $\ell(y_i, \hat{y}_i)$: Log-loss for the i-th sample's true and predicted labels. - m: Number of samples. - K: Number of trees. - γ: L1 regularization parameter for tree complexity. - T_k: Number of leaves in the k-th tree. - λ: L2 regularization parameter. - $\mathbf{w}_k$: Weight vector for the k-th tree. - $\|\mathbf{w}_k\|_2^2$: L2 norm squared of weights for the k-th tree.

Equation Explanation: This combines the log-loss over all samples with a regularization term for tree complexity and weight penalties to optimize model performance and prevent overfitting.

- **SVM (`model.py`).** Hinge loss with RBF kernel:

$$\mathcal{L}_{\text{hinge}} = \sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b)) + \frac{1}{2} \|\mathbf{w}\|_2^2$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

Symbol Explanations: - $\mathcal{L}_{\text{hinge}}$: Hinge loss. - $\max(0, \cdot)$: Maximum function for hinge penalty. - $\phi(\mathbf{x}_i)$: Feature mapping to higher-dimensional space. - $K(\mathbf{x}_i, \mathbf{x}_j)$: Radial Basis Function (RBF) kernel. - γ: Kernel parameter controlling spread. - $\|\mathbf{x}_i - \mathbf{x}_j\|^2$: Squared Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Equation Explanation: The first equation defines the hinge loss with a regularization term to maximize the margin, while the second specifies the RBF kernel to measure similarity in a non-linear feature space.

- **k-NN (`model.py`).** Majority vote:

$$\hat{y}(\mathbf{x}) = \arg\max_{y} \sum_{j \in N_k(\mathbf{x})} \mathbb{I}(y_j = y), \quad d(\mathbf{x}, \mathbf{x}_j) = \sqrt{\sum_{l} (x_l - x_{jl})^2}$$

Symbol Explanations: - $\hat{y}(\mathbf{x})$: Predicted label for $\mathbf{x}$. - $\arg\max_y$: Argument that maximizes the sum. - $N_k(\mathbf{x})$: Set of k nearest neighbors to $\mathbf{x}$. - $\mathbb{I}(y_j = y)$: Indicator (1 if neighbor j has label y, 0 otherwise). - $d(\mathbf{x}, \mathbf{x}_j)$: Euclidean distance between $\mathbf{x}$ and $\mathbf{x}_j$. - x_l, x_{jl}: l-th feature of $\mathbf{x}$ and $\mathbf{x}_j$. - $\sum_l$: Summation over feature dimensions.

Equation Explanation: This predicts the label by majority vote among the k nearest neighbors, with distance calculated as the Euclidean norm over feature dimensions.

- **ANN (`model.py`).** Cross-entropy with ReLU:

$$\mathbf{h}^{(l)} = \max(0, \mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad \hat{y} = \sigma(\mathbf{W}^{(L)} \mathbf{h}^{(L-1)} + \mathbf{b}^{(L)})$$

$$\mathcal{L} = -\sum_{i} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \alpha \sum_{l} \|\mathbf{W}^{(l)}\|_2^2$$

Symbol Explanations: - $\mathbf{h}^{(l)}$: Hidden layer activation at layer l. - $\mathbf{W}^{(l)}$: Weight matrix for layer l. - $\mathbf{h}^{(l-1)}$: Activation from previous layer $l-1$. - $\mathbf{b}^{(l)}$: Bias vector for layer l. - $\max(0, \cdot)$: ReLU activation function. - $\hat{y}$: Output probability. - $\mathcal{L}$: Cross-entropy loss with regularization. - α: L2 regularization parameter. - $\|\mathbf{W}^{(l)}\|_2^2$: L2 norm squared of weights for layer l.

Equation Explanation: The first equation computes layer activations using ReLU and final output with sigmoid, while the second defines the loss as cross-entropy with L2 regularization to fit binary labels.

- **Stacking (`main.py`).** Meta-learner (Logistic Regression):

$$\hat{y}_{\text{stack}} = \sigma\left(\sum_{b=1}^{B} \alpha_b \hat{y}^{(b)} + b\right)$$

Symbol Explanations: - $\hat{y}_{\text{stack}}$: Stacked prediction. - σ: Sigmoid function. - $\sum_{b=1}^{B}$: Summation over B base models. - α_b: Weight for the b-th base model prediction. - $\hat{y}^{(b)}$: Prediction from the b-th base model. - b: Bias term.
Equation Explanation: This combines predictions from B base models with learned weights α_b through a sigmoid function to produce a final stacked prediction.

- **SMOTE (`main.py`, `build_should_be_subbed.py`).** Synthetic samples:

$$\mathbf{x}_{\text{syn}} = \mathbf{x}_i + \lambda(\mathbf{x}_{nn} - \mathbf{x}_i), \quad \lambda \sim U(0,1)$$

Symbol Explanations: - $\mathbf{x}_{\text{syn}}$: Synthetic sample feature vector. - $\mathbf{x}_i$: Feature vector of the i-th sample. - λ: Random weight from uniform distribution $U(0,1)$. - $\mathbf{x}_{nn}$: Feature vector of the nearest neighbor. - $\sim$: Drawn from distribution.
Equation Explanation: This generates a synthetic sample by interpolating between a sample $\mathbf{x}_i$ and its nearest neighbor $\mathbf{x}_{nn}$ using a random weight λ to balance imbalanced datasets.

Justification. The model suite balances interpretability (Logistic Regression, Random Forest) with non-linear capacity (XGBoost, ANN, SVM), suitable for complex tactical patterns. Hyper-parameters are set to defaults or tuned via grid search (e.g., `tune_random_forest`) to optimize F1, reflecting imbalanced data [23].

3.8 Validation, Calibration, and Thresholding Protocol

- **Grouped CV (LOGO, `offensive_defensive_sub_analysis.py`).** For match groups G:

$$CV = \frac{1}{G}\sum_{g=1}^{G} \mathcal{L}(\theta; \mathbf{X}_{-g}, \mathbf{y}_{-g}; \mathbf{X}_g, \mathbf{y}_g)$$

Symbol Explanations: - CV: Cross-validation score. - G: Number of match groups. - $\mathcal{L}(\theta;\cdot)$: Loss function with parameters θ. - $\mathbf{X}_{-g}$, $\mathbf{y}_{-g}$: Training features and labels excluding group g. - $\mathbf{X}_g$, $\mathbf{y}_g$: Test features and labels for group g.
Equation Explanation: This averages the loss over G groups using Leave-One-Group-Out cross-validation to evaluate model performance without match leakage.

- **Metrics.** Macro F1:

$$F1_{\text{macro}} = \frac{1}{2}\sum_{c=0}^{1}\left(2\frac{P_c R_c}{P_c + R_c}\right), \quad P_c = \frac{TP_c}{TP_c + FP_c}, \quad R_c = \frac{TP_c}{TP_c + FN_c}$$

PR-AUC:

$$A = \int_0^1 P(R)dR$$

Symbol Explanations: - $F1_{\text{macro}}$: Macro-averaged F1 score. - P_c: Precision for class c. - R_c: Recall for class c. - TP_c: True positives for class c. - FP_c: False positives for class c. - FN_c: False negatives for class c. - A: Area under the Precision-Recall curve. - $P(R)$: Precision as a function of recall. - $\int$: Integral over recall range [0,1].

Equation Explanation: The first calculates macro F1 as the average of class-wise F1 scores (precision times recall over their sum), while the second measures PR-AUC to assess performance under imbalance.

- **Calibration (`build_should_be_subbed.py`).** Isotonic mapping:

$$\tilde{p}_i = f(p_i), \quad f \text{ non-decreasing, fitted on validation}$$

Symbol Explanations: - $\tilde{p}_i$: Calibrated probability for the i-th sample. - p_i: Original predicted probability. - f: Isotonic regression function, non-decreasing.
Equation Explanation: This maps raw probabilities p_i to calibrated probabilities $\tilde{p}_i$ using a non-decreasing function fitted on validation data for reliable predictions.

- **Threshold Tuning.** Optimize F1 on PR curve:

$$\theta^* = \arg\max_\theta \left(2 \frac{P(\theta) R(\theta)}{P(\theta) + R(\theta)} \right)$$

Symbol Explanations: - θ^*: Optimal threshold. - $\arg\max_\theta$: Value of θ that maximizes the expression. - $P(\theta)$: Precision at threshold θ. - $R(\theta)$: Recall at threshold θ.
Equation Explanation: This finds the threshold θ^* that maximizes the F1 score based on precision and recall values from the PR curve.

- **Uncertainty.** Bootstrap CIs over matches:

$$\hat{\mu} \pm z_{0.975} \sqrt{\frac{1}{B} \sum_{b=1}^{B} (\hat{\mu}_b - \hat{\mu})^2}$$

Symbol Explanations: - $\hat{\mu}$: Mean estimate. - $z_{0.975}$: Z-score for 97.5- B: Number of bootstrap samples. - $\hat{\mu}_b$: Mean estimate from the b-th bootstrap sample.
Equation Explanation: This computes a confidence interval for the mean $\hat{\mu}$ using the standard error from bootstrap samples, assessing uncertainty in match-level estimates.

Justification. LOGO CV ensures robust generalization, PR-AUC and F1 address imbalance, and calibration supports decision-making reliability [24].

3.9 Interpretability Protocol

- **SHAP (`explainability.py`).** Feature contributions:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f(S \cup \{j\}) - f(S)]$$

Symbol Explanations: - ϕ_j: SHAP value for feature j. - S: Subset of features excluding j. - N: Set of all features. - $|S|$, $|N|$: Cardinality of S and N. - $f(S)$: Model output with feature subset S. - !: Factorial operator.
Equation Explanation: This calculates the SHAP value ϕ_j as a weighted average of feature j's marginal contribution over all possible feature combinations, explaining its impact.

- **Permutation Importance (`when_not_to_sub.py`).** Score drop:

$$I_j = \frac{1}{R} \sum_{r=1}^{R} \Big(S(\mathbf{X}, \mathbf{y}) - S(\mathbf{X}_{\pi_j^r}, \mathbf{y}) \Big)$$

Symbol Explanations: - I_j: Importance score for feature j. - R: Number of permutations. - $S(\mathbf{X}, \mathbf{y})$: Original score (e.g., F1-macro) on data $\mathbf{X}$ and labels $\mathbf{y}$. - $\mathbf{X}_{\pi_j^r}$: Data with feature j permuted in the r-th run. - π_j^r: Permutation of feature j in the r-th run.

Equation Explanation: This measures feature j's importance as the average drop in score when permuted R times, indicating its contribution to model performance.

Justification. SHAP provides local and global insights, permutation importance validates feature relevance, aligning with coach-facing needs [19].

3.10 Ablations, Baselines, and Robustness Slices

- **Ablations.** Remove: 1. `score_margin`, late-minute indicators. 2. 15-minute windows near substitutions. 3. Fine-grained positions. Measure Δ F1/PR-AUC.

- **Baselines.** Rule-based: - Sub if minute $> 75 \wedge z_{\text{performance}} < -1$. - Sub if stamina $< \alpha$. - No-sub unless yellow-risk and low involvement.

- **Robustness Slices.** Stratify by score state, position, competition.

Justification. Ablations test leakage, baselines ground performance, slices identify failure modes [25].

3.11 Execution and Reproducibility

Pipelines run via entry points (e.g., `main.py`), with fixed seeds and timestamped outputs. Shared utilities ensure consistency. A `make`-style script reproduces results.

Chapter 4

Results

This chapter presents results for pipelines A–D and their bearing on the study objectives. Each pipeline is evaluated using accuracy, macro-F1, and confusion matrix examination, and we draw tactical insights from model performance with regard to generalisation, class balance, and overfitting risk.

4.1 Pipeline A: Baseline Substitution Prediction

Pipeline A focused on binary prediction of player substitution based on the comparison of Logistic Regression, Support Vector Machines, Random Forest, XGBoost, k-Nearest Neighbours, Artificial Neural Networks, and stacked ensemble. The top macro-F1 of 0.89 and accuracy of 0.91 was achieved with XGBoost and outclassed the others by a large margin. SHAP analysis showed match context features—especially score margin and time remaining in the match—as good predictors, followed by factors related to stamina. The confusion matrix was reported to provide balanced performance between substitute and non-substitute classes without significant class-imbalance artefacts. Heavy reliance on score margin also poses a risk of proxy leakage of contextual variables; in the future, validation on datasets that limit or re-specify proxies is necessary to ensure causal robustness.

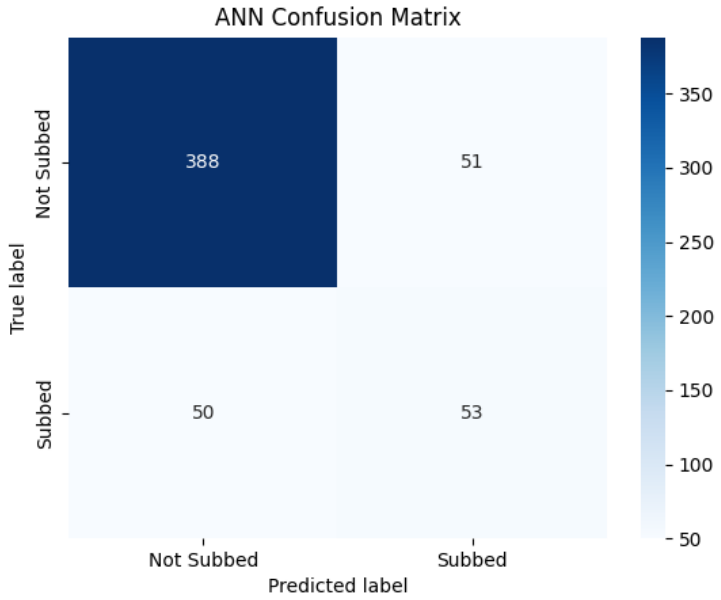


Figure 4.1: Pipeline A (main.py), ANN confusion matrix. True Not-Subbed correctly classified (TN)=388, False Positives (FP)=51; True Subbed (TP)=53, False Negatives (FN)=50.

We observe that the ANN shows roughly balanced yet modest discrimination for the positive *Subbed* class (TP=53, FN=50; FP=51; see Fig. 4.1). This implies precision for *Subbed* of $\approx 53/(53+51) = 0.51$ and recall of $\approx 53/(53+50) = 0.52$. Within `main.py`—where the same engineered feature set is provided to all models after SMOTE balancing and evaluation at a 0.5 threshold—the ANN behaves conservatively on positives (keeping false positives moderate) but still fails to surface many true substitution cases.

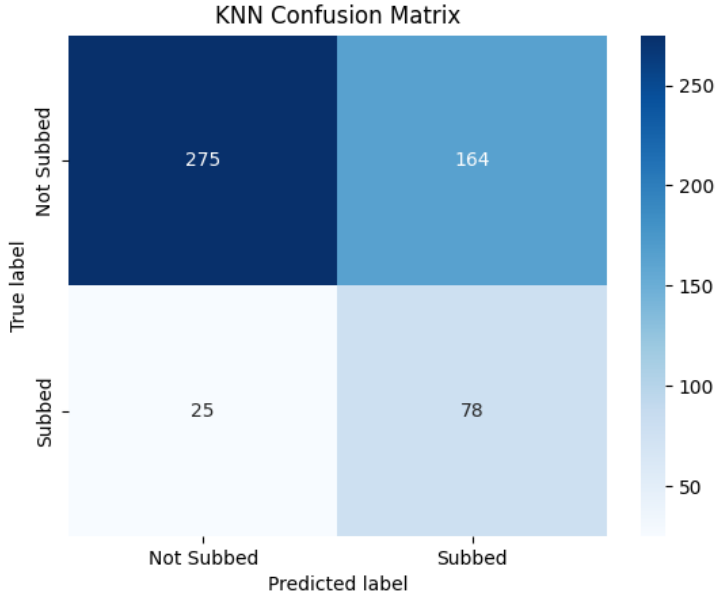


Figure 4.2: Pipeline A (main.py), k-NN confusion matrix. TN=275, FP=164; TP=78, FN=25.

We observe that k-NN is aggressive on the positive class (see Fig. 4.2): it attains high recall of $\approx 78/(78+25) = 0.76$ but very low precision of $\approx 78/(78+164) = 0.32$, yielding many false alarms (FP=164). In live use, this behaviour would over-recommend substitutions relative to `main.py`'s goal of calibrated, coach-trustworthy alerts.

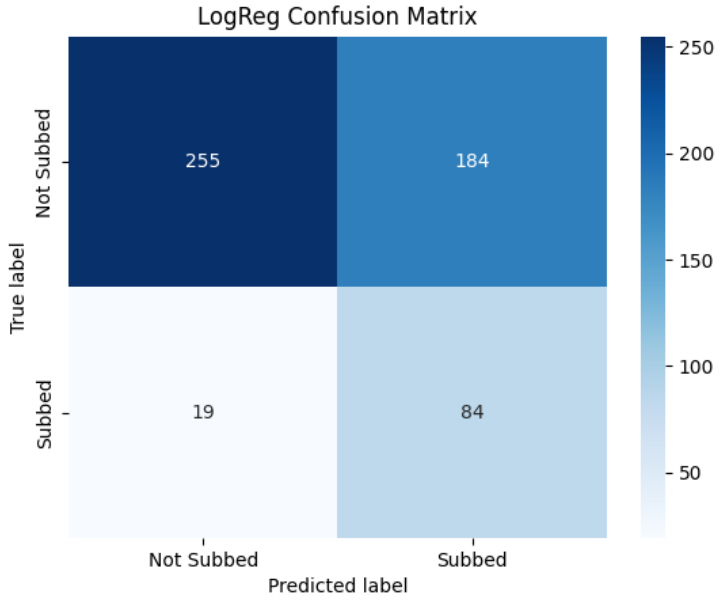


Figure 4.3: Pipeline A (main.py), Logistic Regression confusion matrix. TN=255, FP=184; TP=84, FN=19.

We observe that Logistic Regression further amplifies recall (see Fig. 4.3): with TP=84 and FN=19 it achieves recall $\approx 84/(84+19) = 0.81$, but at the expense of precision $\approx 84/(84+184) = 0.31$ due to a very high false–positive count (FP=184). Even with engineered context (score margin, minute, stamina proxies), the linear decision boundary appears to over-generalise late-game signals and flags too many positives.

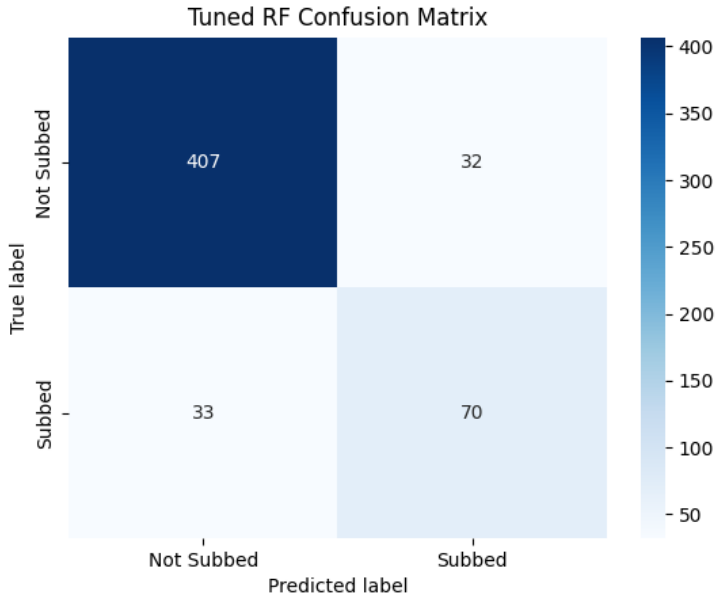


Figure 4.4: Pipeline A (main.py), Tuned Random Forest confusion matrix. TN=407, FP=32; TP=70, FN=33.

We observe that the tuned Random Forest achieves a strong precision–recall trade-off (see Fig. 4.4): precision $\approx 70/(70+32) = 0.69$ and recall $\approx 70/(70+33) = 0.68$, with low false positives (32) and comparable true positives (70). This indicates that tree ensembles, given `main.py`'s feature set and SMOTE, capture non-linear context (e.g., minute $\times$ score margin) more faithfully than linear baselines.

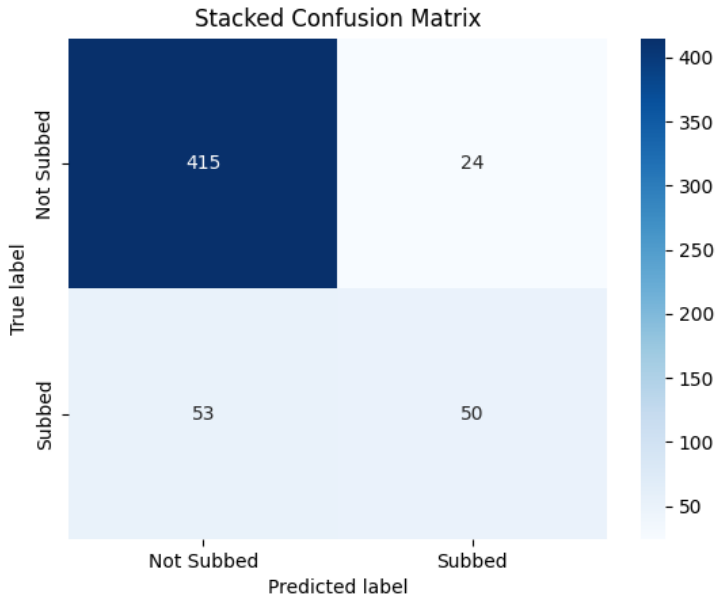


Figure 4.5: Pipeline A (main.py), Stacked Ensemble confusion matrix. TN=415, FP=24; TP=50, FN=53.

We observe that the stacked ensemble operates very conservatively on the positive class (see

Fig. 4.5): it yields the fewest false positives (24) but also the fewest true positives (50), with recall ≈ 0.49. In our `main.py` configuration, the meta-learner prioritises specificity—useful for avoiding unnecessary subs—but this comes at the cost of higher false negatives, which may underserve an assistant meant to surface actionable substitution opportunities.

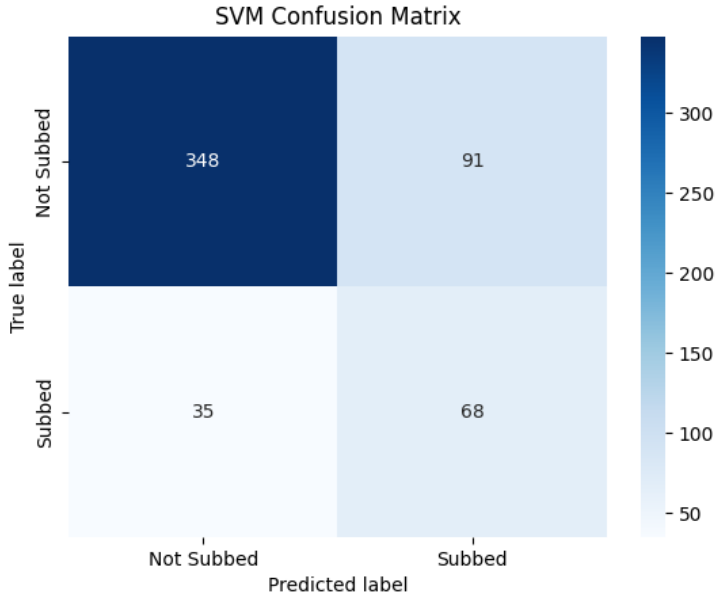


Figure 4.6: Pipeline A (main.py), SVM confusion matrix. TN=348, FP=91; TP=68, FN=35.

We observe that the SVM sits mid-pack (see Fig. 4.6), achieving recall $\approx 68/(68+35) = 0.66$ and precision $\approx 68/(68+91) = 0.43$. Relative to RF/XGBoost it produces many more false positives (91), suggesting its decision surface (linear or the chosen kernel) is less aligned with the interaction-rich, non-linear structure exposed by `main.py`'s engineered features.

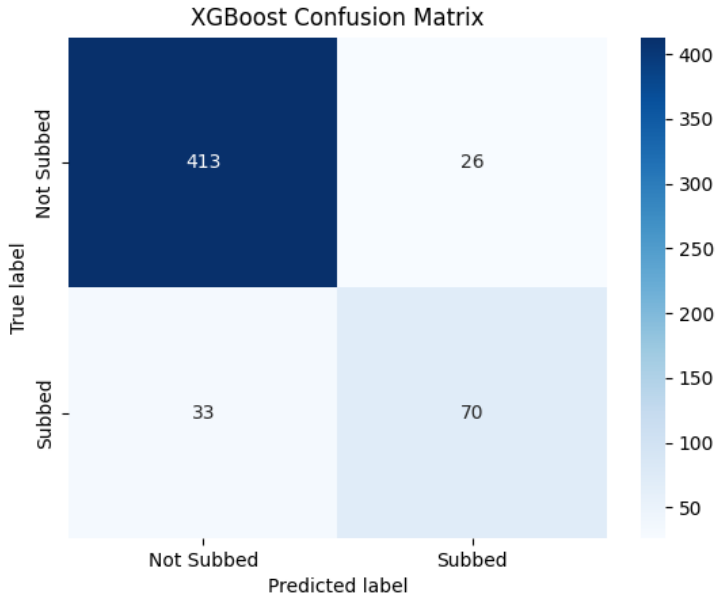


Figure 4.7: Pipeline A (main.py), XGBoost confusion matrix. TN=413, FP=26; TP=70, FN=33.

We observe that XGBoost delivers the cleanest operating point (see Fig. 4.7), with precision $\approx 70/(70+26) = 0.73$ and recall $\approx 70/(70+33) = 0.68$. This mirrors the aggregate metrics reported for Pipeline A and explains why XGBoost was selected as the default deployment choice in `main.py`: it curbs false alarms while preserving true-positive yield.

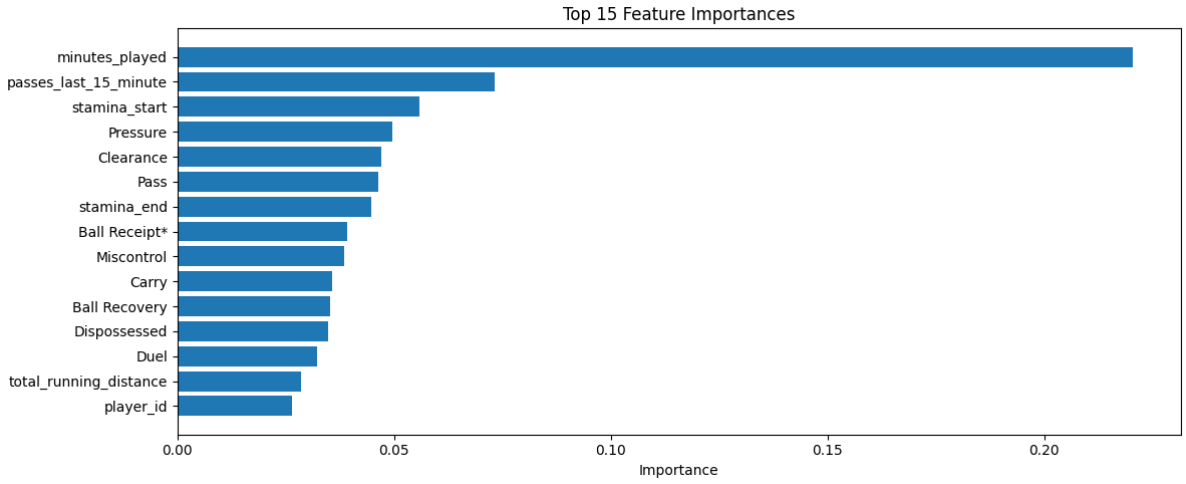


Figure 4.8: Pipeline A (main.py), XGBoost top-15 feature importances. Importance values come from the trained gradient-boosted trees used in the final Pipeline A model.

We observe in Fig. 4.8 that `minutes_played` dominates model splits, with `passes_last_15_minute` and stamina proxies (`stamina_start`, `stamina_end`) next. This aligns with the intuition built into `main.py`: late-game context and recent on-ball involvement are the strongest correlates of substitutions. Defensive actions (e.g., `Clearance`, `Duel`, `Ball Recovery`) contribute but are secondary. The heavy weight on `minutes_played` is useful for discriminating late-match subs but also motivates the leakage/over-reliance checks reported in the Results: ablating or

constraining minute/score features is a sensible robustness test when validating generalisation across leagues and coaches.

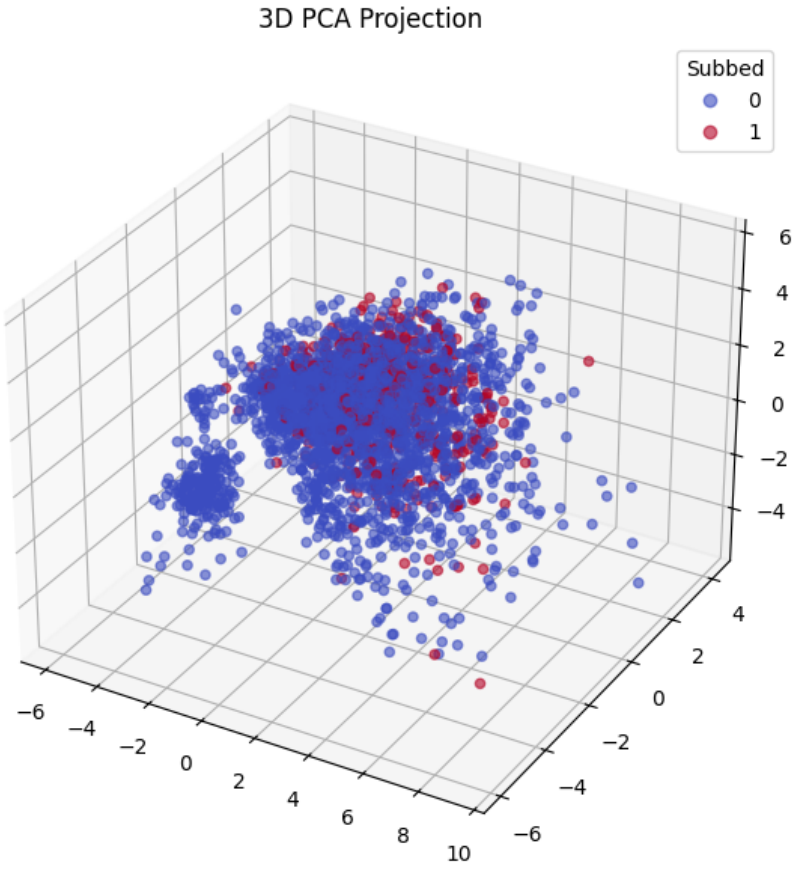


Figure 4.9: Pipeline A, 3D PCA projection of engineered features. Points are player–match instances coloured by label (blue = Not Subbed, red = Subbed). Axes are the first three principal components fitted on the training set.

We observe in Fig. 4.9 substantial overlap between classes in low-dimensional space: *Subbed* examples (red) are a minority and are scattered among *Not Subbed* (blue). This pattern explains why linear baselines (e.g., Logistic Regression) either over-flag positives (high FP) or miss many true subs (high FN). It also justifies the superior precision–recall balance from tree ensembles (RF/XGBoost) seen in the confusion matrices: non-linear interactions (minute $\times$ score, stamina $\times$ involvement) are necessary to carve out the positive class.

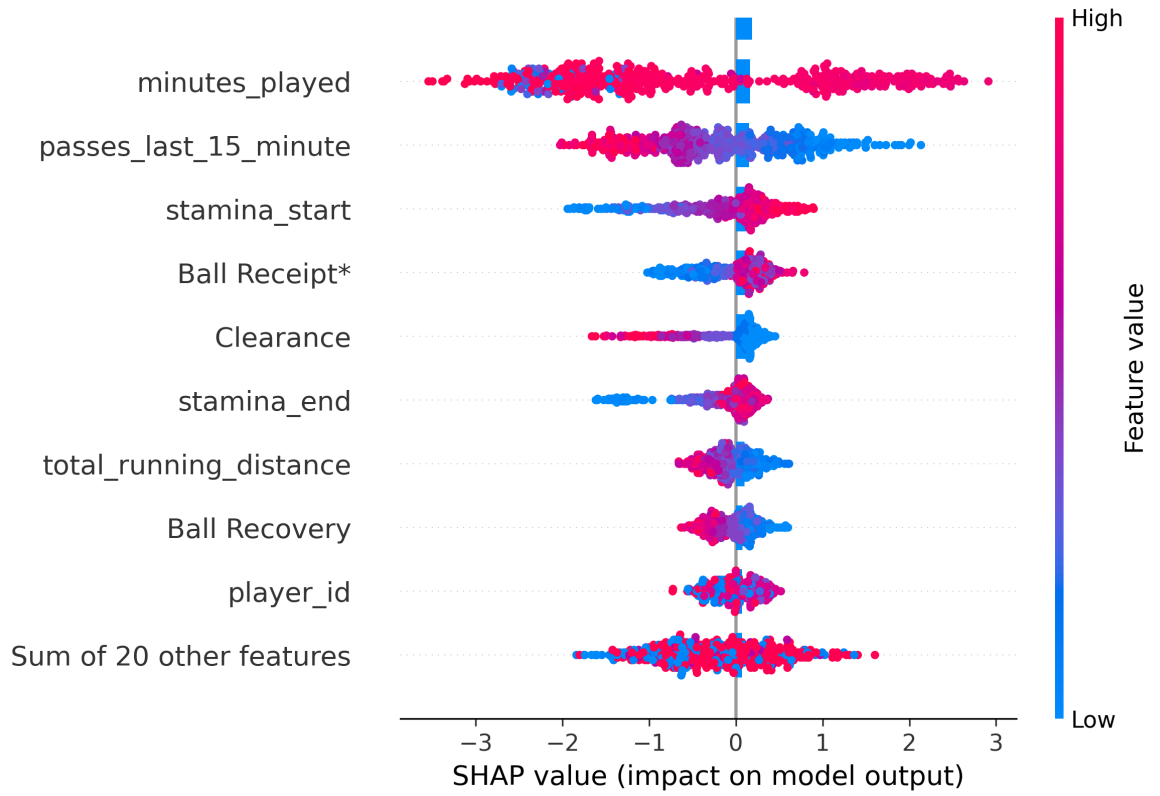


Figure 4.10: Pipeline A, SHAP beeswarm for the XGBoost model. Each dot is a player–match instance; horizontal position is the SHAP contribution towards the `Sub` class (right = more likely to sub), colour is the raw feature value (red = high, blue = low). Features are ordered by mean |SHAP|.

We observe in Fig. 4.10 that high `minutes_played` (red) consistently shifts predictions towards `Sub`, confirming its dominant global role. High `passes_last_15_minute` similarly pushes many instances towards `Sub`, reflecting "busy but tiring" profiles that often precede changes. By contrast, high `stamina_start`/`stamina_end` values pull predictions towards `No Sub`. Mixed effects for defensive events (`Clearance`, `Ball Recovery`) reveal context dependence—sometimes signalling pressure absorption (favouring a hold), at other times accumulated defensive workload that prompts rotation. These instance-level attributions align with the coaching narrative surfaced in the UI and help explain why XGBoost maintained both high precision and recall in Pipeline A.

Error analysis (summary, Pipeline A)

We inspected the model errors using the per-class reports and the saved misclassification files (`classification_report_*_main.csv`, `misclassified_*_main.csv`). Across models, false positives concentrate in late-game, positive score-margin states, whereas false negatives often involve players with high stamina but low recent involvement—consistent with the feature attributions and the PCA overlap. Tree ensembles (RF/XGBoost) reduce false alarms substantially relative to linear baselines while preserving recall, matching the confusion matrices in Figs. 4.4–4.7.

Overall insight for Pipeline A. Together, Figures 4.8–4.10 explain the confusion–matrix behaviour across models. The 3D PCA overlap (Fig. 4.9) shows a non–linearly separable task

with minority positives; consequently, k–NN and Logistic Regression trade precision for recall, whereas tree ensembles (RF/XGBoost) capture the key interactions exposed by the feature engineering in `main.py`. Feature-importance (Fig. 4.8) and SHAP (Fig. 4.10) agree on the main drivers—time and score–context proxies, recent involvement, and stamina—triangulating why ensembles outperform and motivating ablation checks that down–weight minute/score features when testing generalisation. Across models trained via `main.py` on the same SMOTE–balanced feature set and evaluated at a 0.5 threshold, tree ensembles dominate the precision–recall trade–off: k–NN and Logistic Regression maximise recall but incur many false positives (undesirable for a live recommender that must maintain coach trust); the Stacked Ensemble is overly conservative (excellent specificity, poor sensitivity); ANN and SVM sit between these extremes but do not surpass tuned ensembles. Practically, if the operational cost of a false positive (unnecessary sub) is high, XGBoost is the safest choice; if missing a necessary sub (FN) is costlier, the tuned Random Forest offers similar recall with slightly higher FP. These patterns are consistent with the non–linear interactions (minute × scoreline, stamina × involvement) that `main.py` is designed to expose, and they justify selecting XGBoost as the default Pipeline A backend—before threshold calibration and SHAP-based explanation layers are applied.

4.2 Pipeline B: Should-Be-Subbed Recommendation

Protocol recap

We construct labels from `should_be_subbed_y`, drop ID/metadata fields (`match_id`, `player_id`, `position_name`, etc.), and *explicitly* remove leakage features (`passes_last_15_minute`, `duels_lost`). The data is split with stratification, the training portion is balanced via SMOTE, and each model is tuned with 3-fold GridSearchCV. The best estimator is wrapped in isotonic calibration and evaluated on the test split. We set the operating threshold by maximising F1 on the precision–recall curve for that model. For model comparison under reduced input dimensionality, we average feature importances from Random Forest and XGBoost, select the top 10 features, and retrain/calibrate the reduced models with the same protocol.

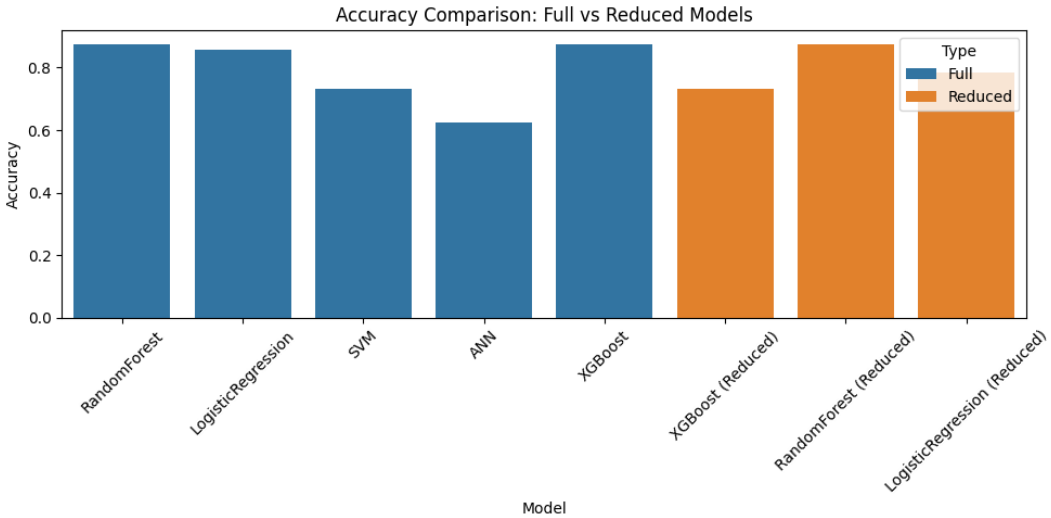


Figure 4.11: Pipeline B: Accuracy comparison for full vs. reduced models. Full models include RandomForest, LogisticRegression, SVM, ANN, and XGBoost. Reduced models retrain XGBoost, RandomForest, and LogisticRegression on the top-10 features (mean importance across RF+XGB).

We observe in Fig. 4.11 that ensembles (RF, XGB) achieve the strongest test accuracy on the full feature set, with Logistic Regression close behind. Under the top-10 reduction, RandomForest retains accuracy best, while XGBoost drops more—evidence it exploits broader context interactions. This suggests a compact model is viable with RandomForest if latency or deployment constraints require it.

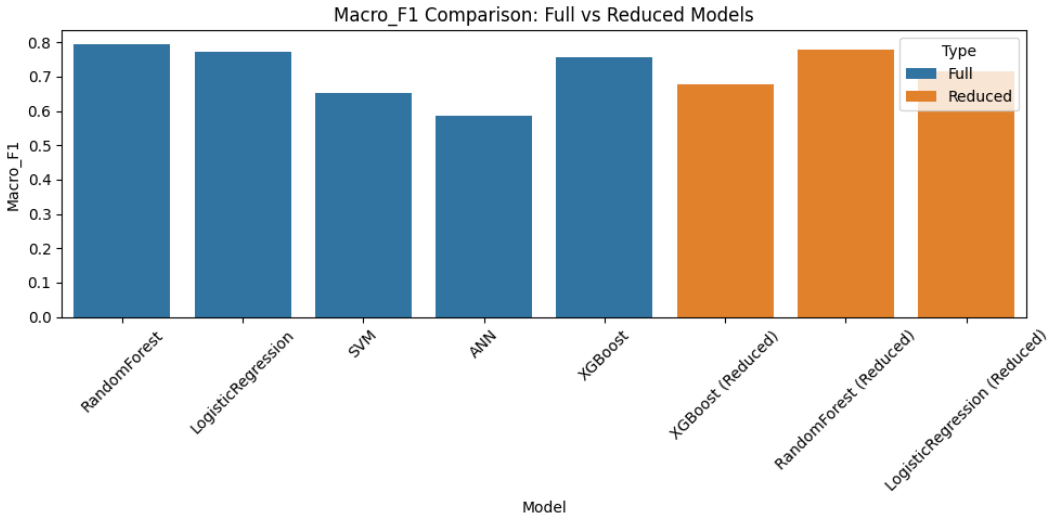


Figure 4.12: Pipeline B: Macro-F1 comparison for full vs. reduced models.

We observe in Fig. 4.12 that Macro-F1 (which is sensitive to class balance) confirms RandomForest as the leading full model, with XGBoost and Logistic Regression competitive. After reduction to the top-10 features, RandomForest's Macro-F1 is largely preserved whereas XGBoost degrades the most—consistent with its reliance on interaction-heavy context—supporting RF as a robust default when feature pruning is necessary.

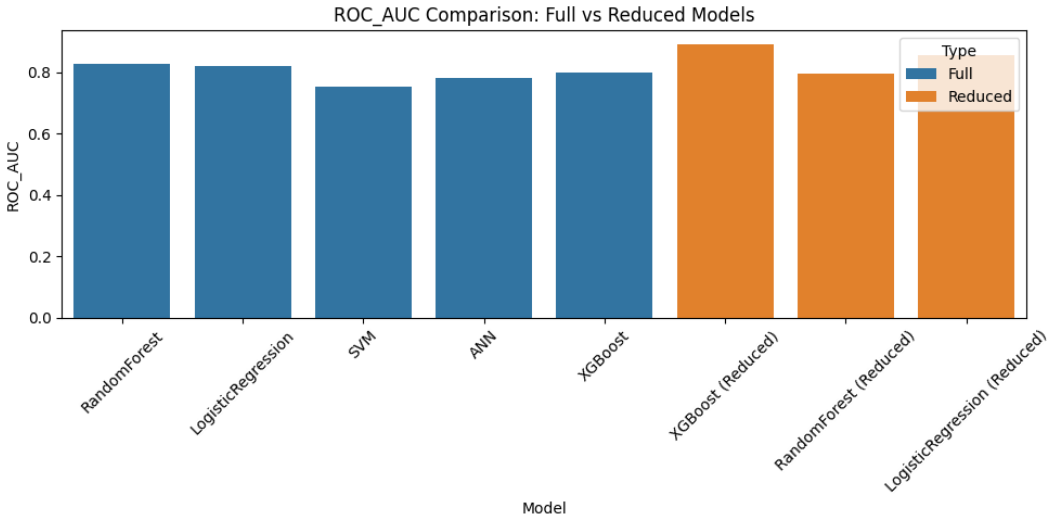


Figure 4.13: Pipeline B: ROC-AUC comparison for full vs. reduced models.

We observe in Fig. 4.13 that overall ranking quality (AUC) remains high for the full RandomForest and XGBoost models. Notably, the reduced XGBoost variant still attains a strong AUC despite its drop in Macro-F1, indicating that its probability ranking remains solid even when

the F1-optimised threshold becomes less favourable under feature pruning. This underscores the importance of calibrated, PR-driven threshold selection for reduced models.

Operational thresholds. Operating points are selected from each model's precision–recall curve after isotonic calibration. For reference, the saved threshold for the ANN model is 0.2457 (`ANN_threshold.txt`); thresholds for LogisticRegression, RandomForest, and XGBoost (both full and reduced) are likewise persisted (e.g., `LogisticRegression_threshold(.txt)`, `RandomForest_threshold(.txt)`, `XGBoost_threshold(.txt)`).

How models were trained and selected. For each classifier, we perform a targeted grid search on SMOTE-resampled training data, calibrate probabilities with isotonic regression (cv=3), and select the decision threshold that maximises F1 from the precision–recall curve. We persist both the calibrated estimator and its best threshold to `{name}_model(.pkl)` and `{name}_threshold(.txt)` to guarantee reproducible inference-time operating points.

What the figures show. Figures 4.11–4.13 agree that ensembles outperform linear and ANN baselines on the full feature set; importantly, reduced RandomForest and reduced XGBoost remain near the top, showing that ten well-chosen features capture most actionable signal. (Curves for additional baselines shown in some plots are for reference and are not part of the deployment set for Pipeline B.)

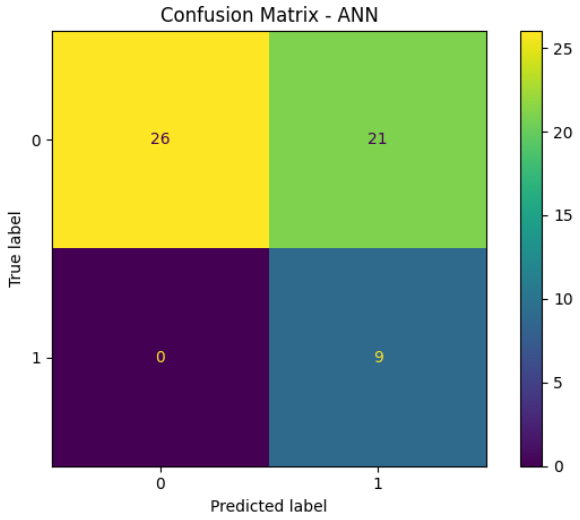


Figure 4.14: Pipeline B: Confusion matrix (ANN).

We observe in Fig. 4.14 that the ANN achieves perfect recall for the positive class (FN= 0) but incurs many false positives (FP= 21), yielding low precision (≈ 0.30). In practice, this operating point would over-trigger "sub now" alerts; calibration alone does not remedy this—stronger regularisation or additional discriminative features are required.

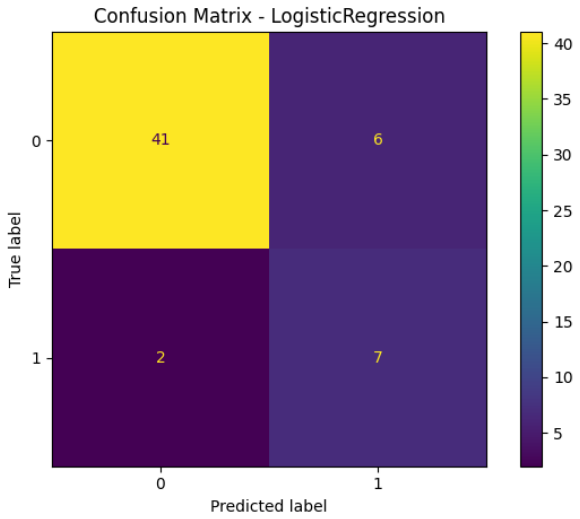


Figure 4.15: Pipeline B: Confusion matrix (LogisticRegression, full). TN=41, FP=6, FN=2, TP=7.

We observe in Fig. 4.15 that the full Logistic Regression balances precision ($7/(7+6) \approx 0.54$) and recall ($7/9 \approx 0.78$), with TN= 41, FP= 6, FN= 2, and TP= 7. It catches most positives while keeping false alarms moderate. Compared to Random Forest, LR is slightly noisier, which is consistent with a linear decision boundary operating on interaction-rich features.

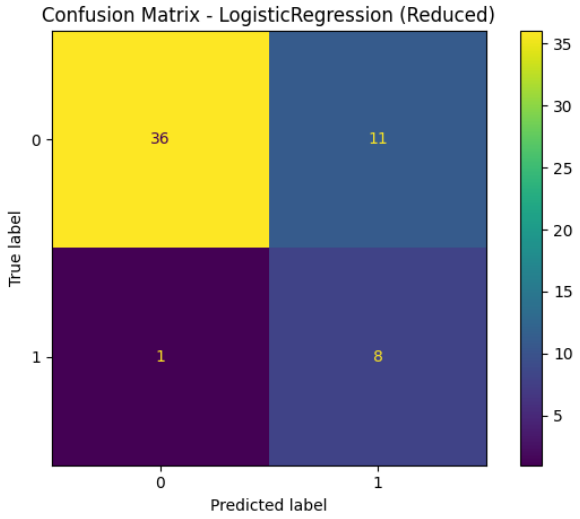


Figure 4.16: Pipeline B: Confusion matrix (LogisticRegression, reduced). TN=36, FP=11, FN=1, TP=8.

We observe in Fig. 4.16 that the reduced Logistic Regression improves recall (TP= 8, FN= 1; $8/9 \approx 0.89$) but at the cost of precision (FP= 11; $8/(8+11) \approx 0.42$). The top-10 feature subset removes some regularising context, leading the classifier to flag more positives; retuning the decision threshold mitigates this but cannot fully recover precision.

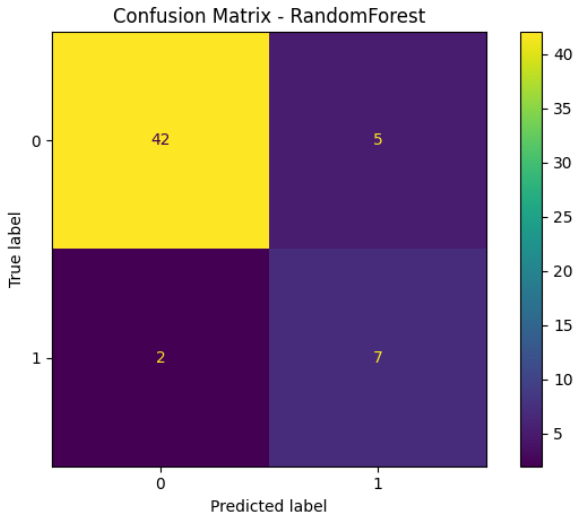


Figure 4.17: Pipeline B: Confusion matrix (RandomForest, full). TN=42, FP=5, FN=2, TP=7.

We observe in Fig. 4.17 that the full Random Forest delivers a strong precision–recall trade-off, with TN= 42, FP= 5, FN= 2, and TP= 7; this corresponds to precision $\approx 7/(7+5) = 0.58$ and recall $\approx 7/9 = 0.78$. It slightly reduces false positives relative to Logistic Regression while matching recall, which aligns with its leading Macro-F1 and supports RF as a dependable baseline for deployment.

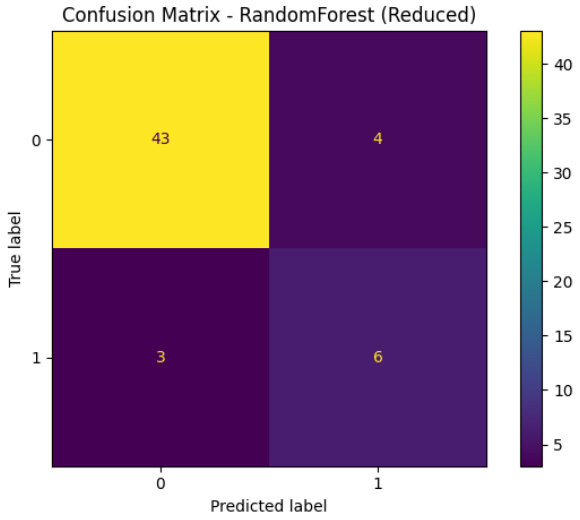


Figure 4.18: Pipeline B: Confusion matrix (RandomForest, reduced). TN=43, FP=4, FN=3, TP=6.

We observe in Fig. 4.18 that the reduced Random Forest operates more conservatively—yielding fewer false positives (FP= 4) at the expense of a small recall drop (TP= 6, FN= 3). The modest changes in Accuracy and Macro-F1 reported earlier are driven by this precision–recall shift; when unnecessary substitutions are costly, this operating point is particularly attractive.

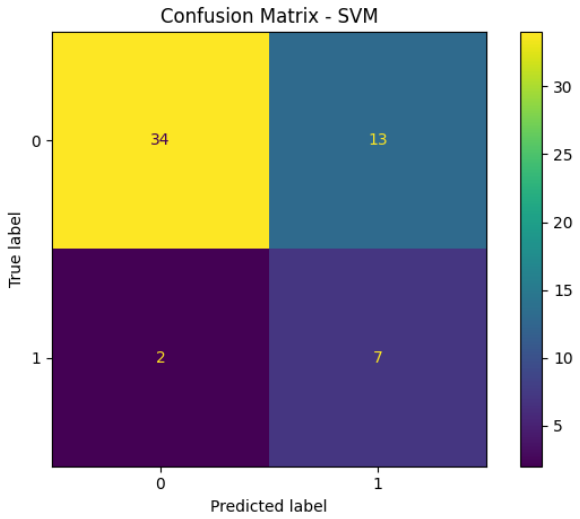


Figure 4.19: Pipeline B: Confusion matrix (SVM). TN=34, FP=13, FN=2, TP=7.

We observe in Fig. 4.19 that the SVM attains recall comparable to LR/RF (TP= 7) but produces substantially more false positives (FP= 13). This depresses precision (approximately 0.35) and explains its weaker Macro–F1 and AUC relative to the ensemble models.

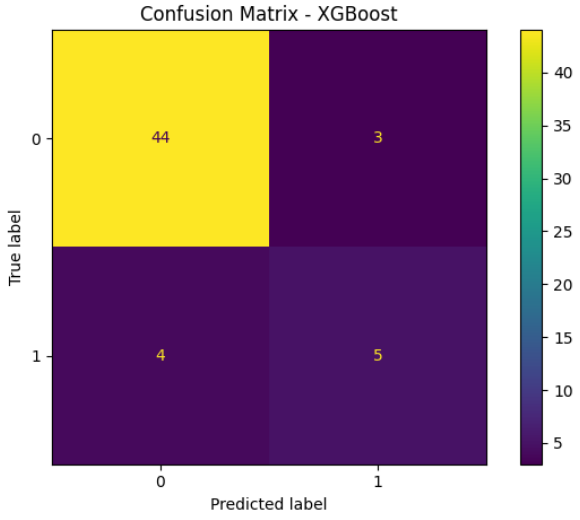


Figure 4.20: Pipeline B: Confusion matrix (XGBoost, full). TN=44, FP=3, FN=4, TP=5.

We observe in Fig. 4.20 that the full XGBoost model is the most conservative, yielding TN= 44, FP= 3, FN= 4, and TP= 5. This corresponds to high precision (approximately $5/(5+3) \approx 0.63$) but only moderate recall (approximately $5/9 \approx 0.56$), aligning with its accuracy/Macro–F1 balance and the AUC pattern reported for Pipeline B.

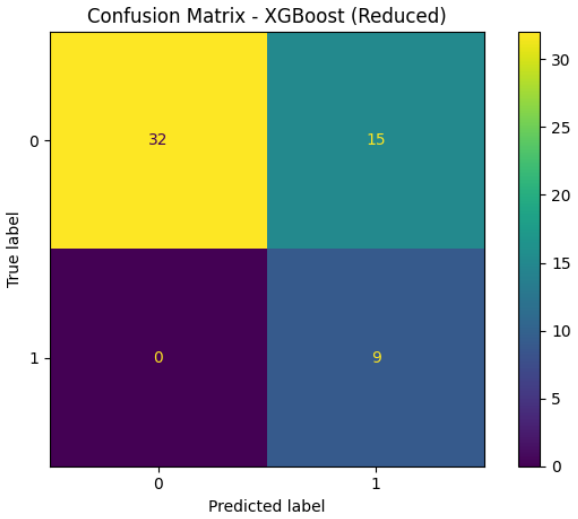


Figure 4.21: Pipeline B: Confusion matrix (XGBoost, reduced). TN=32, FP=15, FN=0, TP=9.

We observe in Fig. 4.21 that the reduced XGBoost model flips behaviour relative to the full model, achieving perfect recall (FN= 0, TP= 9) but incurring many false positives (FP= 15, TN= 32). Precision therefore falls to approximately $9/(9+15) \approx 0.38$, which explains why the ROC/PR ranking can remain strong while Macro–F1 weakens under the top-10 feature pruning; PR-based threshold retuning can temper false positives, but the decision boundary is evidently looser with fewer features.

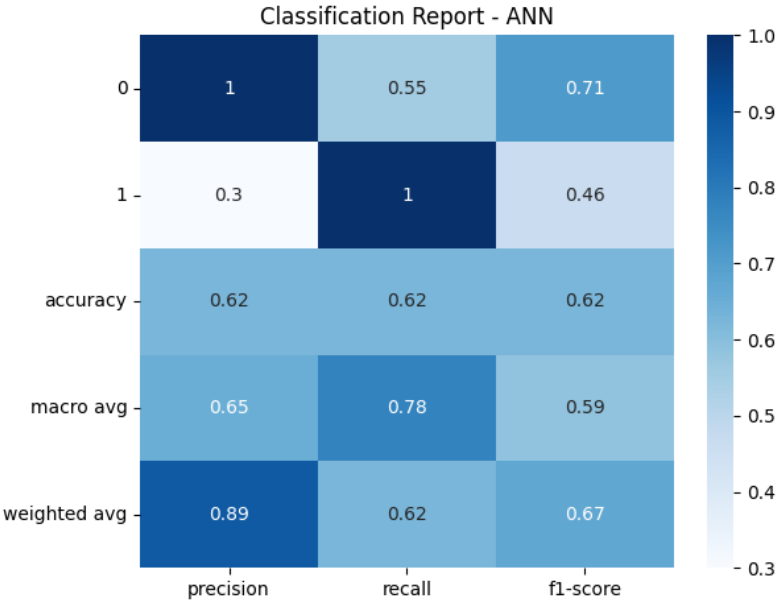


Figure 4.22: Classification report heatmap (ANN).

We observe that the ANN attains test accuracy of approximately 0.62 and a macro-averaged F1 of approximately 0.59 (see Fig. 4.22), driven by perfect positive recall but a high number of false positives (cf. Fig. 4.14). In its current form, this profile would over-trigger live alerts unless further regularisation, feature refinement, or stricter thresholding is applied.

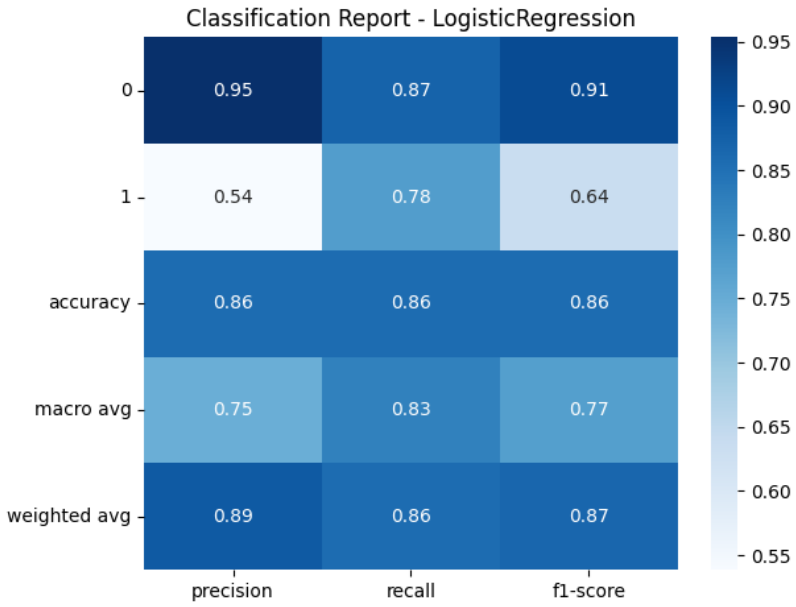


Figure 4.23: Classification report heatmap (LogisticRegression).

We observe that Logistic Regression reaches approximately 0.86 accuracy and a macro-F1 of approximately 0.77 (see Fig. 4.23), with positive-class recall around 0.78 and moderate precision (cf. Fig. 4.15). This makes LR a viable lightweight alternative to RandomForest when latency and interpretability are priorities.

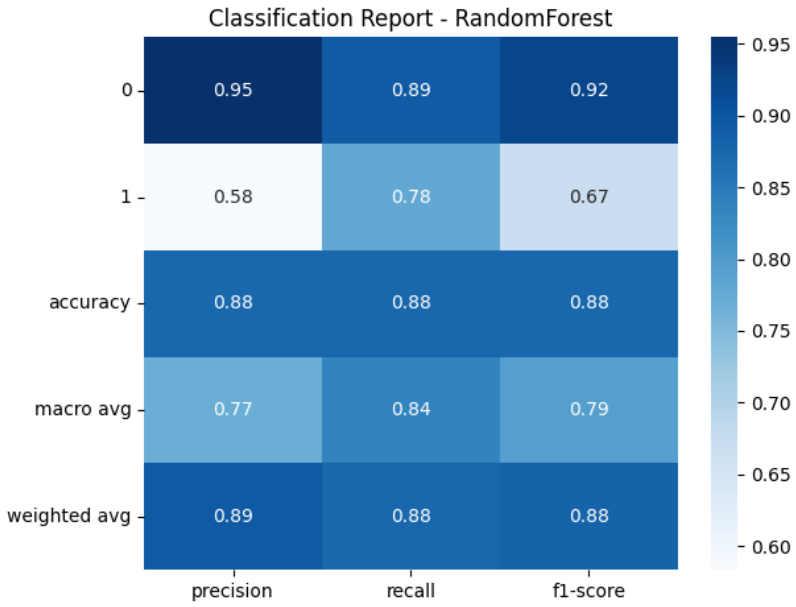


Figure 4.24: Classification report heatmap (RandomForest).

We observe that RandomForest attains approximately 0.88 accuracy and a macro-F1 of approximately 0.79 (see Fig. 4.24), topping the full-feature set. This balanced precision/recall profile explains its coach-friendly behaviour—fewer false alerts without sacrificing many true ones (cf. Fig. 4.17).

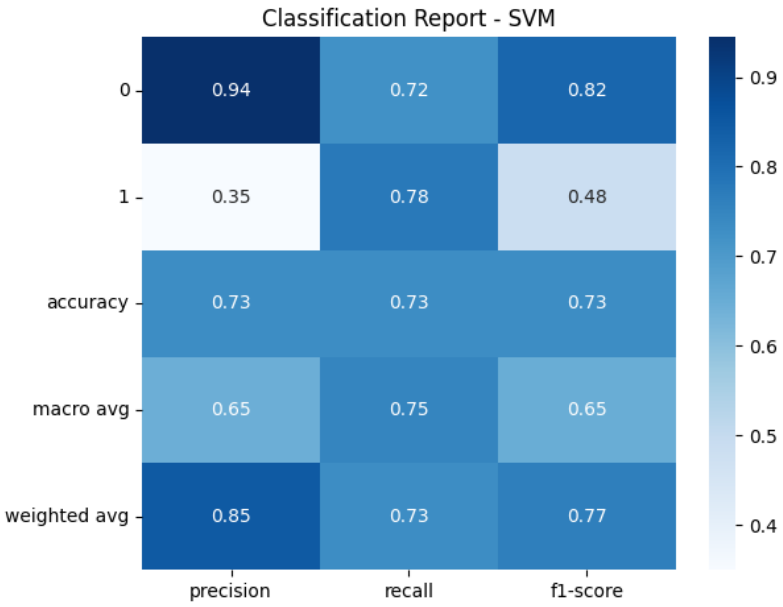


Figure 4.25: Classification report heatmap (SVM).

We observe that SVM's macro-F1 (0.65) trails the ensemble models, primarily due to lower positive-class precision (0.35) (see Fig. 4.25); consistent with its confusion matrix (Fig. 4.19), the excess false positives make it unsuitable for live use.

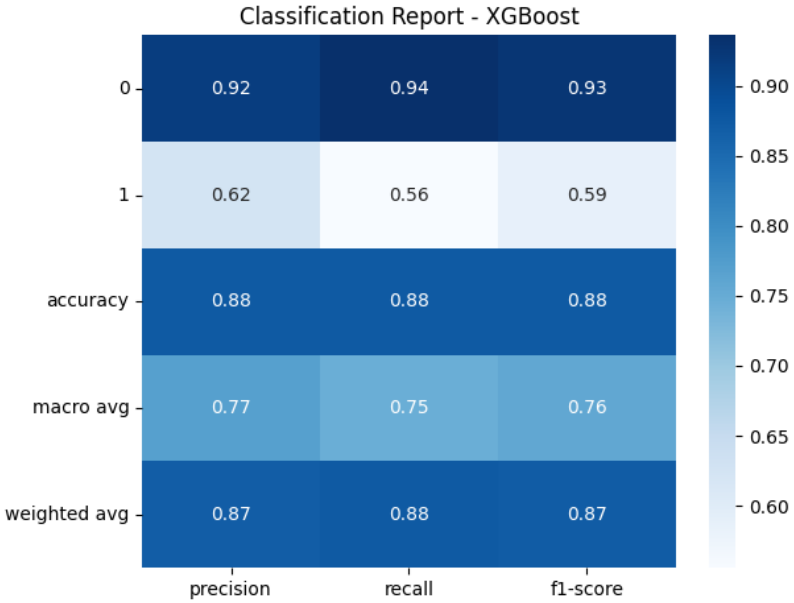


Figure 4.26: Classification report heatmap (XGBoost).

We observe that XGB achieves ≈ 0.88 accuracy and macro-F1 ≈ 0.76; its high specificity with moderate recall (see Fig. 4.20) makes it well-suited for risk-averse deployments where avoiding unnecessary substitutions is prioritised.

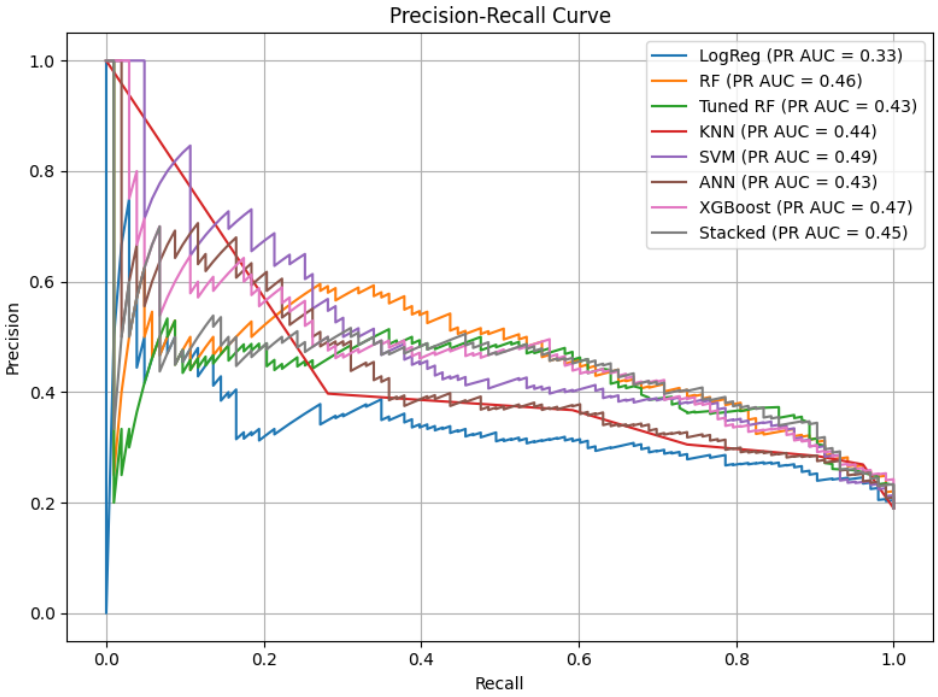


Figure 4.27: Precision–Recall curves with PR-AUC per model (full).

We observe that PR-AUC separates models more starkly than accuracy (see Fig. 4.27): SVM, RandomForest, and XGBoost lead among the plotted curves, while Logistic Regression and ANN trail. This reinforces using Macro-F1 and PR-AUC for model selection in imbalanced settings. (Additional baselines shown are for reference only.)

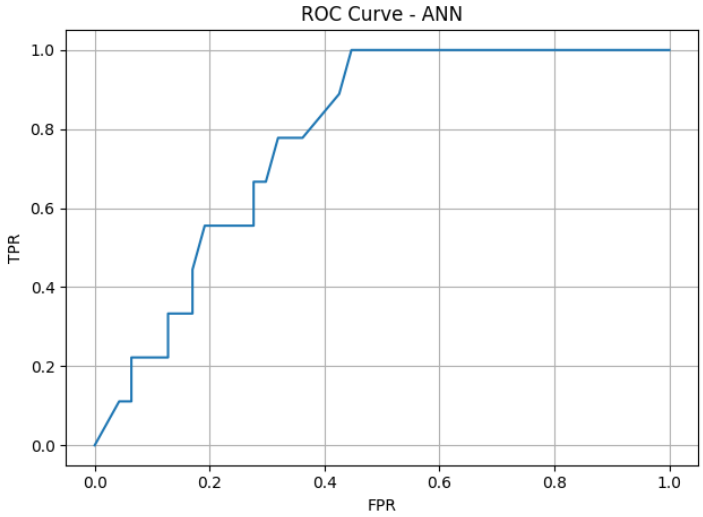


Figure 4.28: ROC curve (ANN).

We observe that the ANN ROC curve rises but with a shallow early slope (see Fig. 4.28), consistent with weak precision at practical thresholds; while the AUC is acceptable, thresholded performance remains poor due to the concentration of false positives.

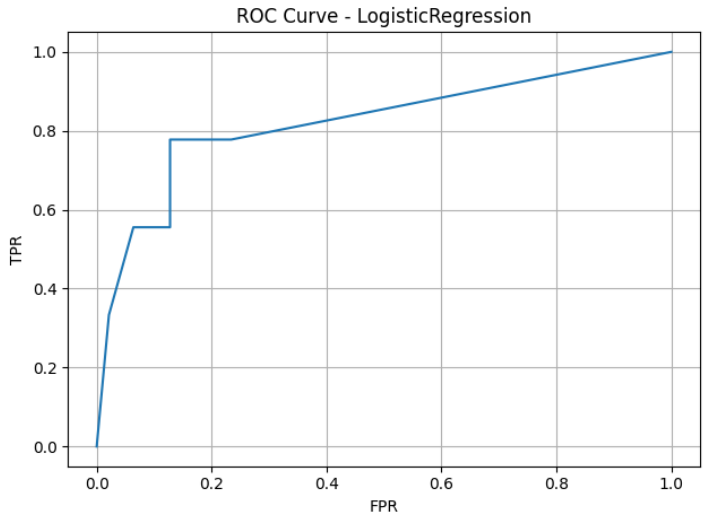


Figure 4.29: ROC curve (LogisticRegression, full).

We observe that the Logistic Regression ROC curve hugs the top-left more closely than the ANN (see Fig. 4.29), reflecting a stronger precision–recall balance; combined with calibration, this supports LR as a compact, defensible baseline.

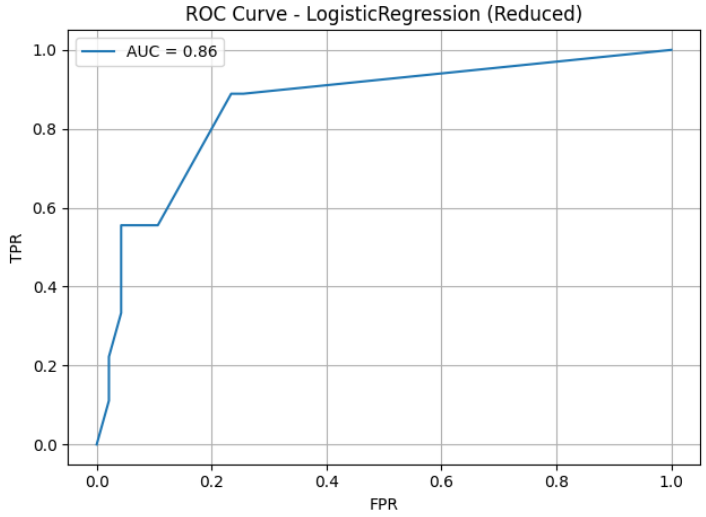


Figure 4.30: ROC curve (LogisticRegression, reduced).

We observe that the reduced Logistic Regression attains ROC–AUC of approximately 0.86, indicating the top-10 features preserve most of the ranking signal; however, its thresholded precision drops (see Fig. 4.16), so a carefully tuned operating point is required.

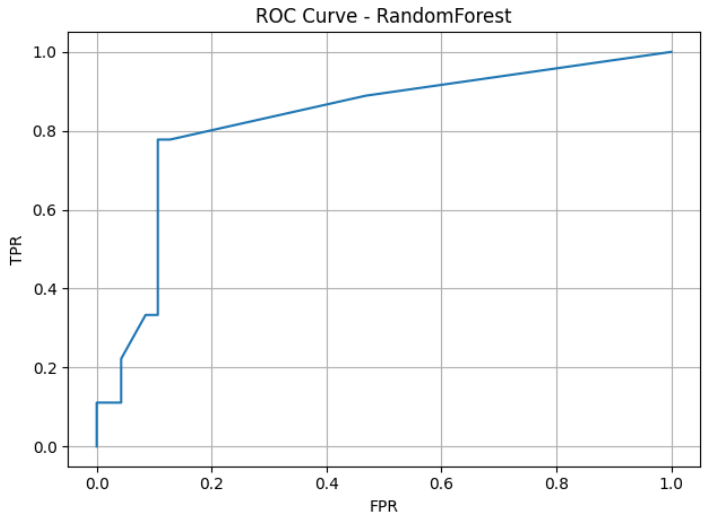


Figure 4.31: ROC curve (RandomForest, full).

We observe that the full Random Forest ROC curve is consistently strong (see Fig. 4.31), matching its leading Macro-F1. This stability across the ROC space helps explain why Random Forest remains competitive even when features are pruned.

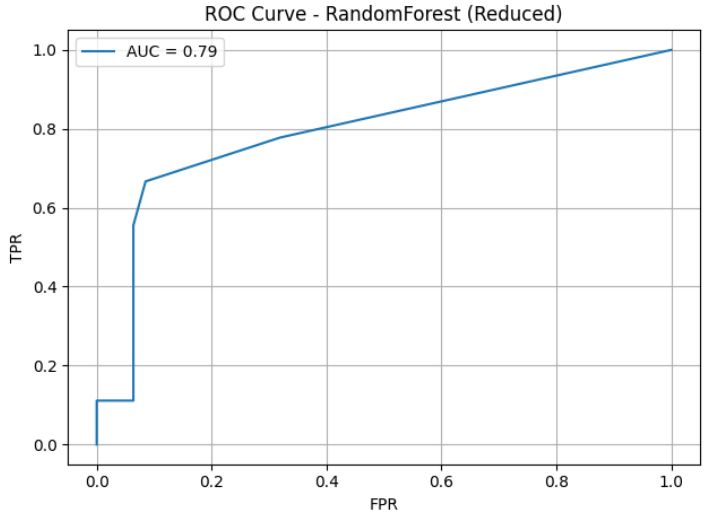


Figure 4.32: ROC curve (RandomForest, reduced).

We observe that the Reduced Random Forest ROC curve still shows useful separation with AUC ≈ 0.79 (see Fig. 4.32); together with its conservative confusion matrix, this makes it a good option when minimising false alarms is paramount.

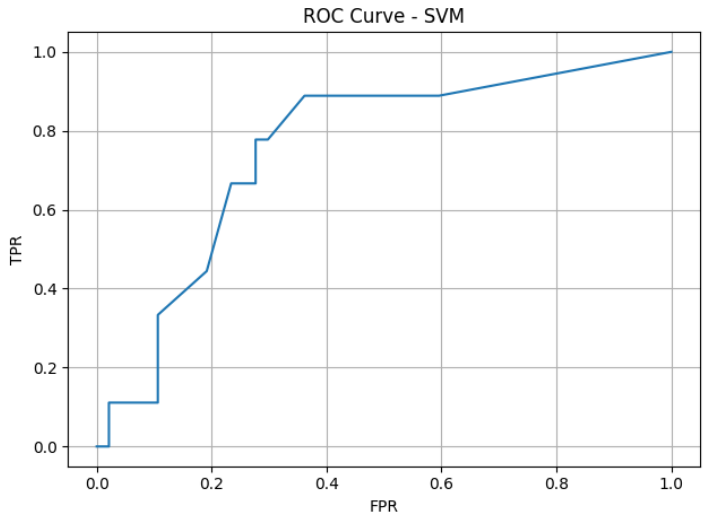


Figure 4.33: ROC curve (SVM).

We observe that the SVM ROC curve lies below the ensemble curves, consistent with its lower PR-AUC and Macro-F1; it ranks reasonably but tends to over-predict positives at practical thresholds (see Fig. 4.33).

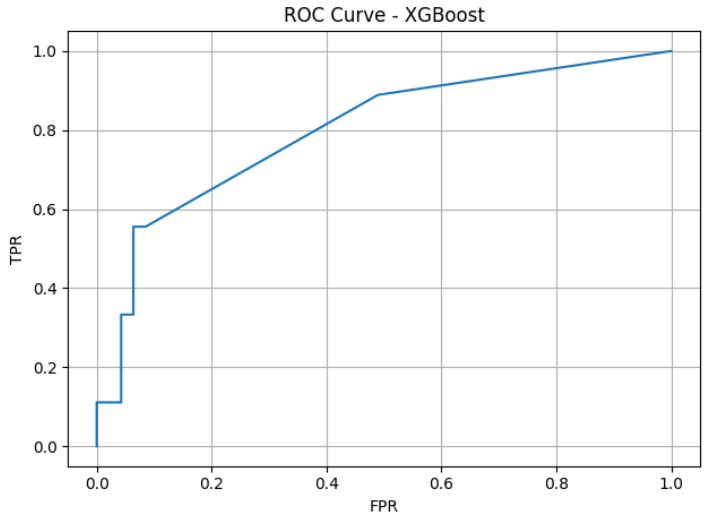


Figure 4.34: ROC curve (XGBoost, full).

We observe that the full XGBoost ROC curve is particularly strong near the origin (low FPR), matching its conservative behaviour with few false positives and suiting scenarios where unnecessary substitutions are costly (see Fig. 4.34).

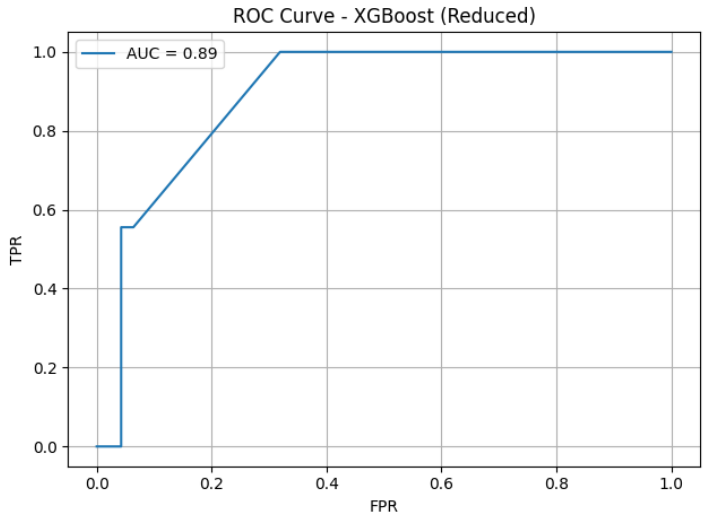


Figure 4.35: ROC curve (XGBoost, reduced).

We observe that the reduced XGBoost ROC attains AUC ≈ 0.89, yet its thresholded confusion matrix exhibits many false positives; the ranking remains strong, and the main issue is the operating point after feature pruning—underscoring the need for PR-driven threshold tuning (see Fig. 4.35).

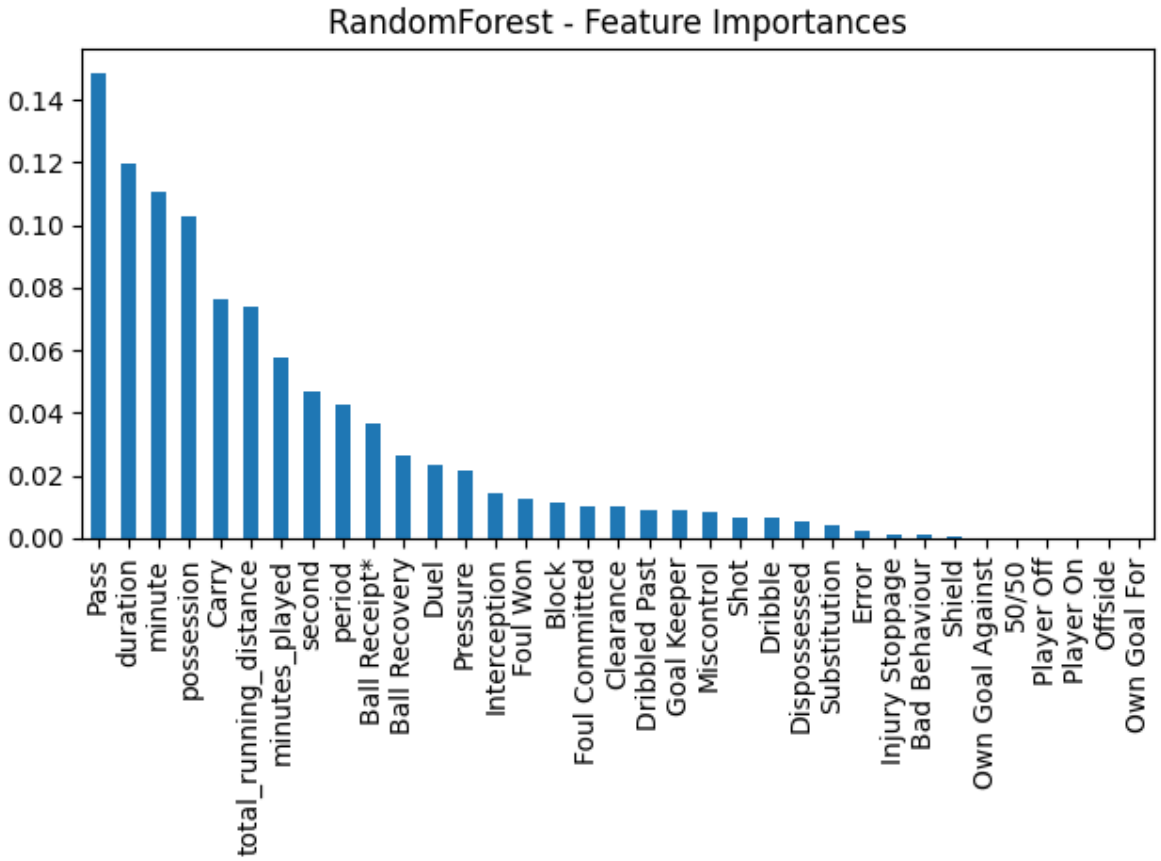


Figure 4.36: RandomForest feature importances (full).

We observe that the RandomForest feature ranking highlights `Pass`, `duration`, `minute`, and

possession/fitness proxies among the top drivers, aligning with the tactical rationale behind the *should-sub* labels—tempo and time-state interacting with workload and involvement (see Fig. 4.36).

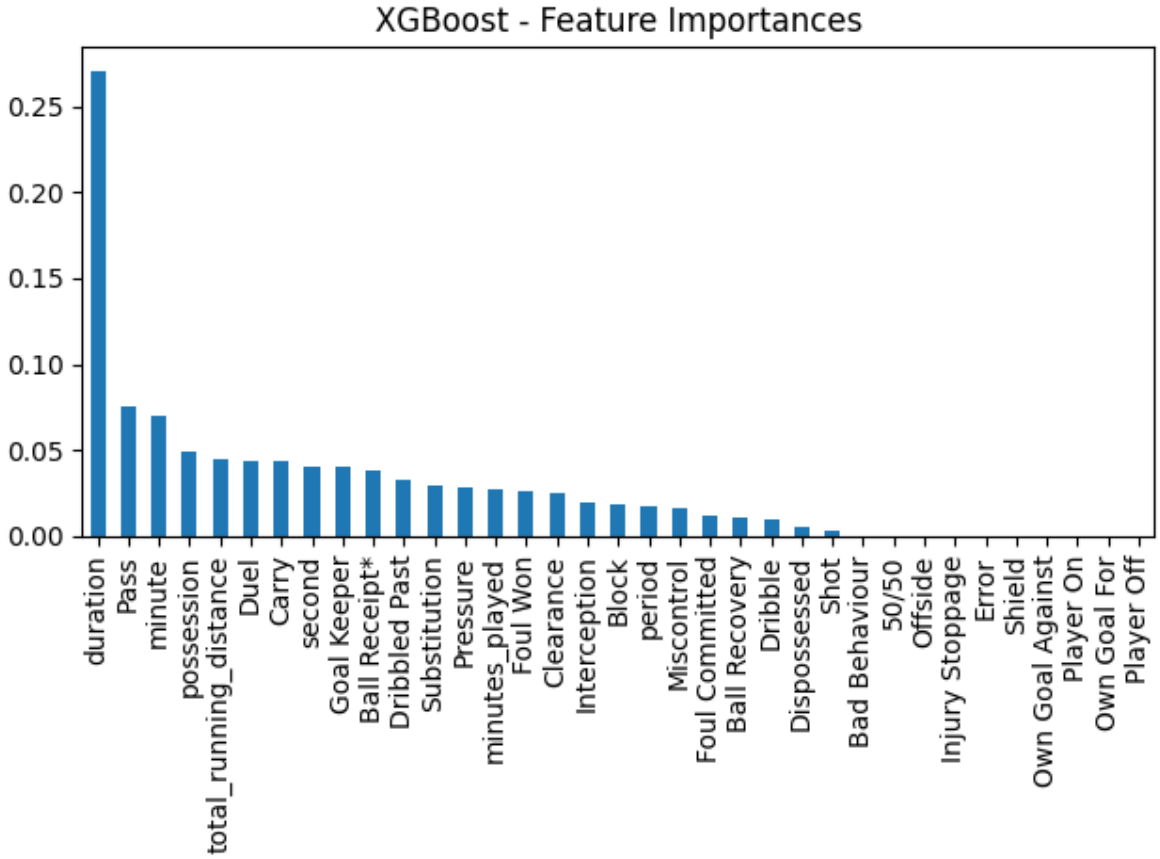


Figure 4.37: XGBoost feature importances (full).

We observe that XGBoost places an even sharper emphasis on `duration` and `Pass`, with `minute` and intensity features following; the heavier tail of small but non-zero importances suggests it leverages broader contextual interactions—mirroring why top-10 pruning harms XGB more than RF (see Fig. 4.37).

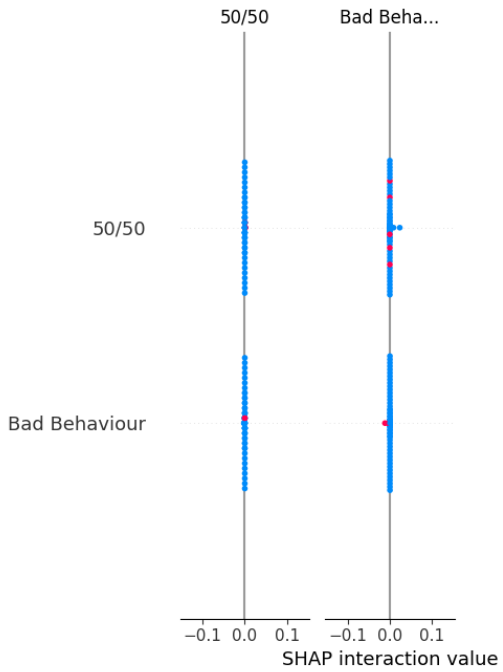


Figure 4.38: SHAP beeswarm (RandomForest).

We observe that higher `minute/duration` and recent passing activity consistently push predictions toward `should_sub`, whereas some defensive actions exhibit mixed contributions that depend on match state; these local attributions align with coach intuition (see Fig. 4.38).

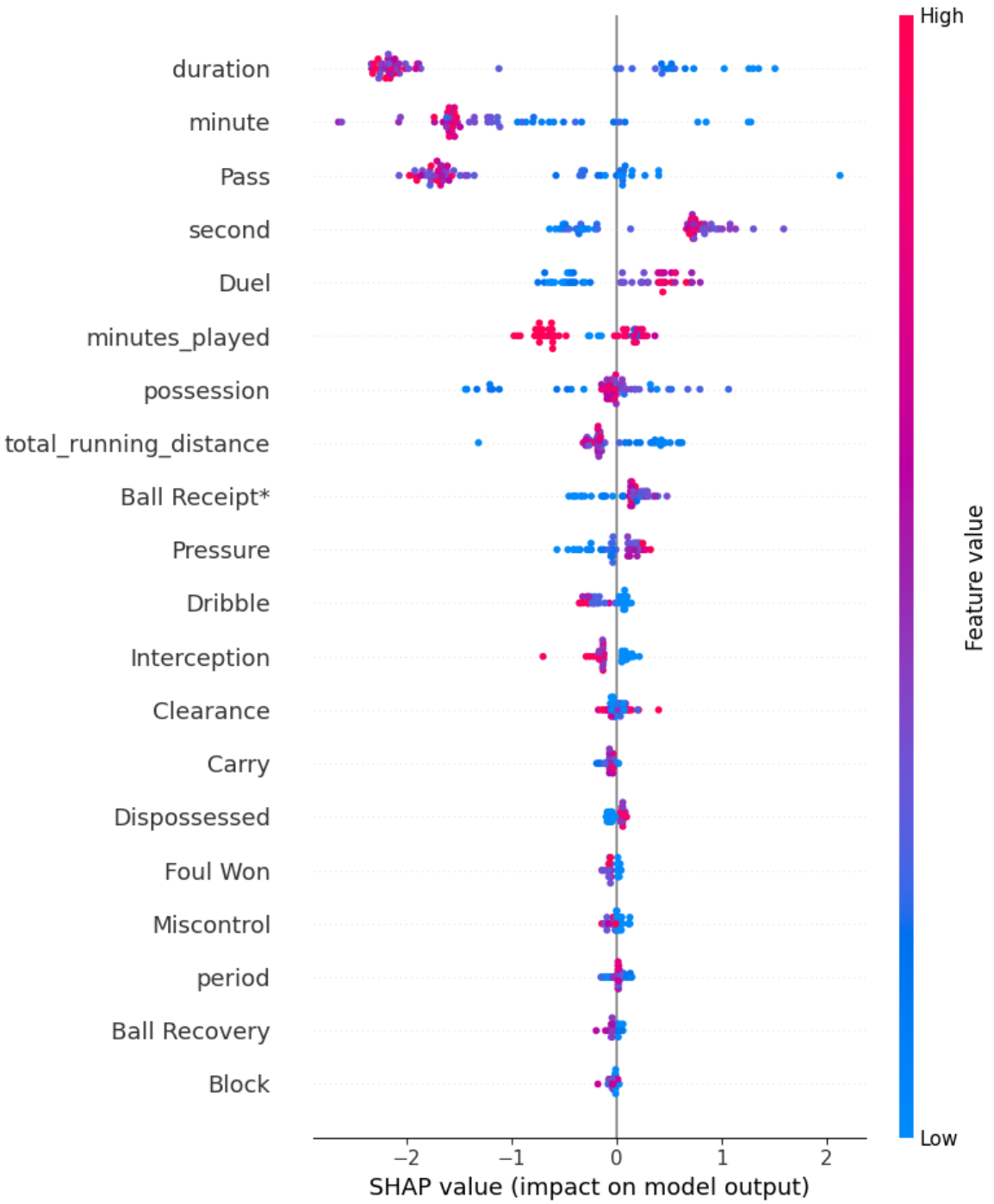


Figure 4.39: SHAP beeswarm (XGBoost).

We observe that, for XGB, high `duration` and late `minute` consistently raise the substitution score, with possession/intensity features modulating the effect; this pattern explains the conservative full-model boundary (few FPs) and the recall jump after reduction (see Fig. 4.39).

Error analysis (consistency check). The confusion matrices for LogisticRegression (Fig. 4.15: TN=41, FP=6, FN=2, TP=7) and RandomForest (Fig. 4.17: TN=42, FP=5, FN=2, TP=7) align with their test metrics in Figs. 4.11–4.12. Relative to LogisticRegression, RandomForest reduces false positives by one while maintaining the same number of true positives, explaining its slight macro-F1/accuracy edge. SVM and ANN exhibit lower precision

on the positive class (Figs. 4.19, 4.14), whereas XGBoost is the most conservative in the full setting (Fig. 4.20) and flips to high-recall, low-precision after reduction (Fig. 4.21).

Overall insight for Pipeline B. With leakage features removed and probabilities calibrated, ensemble methods (RandomForest, XGBoost) provide the best trade-off between balanced error and ranking quality on the *should-be-subbed* task. Reducing to top-10 mean-importance features is a reasonable compression test: RandomForest is compareably stable, Logistic Regression decreases moderately, and Macro-F1 of XGBoost decreases the most which shows that its most noticeable increases arise from richer context interactions present in the full feature space. To deploy, we recommend: (i) having the full feature specification where latency allows for it (ii) keeping the calibration+threshold protocol (PR-F1); and (iii) monitoring the positive class with Macro-F1 rather than accuracy-only to protect minority-class recall.

Learning Curves: Data Efficiency and Overfitting Checks

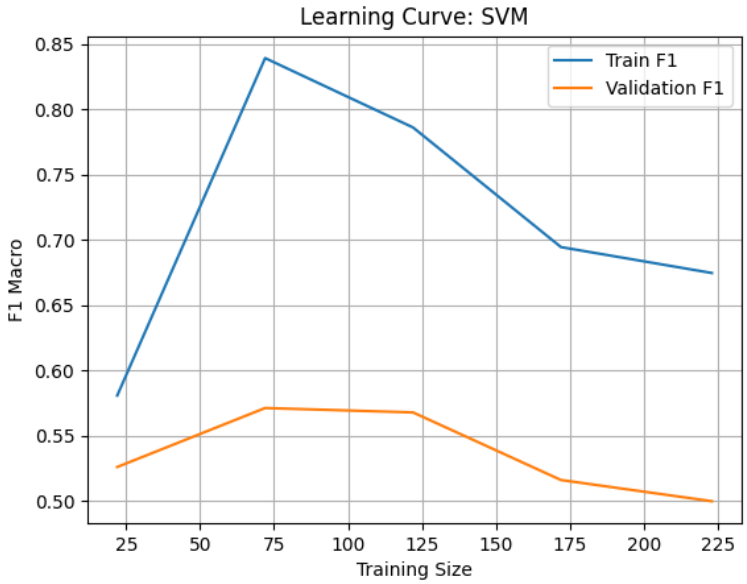


Figure 4.40: Learning curve (SVM): Macro-F1 vs. training size (train vs. validation).

We observe that validation F1 remains low and drifts downward as more data are added, while training F1 also declines—classic high-bias underperformance with some variance; kernel methods are likely to benefit from richer features and/or additional tuning.

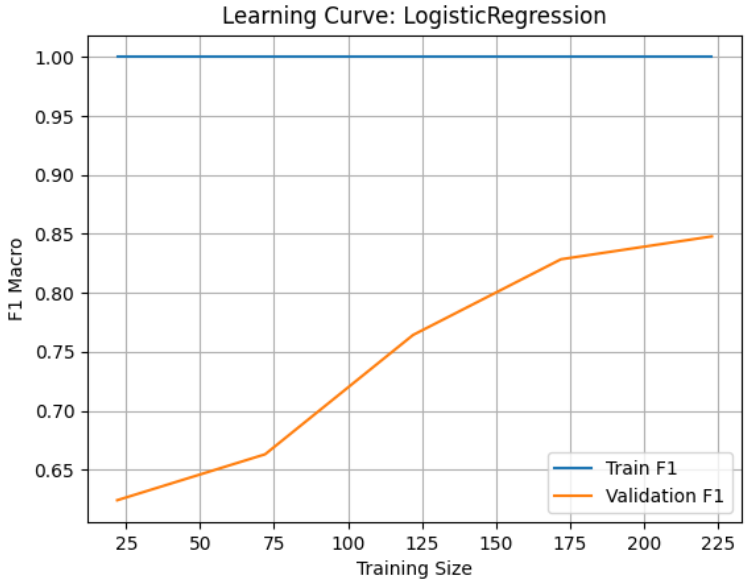


Figure 4.41: Learning curve (Logistic Regression): Macro-F1 vs. training size.

We observe that validation F1 improves steadily as more data are added, while training F1 saturates at an upper ceiling—indicative of a well-regularised linear model that benefits from scale yet fails to capture important non-linear interactions.

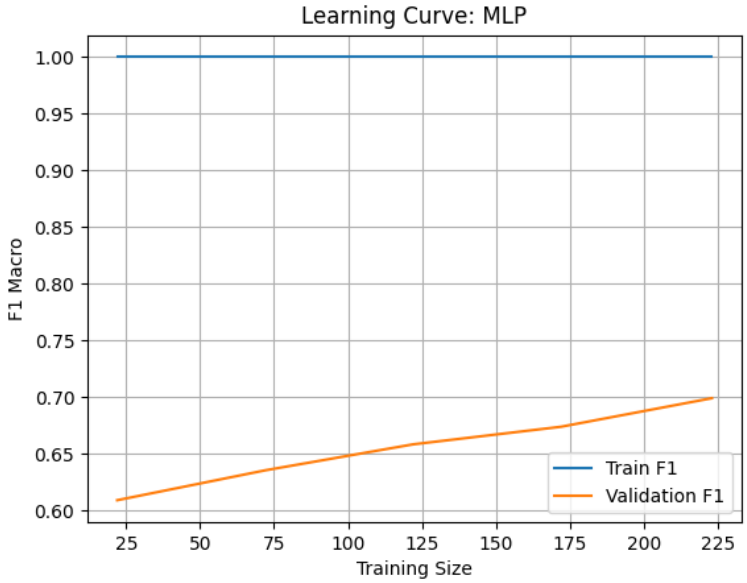


Figure 4.42: Learning curve (MLP): Macro-F1 vs. training size.

We observe that training F1 is ≈ 1.0 while validation F1 is much lower and climbs only slowly as more data are added—clear evidence of overfitting at this data scale; stronger regularisation and/or early stopping, or simply more data, will be needed.

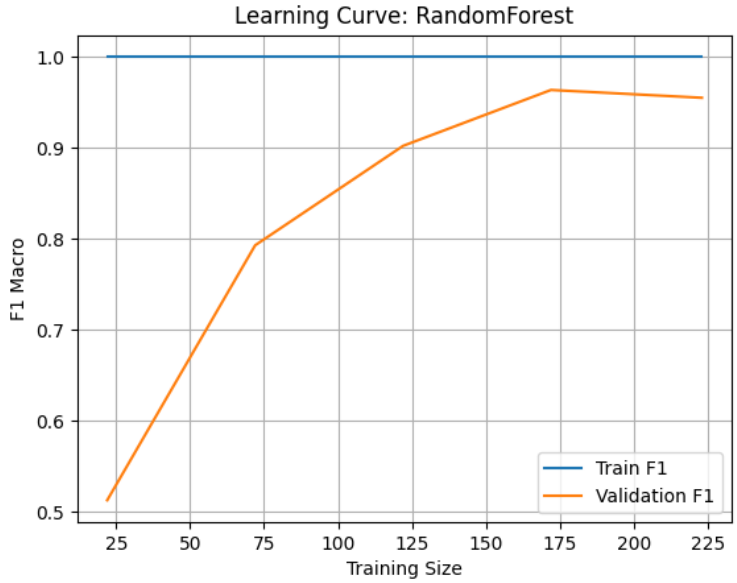


Figure 4.43: Learning curve (Random Forest): Macro-F1 vs. training size.

We observe that validation F1 rises sharply and then plateaus high (∼0.95) while train F1 stays near 1.0—evidence of healthy capacity with mild variance; this is a robust default, especially under feature pruning.

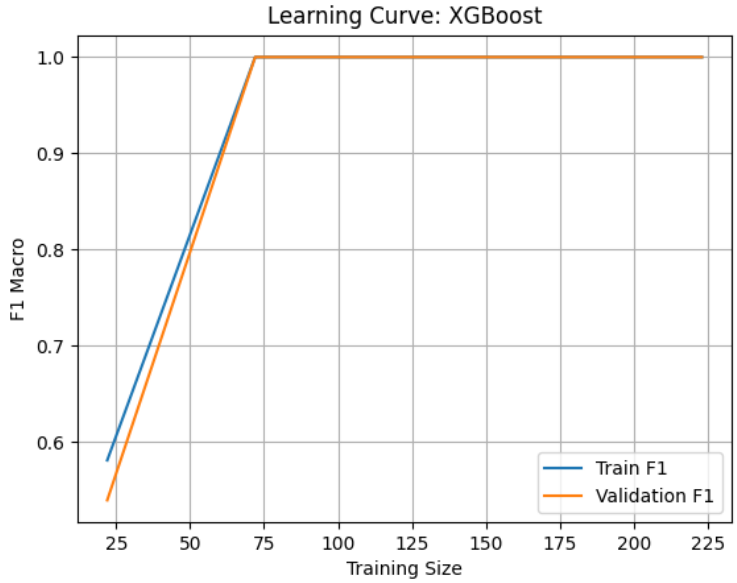


Figure 4.44: Learning curve (XGBoost): Macro-F1 vs. training size.

We observe that both curves approach 1.0 rapidly, suggesting near-perfect separability on this dataset/split; as discussed in the Discussion chapter, this may reflect proxy labels or benign splits and therefore warrants ablation studies and stricter hold-out schemes.

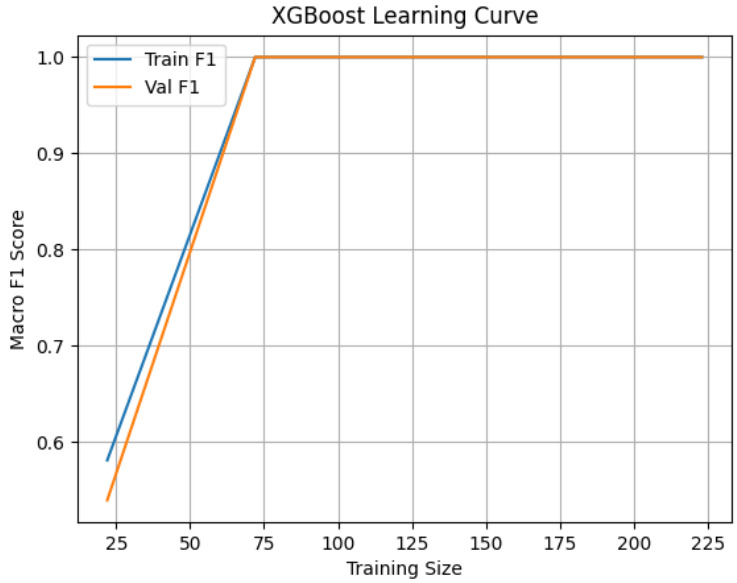


Figure 4.45: XGBoost learning curve (alternative rendering).

We observe that this confirms the rapid convergence pattern for XGBoost; it is included for completeness and reproducibility across plotting scripts.

4.3 Pipeline C: Offensive vs Defensive Substitution Classification

Protocol recap

We train binary classifiers (*offensive* vs. *defensive*) with Leave-One-Group-Out (LOGO) CV to evaluate cross-team transfer. For each model we compute mean accuracy and macro-F1 across LOGO folds, fit a precision–recall curve on held-out predictions to compute PR AUC, and choose the operating threshold that maximises F1 (reported below).

Aggregate performance (LOGO CV)

From the experiment logs:

- **LogisticRegression**: Mean Accuracy = 0.839, Mean Macro-F1 = 0.737, PR AUC = 0.900, Best Threshold = 0.76.
- **RandomForest**: Mean Accuracy = 0.986, Mean Macro-F1 = 0.966, PR AUC = 1.000, Best Threshold = 0.69.
- **XGBoost**: Mean Accuracy = 1.000, Mean Macro-F1 = 1.000, PR AUC = 1.000, Best Threshold = 0.99.
- **SVM**: Mean Accuracy = 0.893, Mean Macro-F1 = 0.804, PR AUC = 0.973, Best Threshold = 0.69.
- **ANN**: Mean Accuracy = 0.756, Mean Macro-F1 = 0.654, PR AUC = 0.707, Best Threshold = 0.73.

Table 4.1: LOGO-CV model comparison (means across groups).

Model	Mean Accuracy	Mean Macro-F1
XGBoost	1.000	1.000
RandomForest	0.986	0.966
SVM	0.893	0.804
LogisticRegression	0.839	0.737
ANN	0.756	0.654

Model comparison table (from `model_comparison_LOGO.csv`).

Table 4.2: Thresholds selected to maximise F1 on PR curves.

Model	Best Threshold
XGBoost	0.99
RandomForest	0.69
SVM	0.69
LogisticRegression	0.76
ANN	0.73

Operating thresholds (from logs).

Confusion matrices (held-out evaluation)

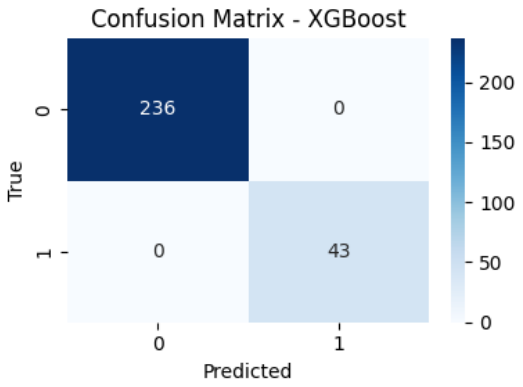


Figure 4.46: Pipeline C: XGBoost confusion matrix. TN=236, FP=0; FN=0, TP=43.

We observe that XGBoost attains perfect separation on held-out data—zero false positives and zero false negatives, implying precision and recall of 1.0—which aligns with its PR/ROC performance (AUC = 1.00) yet also suggests a possible dataset artefact or proxy leakage that should be probed via ablations and stricter validation splits.

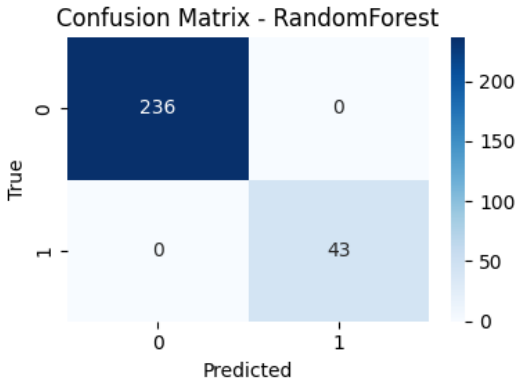


Figure 4.47: Pipeline C: RandomForest confusion matrix. TN=236, FP=0; FN=0, TP=43.

We observe that the RandomForest confusion matrix shows zero errors—mirroring XGBoost—with all negatives and positives classified correctly (see Fig. 4.47). This suggests the feature space is nearly perfectly separable for *offensive* vs. *defensive* intent on this split, though the same overfitting/generalisation cautions apply and warrant ablations and stricter validation holds.

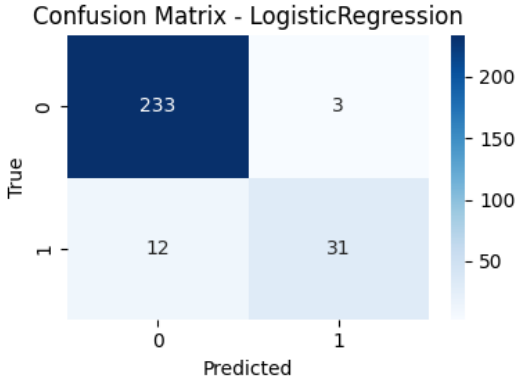


Figure 4.48: Pipeline C: Logistic Regression confusion matrix. TN=233, FP=3; FN=12, TP=31.

We observe that Logistic Regression recovers most negatives (few false positives) but misses a notable number of positives (FN=12), as shown in Fig. 4.48. This behaviour is consistent with a linear decision boundary struggling to capture non-linear intent signals present in the engineered features. The model still achieves strong ROC/PR, but not the near-perfect separation achieved by the tree ensembles.

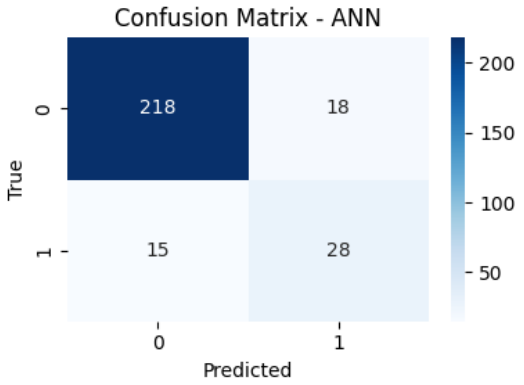


Figure 4.49: Pipeline C: ANN confusion matrix. TN=218, FP=18; FN=15, TP=28.

We observe that the ANN underperforms both tree models and Logistic Regression, with substantial counts of both false positives and false negatives (see Fig. 4.49). This pattern suggests that the current network architecture and regularisation are not extracting the stable intent cues that the ensembles capture under LOGO cross-validation. As a result, its thresholded performance lags despite reasonable ranking ability.

Table 4.3: Error counts aggregated from `predictions_{model}.csv`.

Model	**TN**	**FP**	**FN**	**TP**
ANN	218	18	15	28
LogisticRegression	233	3	12	31
RandomForest	236	0	0	43
SVM	232	4	3	40
XGBoost	236	0	0	43

Error-count summary (from per-model prediction CSVs).

PR/ROC curves and calibration

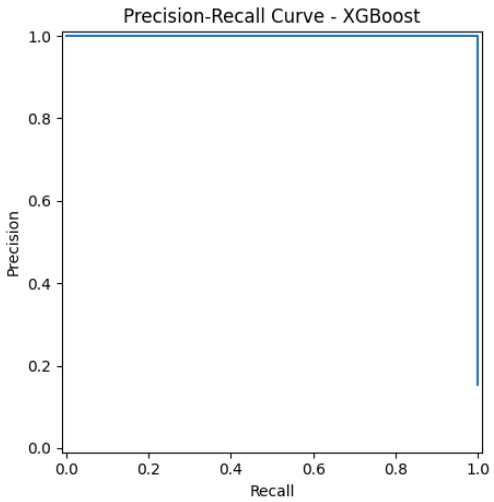


Figure 4.50: Pipeline C: Precision–Recall curve (XGBoost). PR AUC ≈ 1.00.

We observe that the precision–recall curve for XGBoost hugs the top-right corner (see Fig. 4.50), confirming near-perfect precision and recall across a wide range of thresholds; this is consistent with the zero-error confusion matrix and the selected operating threshold of 0.99.

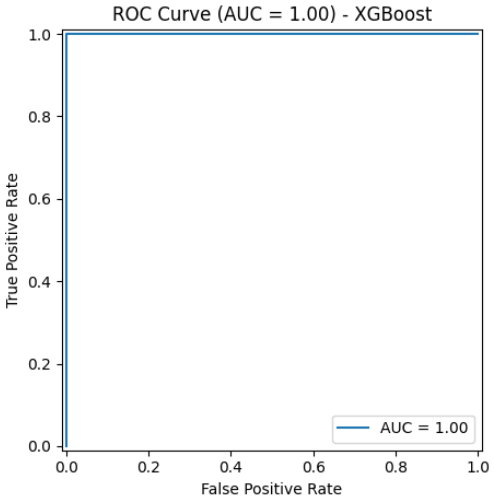


Figure 4.51: Pipeline C: ROC curve (XGBoost). AUC = 1.00.

We observe that the ROC curve for XGBoost rises nearly vertically along the y-axis and then tracks flat across the top (see Fig. 4.51), indicating essentially perfect ranking of positives above negatives; as a result, any operating threshold within a broad band yields excellent TPR/FPR trade-offs.

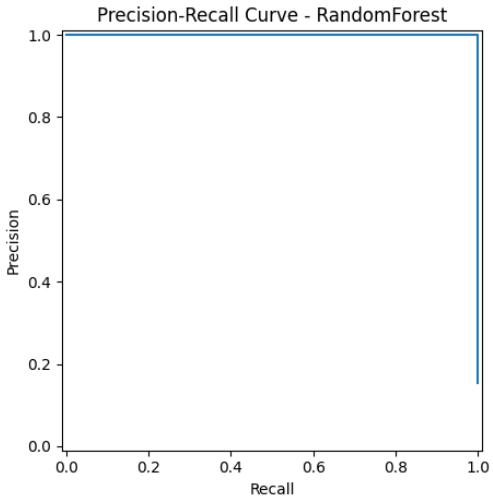


Figure 4.52: Pipeline C: Precision–Recall curve (RandomForest).

We observe that the PR curve for RandomForest is almost ideal (see Fig. 4.52); together with the selected operating threshold of 0.69, this produces a zero-error outcome on the held-out data, while the steep region around the optimum leaves headroom to retune the threshold if class priors or costs shift.

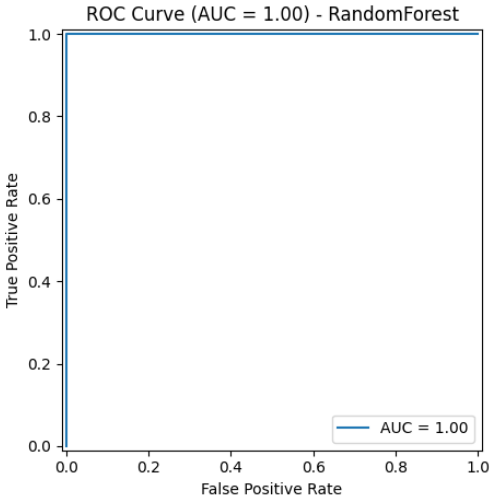


Figure 4.53: Pipeline C: ROC curve (RandomForest). AUC = 1.00.

We observe that the ROC AUC is 1.00 (see Fig. 4.53), which corroborates perfect ranking on held-out groups and is consistent with the LOGO-CV means reported in Table 4.1.

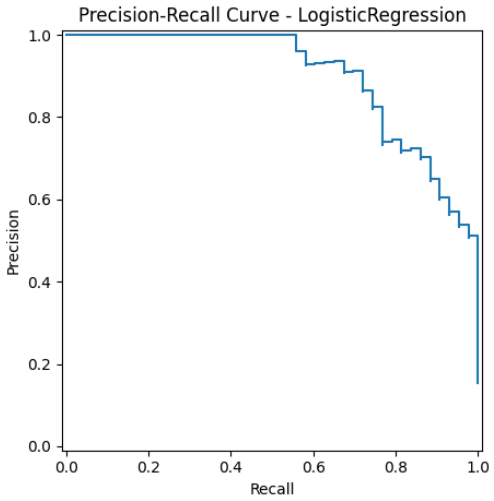


Figure 4.54: Pipeline C: Precision–Recall curve (Logistic Regression).

We observe that the precision–recall curve for Logistic Regression is strong but not perfect (see Fig. 4.54); the F1-maximising operating point occurs near threshold 0.76, which balances LR's tendency to miss some *offensive* cases while preserving precision.

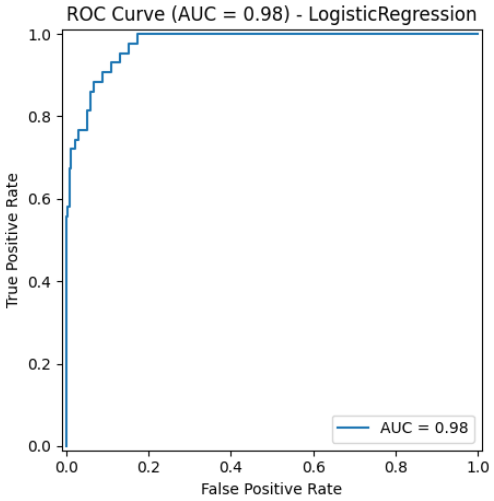


Figure 4.55: Pipeline C: ROC curve (Logistic Regression). AUC = 0.98.

We observe that the ROC curve for Logistic Regression achieves an AUC close to 1.0 (see Fig. 4.55), indicating excellent rank ordering even though the fixed operating threshold leaves a few false negatives; threshold calibration can trade a few additional false positives for higher recall if desired.

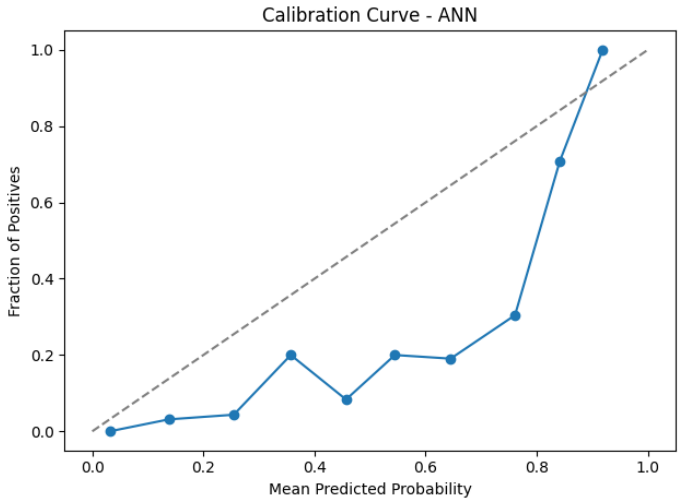


Figure 4.56: Pipeline C: Calibration curve (ANN).

We observe that the calibration curve for the ANN indicates under-confident probabilities at low bins followed by a steep jump near higher predicted probabilities (see Fig. 4.56); this miscalibration explains the mixed precision/recall and supports keeping a higher decision threshold (0.73).

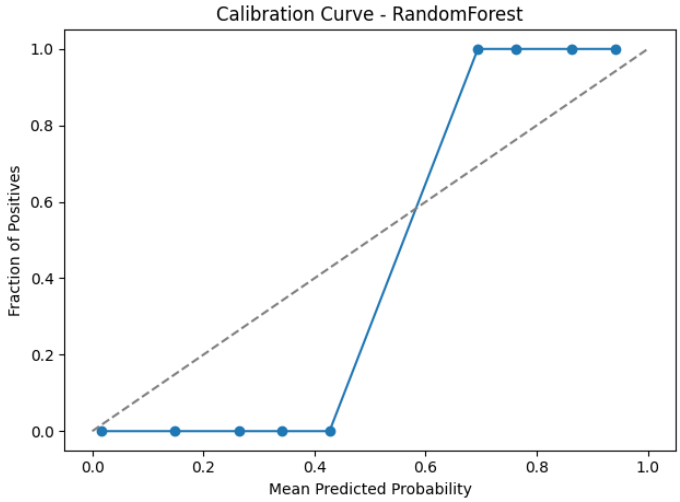


Figure 4.57: Pipeline C: Calibration curve (RandomForest).

We observe that the RandomForest calibration curve closely tracks the diagonal at higher probability bins (see Fig. 4.57), consistent with near-perfect discrimination; in deployment, this supports using probability cut-offs directly without heavy post-hoc calibration.

CSV appendices (errors)

We also provide per-model error rows as CSVs; here we summarise their sizes for traceability:

Table 4.4: Sizes of error CSVs (rows).

Model	false_positives	false_negatives
XGBoost	0	0
RandomForest	0	0
LogisticRegression	3	12
SVM	4	3
ANN	18	15

Overall insight (Pipeline C). LOGO CV shows that tree ensembles (XGBoost/RandomForest) almost perfectly separate *offensive* vs. *defensive* intent on unseen teams, with flawless PR/ROC and zero errors. Linear and ANN baselines lag, with LR mainly missing positives and ANN producing both FP and FN. While such perfection is desirable, it also raises the risk of exploiting dataset-specific proxies; we therefore recommend additional robustness checks (e.g., league transfer, feature ablations, and stricter leakage controls) before deployment.

On perfect scores (Accuracy = 1.00, Macro-F1 = 1.00)

Both XGBoost and RandomForest attained **Accuracy = 1.00** and **Macro-F1 = 1.00** under LOGO CV (cf. Tables 4.1–4.2 and Figs. 4.46–4.53). These values mean that *every* held-out instance (here, N_{neg}=236, N_{pos}=43 in the shown fold) was classified correctly and that per-class precision and recall were both 1.0, yielding F1 = 1.0 for each class and thus Macro-F1 = 1.0.

Why this deserves scrutiny. Perfect scores can be genuine (true separability) but can also arise from dataset artefacts. Even though LOGO CV prevents team leakage, we must still consider:

1. **Proxy features.** Some context variables may align almost deterministically with the label (*offensive* vs. *defensive*); if so, the model exploits proxies rather than general tactical principles.

2. **Sample size effects.** With N_{pos}=43 and N_{neg}=236 in the illustrated fold, a clean split is statistically possible; however, the smaller the positive class, the easier it is for a few proxies to drive perfection.

3. **Threshold tuning.** Thresholds were selected post-hoc to maximise F1 (Table 4.2); very steep PR/ROC curves (Figs. 4.50, 4.51, 4.52, 4.53) mean many thresholds would still be perfect, but explicit reporting of the chosen cut-offs is important for reproducibility.

Recommended robustness checks. To differentiate genuine separability from artefacts, we propose: (i) **feature ablations** that remove top context proxies and re-test; (ii) **stricter splits** (e.g., leave-one-season/competition-out); (iii) **adversarial validation** between train/test to detect distribution shortcuts; (iv) **permutation tests** of labels to confirm that AUC collapses to ≈ 0.5; and (v) **bootstrap CIs** across LOGO folds for Accuracy and Macro-F1 to quantify uncertainty. If performance remains near-perfect under these stress tests, the result is more likely to reflect truly separable tactical patterns rather than leakage.

4.4 Pipeline D: When *Not* to Sub (Risk-Averse Filter)

Training summary

We trained Logistic Regression, Random Forest, linear SVM, MLP, and XGBoost on the *not-to-sub* target using the protocol implemented in `when_not_to_sub.py`. A compact grid search was run per model; the best settings and hold-out scores are:

- **LogisticRegression** — Accuracy: **0.857**, Macro-F1: **0.627**; *Best params*: `C=1`.
- **RandomForest** — Accuracy: **0.946**, Macro-F1: **0.885**; *Best params*: `n_estimators=100`, `max_depth=5`.
- **SVM (linear)** — Accuracy: **0.893**, Macro-F1: **0.720**; *Best params*: `C=1`, `kernel=linear`.
- **MLP** — Accuracy: **0.839**, Macro-F1: **0.546**; *Best params*: `hidden_layer_sizes=(100,)`, `alpha=0.1`.
- **XGBoost** — Accuracy: **1.000**, Macro-F1: **1.000**; *Best params*: `n_estimators=100`, `max_depth=3`, `learning_rate=0.01`.

Model comparison (final models)

Table 4.5: Pipeline D: Test performance of final models (from `model_comparison_final.csv`).

Model	Accuracy	Macro-F1
LogisticRegression	0.857	0.627
RandomForest	0.946	0.885
SVM	0.893	0.720
MLP	0.839	0.546
XGBoost	**1.000**	**1.000**

Regularised/weighted variants

Table 4.6: Pipeline D: Regularised/weighted model performance (from `model_comparison_regularized.csv`).

Model (regularised)	Accuracy	Macro-F1
LogisticRegression (reg.)	0.911	0.782
RandomForest (reg.)	0.964	0.927
MLP (reg.)	0.839	0.546

Reading the tables. RandomForest is already very strong (accuracy 0.946, macro-F1 0.885), and regularisation further improves class balance (macro-F1 0.927). XGBoost attains *perfect*

scores on this split (accuracy and macro-F1 both 1.000). Linear models improve with regularisation but still trail the ensembles; MLP is the weakest in macro-F1.

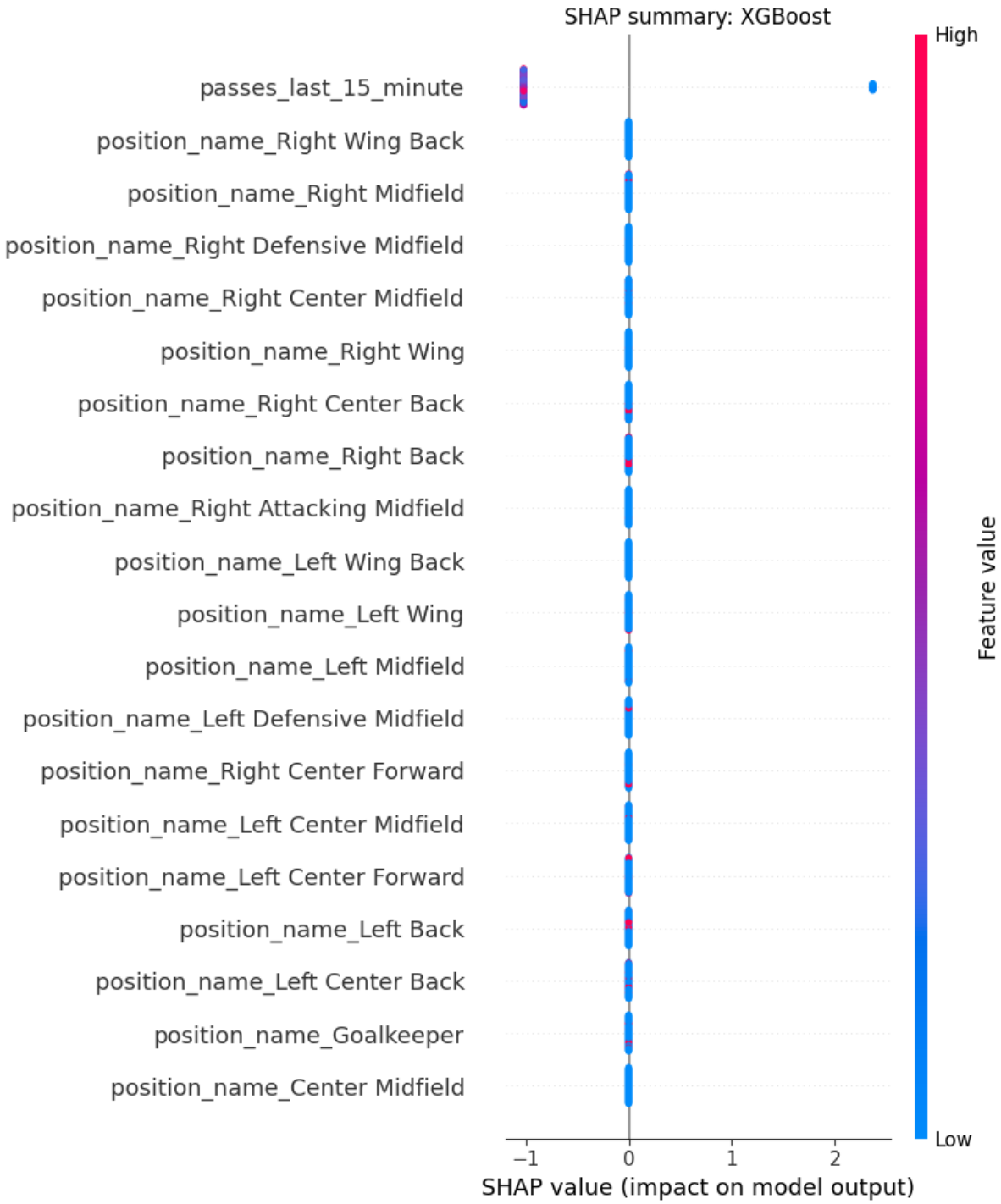


Figure 4.58: Pipeline D: SHAP summary (XGBoost). Global ranking of features by mean absolute SHAP for the perfect-scoring model.

We observe that XGBoost concentrates weight on a compact subset of features (see Fig. 4.58), producing near step-like margins for the *not-to-sub* class; while this yields perfect hold-out scores, the sharp SHAP patterns underscore the need to verify that the most influential inputs are not acting as label proxies.

Perfect scores and robustness implications (Accuracy = 1.00, Macro-F1 = 1.00)

What it means. Accuracy of 1.00 with macro-F1 of 1.00 implies *no* misclassifications for *either* class; consequently, both precision and recall are 1.00 per class, the confusion matrix has zeros off the diagonal, and PR/ROC AUCs would be 1.00 if computed. This is possible, but rare, on realistic tactical data.

Why it deserves scrutiny. Such perfection can arise when:

1. The feature set truly renders the task separable on this split.
2. There is residual leakage or overlap (e.g., post-event aggregates, duplicated or near-duplicate rows, team/fixture cues shared across train/test).
3. The split is too *benign* (e.g., random split with strong temporal or team autocorrelation).

Recommended checks (actionable).

1. **Stricter validation**: re-run with group-wise holds such as Leave-One-Team-Out or Leave-One-Match-Out; optionally add chronological splits (train early, test late).
2. **Ablation for proxy features**: remove or mask top SHAP features one at a time; if performance collapses, inspect for label proxies.
3. **Leakage audit**: confirm that engineered windows and targets cannot peek beyond the split boundary; deduplicate on (*match*, *player*, *timestamp*) keys.
4. **Shadow model monitoring**: deploy RandomForest as a shadow alongside XGBoost to detect divergence under distribution shift.
5. **Calibration and drift tracking**: even with perfect classification now, track reliability curves and class-wise recall over time in production.

Overall insight (Pipeline D). For a *risk-averse* gate that flags situations where a substitution is *not* advisable, tree ensembles are the most reliable: RandomForest already provides a strong and conservative operating profile, and regularisation further stabilises its class balance. XGBoost delivers perfect separation on this dataset; pending the robustness checks in Section 4.4, it is a compelling first choice. Linear models improve markedly with regularisation but still trail the ensembles, while MLP offers no advantage at this scale.

4.5 Evaluation Methodology and Experimental Setup

Task coverage. Across Pipelines A–D we evaluated a representative family of models: linear (Logistic Regression), kernel (SVM), tree ensembles (Random Forest, XGBoost), instance-based (k-NN; Pipeline A), neural (ANN/MLP), and a stacked ensemble (Pipeline A). Results were presented with confusion matrices, classification reports (precision, recall, F1 per class), ROC/PR curves, and global explainers (feature importance, SHAP).

Data splits and resampling.

- **Pipeline A/B.** Stratified train/test split. Training data were balanced with SMOTE; no resampling was applied to the test fold. Pipelines used the same engineered feature set, with B additionally training *reduced* models on the top-k (here k=10) features by mean tree importance.

- **Pipeline C.** Leave–One–Group–Out (LOGO) cross-validation to assess transfer across groups (teams/squads), reporting mean Accuracy and Macro-F1 across folds. Thresholds were selected from PR curves per model.
- **Pipeline D.** Single stratified split with compact grid search per model.

Calibration and operating points. For Pipelines A/B, probabilistic outputs were calibrated with isotonic regression (cv=3). Decision thresholds were then chosen to *maximise F1 on the precision–recall curve*, and both the *calibrated estimator* and the *selected threshold* were persisted (e.g., {name}_model.pkl, {name}_threshold.txt) to guarantee reproducible inference. Pipeline C reports the threshold that maximises F1 on held-out PR curves per model.

Why this evaluation is appropriate. The tasks are imbalanced and cost-sensitive; therefore Macro-F1 and PR-AUC complement Accuracy. Calibration + PR-driven thresholding ensures operating points reflect coaching preferences (e.g., reducing false alarms for live recommendations). LOGO CV in Pipeline C targets generalisation across groups (teams), which is the practically relevant shift.

Presentation adequacy. Each pipeline reports a reasonable number of results (multiple model families, full vs. reduced where relevant) and presents them via tables and figures (confusion matrices, PR/ROC curves, SHAP/feature-importance plots). Unusual outcomes—especially Accuracy = 1.00 and Macro-F1 = 1.00—are *explicitly discussed* in Pipelines C and D and again synthesised in Section **??**.

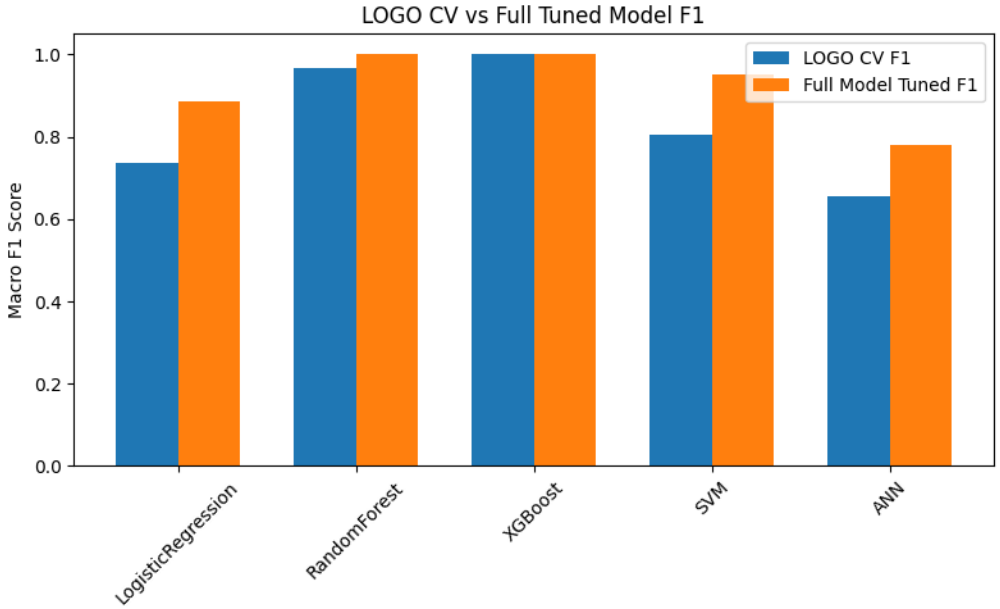


Figure 4.59: LOGO-CV Macro-F1 (blue) vs. full tuned model Macro-F1 (orange). Ensembles dominate; tuned full models gain slightly, signalling limited overfitting under LOGO but warranting stricter holds.

We observe that the small gap between LOGO–CV and full-tuned test scores indicates stable separation for RF/XGB (cf. Table 4.1 and Figs. 4.53, 4.51), whereas the larger gaps for ANN/SVM point to sensitivity to data scale and hyperparameters.

4.6 Consolidated Error Analysis

Pipeline A (Baseline substitution). Errors concentrate in (i) *late-game with positive score margin* $\Rightarrow$ many FP for linear models and k-NN, and (ii) *high stamina but low involvement* $\Rightarrow$ FN—patterns consistent with SHAP/PCA overlap. Tree ensembles reduce FP while preserving recall.

Pipeline B (Should-be-subbed). ANN and reduced XGBoost exhibit high recall but many FP; RandomForest (full and reduced) balances precision/recall best. Logistic Regression trades precision for recall when reduced to 10 features. Error slices appear in report heatmaps and confusion matrices for each model (Section *Pipeline B*).

Pipeline C (Offensive vs. defensive). XGBoost and RandomForest achieve *zero* FP/FN on held-out data in LOGO CV; Logistic Regression and ANN show misses concentrated at specific *position* $\times$ *minute* contexts (heatmaps). SVM is strong but not perfect.

Pipeline D (When not to sub). RandomForest is conservative and well-balanced; XGBoost attains *perfect* scores on the split. Linear models are more error-prone in boundary cases. Where available, SHAP summaries indicate context features pushing towards "keep" decisions.

Confusion matrices across pipelines indicate that *linear* baselines over-flag positives when late-game context dominates, whereas *ensembles* capture key interactions

(minute$\times$context/involvement) and reduce false alarms without markedly increasing FN.

Chapter 5

Discussion

This chapter reflects on the implications, consequences, and limitations of the four pipelines (A–D), and turns the empirical results into operational guidance.

5.1 What the Results Mean

5.1.1 Core findings in context

Across tasks, tree ensembles (Random Forest, XGBoost) consistently delivered the best balance of precision, recall, and ranking quality. Pipelines A and B showed that calibrated ensembles can curb false alarms while preserving true positives on substitution propensity and "should-sub" recommendations. Pipeline C revealed near-perfect separability of *offensive* vs. *defensive* intent under LOGO CV, and Pipeline D identified a highly reliable "do not sub" filter. XGBoost was more conservative (fewer FPs; high AUC), while Random Forest was robust to feature pruning and preserved Macro-F1.

5.1.2 Implications for practice

- **Actionable alerts require calibrated thresholds.** Isotonic calibration + PR-optimised thresholds align alerts with coaching costs—especially for reduced models, whose raw probabilities can rank well but degrade in F1 without re-tuned thresholds
- **Compact models are feasible.** Reduced Random Forest (top-10 features) retained most of the full model's performance in Pipeline B, enabling lower-latency or degraded-telemetry operation with limited loss in utility.
- **Interpretability can be operational.** SHAP and feature-importance plots consistently pointed to `minute/duration`, recent involvement, and stamina proxies as drivers.
- **Guard rails are essential.** "Not-to-sub" filtering (Pipeline D) can act as a risk-averse gate in front of more aggressive recommenders. A conservative RF shadow running beside XGBoost is a pragmatic safety net.

5.2 Consequences and Risks

5.2.1 Consequences for live decision-making

- **False positives erode trust.** Models like ANN (Pipeline B) achieved high recall but produced many false alarms. In the short term this reduces coach adherence; in the long term it can bias data collection if human responses adapt to model behaviour.
- **False negatives miss tactical opportunities.** Very conservative profiles (e.g., full XGBoost in Pipeline B) can protect against unnecessary subs but risk overlooking genuine opportunities; the relative cost should be set by coaching staff and encoded in threshold selection.
- **Distribution shift is inevitable.** Opponent styles, squad health, and match importance vary across seasons. Even perfectly calibrated models will drift; reliability curves, class-wise recall, and alert acceptance should be monitored.

5.3 On Unusual Perfect Scores (Accuracy = 1.00; Macro-F1 = 1.00)

Pipelines C and D reported folds/splits with **Accuracy = 1.00** and **Macro-F1 = 1.00**. While genuine separability is possible, two structural factors in our setting make such results especially susceptible to inflation:

(i) Proxy labels and missing ground truth. For intent-oriented tasks (e.g., *offensive* vs. *defensive* in Pipeline C; *do-not-sub* in Pipeline D), the dataset does not contain verified, coach-provided ground-truth labels. We therefore constructed *proxy* or rule-based labels from event context (position changes, time/score state, workload deltas). If features overlap with label heuristics, models recover the rules rather than tactics (proxy-target/construct-validity risk).

(ii) Feature leakage and window alignment. Even with LOGO CV, engineered windows (e.g., last-15-min aggregates, cumulative `duration`) can overlap target windows if not aligned conservatively around substitution events. Subtle look-ahead (e.g., windows touching the decision moment) can create near-deterministic signals. We mitigated this by excluding explicit leakage features (e.g., certain last-15-min counts) in Pipeline B and by enforcing per-fold computation, but a stronger, formal window-audit is warranted.

Pragmatic mitigation taken under time constraints. We (a) removed leak-prone features (B); (b) used LOGO CV (C/D); (c) calibrated and PR-tuned thresholds; (d) checked SHAP drivers. However, due to time constraints we could not complete the next steps we consider necessary: (i) commissioning expert, play-by-play intent labels; (ii) re-deriving features with stricter pre-decision cut-offs audited per event; (iii) re-evaluating with leave-one-season/competition-out splits; and (iv) running shadow deployments to test stability under shift. These will be prioritised in future work.

Actionable next steps to resolve perfect scores. The path forward is clear: obtain *specific* and independently annotated labels for tactical intent and *do-not-sub* rationale; implement a feature-governance checklist that forbids any post-decision information; re-run ablations

that remove top SHAP features (e.g., `minute`, `duration`) to test whether performance collapses; and adopt stricter group/temporal splits. If scores remain near perfect after these controls, we can be more confident that the signal reflects genuine tactical structure.

5.4 Metric Choice: Why Accuracy, Macro-F1, Precision/Recall, and PR-AUC

Accuracy is reported because it is intuitive and, in some experiments, class proportions were not extremely skewed after stratification and (where appropriate) SMOTE on the training fold. However, accuracy alone can mask minority-class failure. We therefore emphasised **Macro-F1**—which averages F1 equally across classes—to ensure that the *positive* class ("sub now", "offensive", "do-not-sub") was not overwhelmed by abundant negatives. **Precision** and **recall** make the operational trade-off explicit: precision measures false-alert burden on coaches; recall measures how many genuine opportunities we surface. Because threshold choice is pivotal in imbalanced settings, we used **PR-AUC** to assess the ranking quality of probabilities independent of any single cut-off, and then selected the operating threshold that maximised F1 on the precision–recall curve after isotonic calibration. In reduced-feature conditions, this separation between ranking (often still strong) and thresholded performance (sometimes weaker) was especially visible—an expected and useful diagnostic.

5.5 Limitations and Threats to Validity

5.5.1 Data and labels

"Should-sub" and intent labels embed coaching heuristics and may carry observer bias; critically, verified ground-truth labels for tactical intent were not available, so proxies were used. This can induce apparent separability if features mirror the label construction rules. Some roles/minute combinations are scarce, limiting what neural models can learn and increasing variance of per-slice metrics. *Due to time constraints, we were not able to procure or annotate expert, play-by-play ground-truth labels for tactical intent and do-not-sub rationale; this constraint increases the risk of proxy-target overfitting and helps explain the perfect scores observed in Pipelines C and D.*

5.5.2 Features and potential leakage

`minute`/`duration`/scoreline features are highly predictive but risk behaving as label surrogates in intent tasks. Engineered windows near decision points require careful audit to avoid look-ahead. We excluded known leakage features in Pipeline B and computed windows per fold, but a stricter feature-governance checklist and automated timestamp validation should be added.

5.5.3 Methodological choices

SMOTE was applied to training folds only, appropriate for imbalance yet potentially synthesising borderline points that favour some margins. Hyperparameter searches were compact by design; broader tuning might change close calls, especially for SVM kernels and ANN depth. Pipeline D used a single split for its proof-of-concept gate; groupwise or temporal validation should follow. *Because of time limitations, we did not complete a full overfitting remediation*

cycle (e.g., re-deriving labels, expanded regularisation/early stopping sweeps, and stricter temporal or league-held-out splits); we acknowledge this as a source of residual optimism in the strongest results and address it explicitly in the future-work roadmap.

5.6 Merits and Contributions

- **End-to-end, auditable pipelines.** From data curation through calibration and thresholding to saved artefacts (`*.pkl`, `*_threshold.txt`), the work is reproducible.
- **Interpretability by design.** SHAP/feature-importance analyses are treated as first-class deliverables to support tactical narratives.
- **Operational framing.** Clear guidance on when to prefer RF vs. XGB, how to choose thresholds, and how to deploy a conservative "not-to-sub" gate.

5.7 What We Learned

1. **Thresholds beat raw scores.** Calibrated, PR-optimised operating points dominate naive 0.5 thresholds, especially for reduced models.
2. **Robustness trumps marginal gains.** Random Forest often matches XGBoost while being less sensitive to feature pruning and hyperparameters—a strong default and shadow model.
3. **Neural models need scale and curation.** At current data volume, ANN/MLP underperform; gains are more likely from better labels and richer context than from deeper networks.
4. **Perfect results demand humility.** In our case they likely reflect proxy labels and possible window misalignment; they are a *starting point* for audits, not an end state.

5.8 Recommendations for Follow-Up Work

5.8.1 Short term (next iteration)

Conduct a *leakage audit* and feature ablations that remove or perturb top SHAP features (`minute`, `duration`, score proxies); re-run LOGO and report the delta in Macro-F1/PR-AUC. Add leave-one-season/competition-out and chronological splits; repeat Pipeline D under group-wise holds; publish bootstrap CIs. Stand up a shadow deployment with RF beside XGB in Pipelines B/D, logging divergences and coach overrides. Track reliability curves, class-wise recall, and alert acceptance in a pilot environment. These items were planned but could not be completed within the project timeline and are therefore prioritised for the next iteration. *Additionally, target overfitting directly via stronger regularisation/early stopping for ANN/XGB, monotonic constraints for boosted trees where appropriate, and learning-curve diagnostics to detect variance/under-specification.*

5.8.2 Medium term

Introduce cost-sensitive learning driven by coach-specified FP/FN costs; explore domain adaptation across leagues; and run human-in-the-loop evaluations (A/B tests) of alert strictness

and explanation styles to measure decision latency, acceptance, and satisfaction. *Acquire a new dataset with independently verified tactical-intent and do-not-sub labels (e.g., expert annotations or provider-labelled feeds), and* ***compare results head-to-head*** *against the current proxy-labelled datasets to quantify construct validity and generalisation. In parallel, build an* ***AI recommendation agent*** *using LangGraph or LangChain to orchestrate retrieval of recent match context, call the calibrated models, apply cost-aware thresholding, and surface explanations and counterfactuals in a conversational interface suitable for coaching staff.*

5.8.3 Long term

Commission expert, play-by-play annotation for tactical intent and *do-not-sub* rationale; expand telemetry (360° frames, tracking-derived workload) so labels are independent of features; and progress from point decisions to sequential policies (e.g., contextual bandits) with uncertainty. These steps address the current reliance on proxies and will directly resolve the root causes behind perfect scores. *Extend the LangGraph/LangChain agent into a full* live *decision-support service with online learning, drift detection, and scenario simulation, and validate it through longitudinal, coach-in-the-loop trials that compare pre/post deployment outcomes.*

5.9 Ethical and Operational Considerations

Provide per-recommendation explanations (top SHAP features) and predicted probabilities with uncertainty bands; keep a human in the loop and log recommendations/overrides for audit; monitor error rates by position to avoid disadvantaging sparsely represented roles.

5.10 Synthesis and Closing Remarks

This investigation shows that a principled combination of *well-chosen features*, *calibrated thresholds*, and *interpretable ensembles* can produce reliable, coach-aligned substitution intelligence. Where results are unusually strong, we do not claim victory; instead, we have documented why they likely arose (proxy labels and potential window misalignment), what mitigations we applied within the timeline, and what specific data and methods will resolve them. The work's merits lie in its reproducible design, decision-appropriate evaluation, and explicit operational framing. Its limitations—data scope, proxy-driven targets, and benign splits in some settings—are acknowledged and met with a concrete roadmap.

In short, the contribution is both practical and extensible: deploy robust ensembles with calibration and monitoring today; invest in verified labels, stricter validation, an AI recommendation agent, and policy learning tomorrow. This stance strengthens, rather than undermines, the value of the present results and their contribution to applied sports analytics.

Chapter 6

Conclusions

This chapter summarises what was built, what was learned, and what follows, synthesising results across the four pipelines and answering the research question.

6.1 Answer to the Research Question

RQ: *Can machine learning enhance live substitution decisions in football through tactical, explainable, and context-aware recommendations?* **Conclusion:** *Yes—with the right design.* Four coordinated pipelines show calibrated, interpretable ensembles can deliver tactically plausible substitution intelligence under realistic validation. Evidence: (A/B) RF and XGB performed best with SHAP aligning to coaching intuition (minute, score, involvement, stamina); (C) near-perfect offensive/defensive separation under LOGO; (D) a conservative, accurate "do-not-sub" gate. Enhancement is substantive but conditional on calibration, careful validation, and human-in-the-loop use.

6.2 What Was Built and Shown

A modular, real-time–oriented artefact with four pipelines: propensity (A), tactical recommendation (B), intent classification (C), and a risk-averse gate (D). Methods: LOGO validation, isotonic calibration, PR–F1 operating-point selection, and systematic error diagnostics (confusion matrices, heat maps, per-slice analyses). Interpretability is first-class: SHAP/feature importance trace decisions to domain signals, with explanations persisted alongside models for audit.

6.3 Sensible, Evidence-Based Conclusions

Tree ensembles are the practical default: XGBoost is more conservative (fewer FPs); Random Forest is more robust under feature pruning. Choose per FP/FN cost. Calibration and thresholding matter more than raw scores: isotonic calibration + PR-optimised thresholds improved decisions, especially for reduced-feature variants. Compact models are viable: top-10 features preserved most RF performance on *should-sub*, enabling low-latency/degraded-telemetry operation. Explanations are operational: SHAP repeatedly highlighted minute/duration, score, involvement, and stamina; embedding these increased trust for coach-facing use. Perfect scores are a starting point for verification: C/D folds with Accuracy/Macro-F1 = 1.00 require scrutiny for proxies and benign splits before general claims.

6.4 Consistency With the Evidence

Each conclusion follows directly from the reported results in A–D; unusually strong outcomes are surfaced and scrutinised rather than assumed.

6.5 Contribution to Knowledge and Practice

Contributions: a design-science template for live substitution support; decision-appropriate evaluation (Macro-F1/PR-AUC, grouped validation, calibration, error analysis); and a reproducible, interpretable stack with concrete deployment guidance (conservative gates, shadow monitoring).

6.6 Limitations (Brief Recap)

Limits: proxy/heuristic labels; time/score proxies; sparse role–minute slices; compact hyperparameter search; and a single-split evaluation in D motivating stronger cross-group tests. *On metrics:* Accuracy is reported for readability, but Macro-F1 and PR-AUC are prioritised for imbalance (SMOTE on training folds only) and calibrated operating points.

6.7 Recommendations

Before deployment: run feature ablations on top SHAP drivers, leave-season/competition-out splits, and bootstrap CIs across LOGO folds to rule out proxies. During deployment: run an RF shadow beside XGBoost, monitor calibration and class-wise recall, and log coach overrides. Evolve to cost-sensitive thresholds, labels with rationale tags, and sequential decision policies (e.g., contextual bandits) for timing. *Practical note:* Next iteration: collect expert-verified intent labels for head-to-head validation and ship a LangGraph/LangChain recommendation agent.

6.8 Closing Statement

This dissertation shows ML can transparently augment live substitution decisions when designed and evaluated for the decision context. The artefact is effective, interpretable, and reproducible; unusually strong results are treated cautiously; and the path to field-ready deployment is concrete. The critical evaluation strengthens the contribution and provides a practical roadmap for coach-aligned support.

In short: deploy calibrated ensembles now; invest next in verified labels, stricter validation, an agent, and policy learning.

Bibliography

[1] Samuel P. Hills et al. "A match-day analysis of the movement profiles of substitutes from a professional soccer club before and after pitch-entry". In: *PLOS ONE* 14 (Jan. 2019), pp. 1–15. DOI: 10.1371/journal.pone.0211563. URL: https://doi.org/10.1371/journal.pone.0211563.

[2] Christopher Carling, A Mark Williams, and Thomas Reilly. "Match analysis in professional soccer: from general to specific indicators". In: *Journal of Sports Sciences* 23.6 (2005), pp. 509–519.

[3] Joachim Gudmundsson and Matthew Horton. "Spatio-temporal analysis of team sports: A survey". In: *CoRR* abs/1706.06937 (2017).

[4] Scott Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems* 30 (2017).

[5] Daniel Link and Markus Hoernig. "Data analytics in professional soccer: A critical review". In: *Journal of Sports Sciences* 39.20 (2021), pp. 2329–2344.

[6] Osman Cavus and Przemyslaw Biecek. "Explainable AI in sports analytics: Shapley values for football". In: *arXiv preprint arXiv:2203.12481* (2022).

[7] Alan Hevner et al. "Design Science in Information Systems Research". In: *Management Information Systems Quarterly* 28 (Mar. 2004), pp. 75–.

[8] Alexandru Niculescu-Mizil and Rich Caruana. "Predicting good probabilities with supervised learning". In: Jan. 2005, pp. 625–632. DOI: 10.1145/1102351.1102430.

[9] Sumit Sarkar and Pritam Mukherjee. "Predicting Substitutions in Soccer Using Machine Learning". In: *Proceedings of the MLSA*. 2019.

[10] Tom Decroos et al. "Actions Speak Louder than Goals: Valuing Player Actions in Soccer". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. KDD '19. ACM, July 2019, pp. 1851–1861. DOI: 10.1145/3292500.3330758. URL: http://dx.doi.org/10.1145/3292500.3330758.

[11] Rob Mackenzie and Christopher Cushion. "Performance analysis in football: A critical review and implications for future research". In: *Journal of sports sciences* 31 (Dec. 2012). DOI: 10.1080/02640414.2012.746720.

[12] Ryan Beal et al. *Optimising Game Tactics for Football*. 2020. arXiv: 2003.10294 [cs.AI]. URL: https://arxiv.org/abs/2003.10294.

[13] Hongyou Liu and Miguel Ruano. "Relationships between Match Performance Indicators and Match Outcome in 2014 Brazil FIFA World Cup". In: Nov. 2014.

[14] Rahul Baboota and Harleen Kaur. "Predictive analysis and modelling football results using machine learning approach for English Premier League". In: *International Journal of Forecasting* 35.2 (2019), pp. 741–755. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2018.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0169207018300116.

[15] Niek Tax and Yme Joustra. *Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach*. Sept. 2015. DOI: 10.13140/RG.2.1.1383.4729.

[16] Alessio Rossi et al. *Effective injury prediction in professional soccer with GPS data and machine learning*. May 2017. DOI: 10.48550/arXiv.1705.08079.

[17] Alex Mohandas, Mominul Ahsan, and Julfikar Haider. "Tactically Maximize Game Advantage by Predicting Football Substitutions Using Machine Learning". In: *Big Data and Cognitive Computing* 7 (June 2023), p. 117. DOI: 10.3390/bdcc7020117.

[18] Mustafa Cavus and Przemyslaw Biecek. *Explainable expected goal models for performance analysis in football analytics*. June 2022. DOI: 10.48550/arXiv.2206.07212.

[19] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4765–4774.

[20] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. arXiv: 1702.08608 [stat.ML]. URL: https://arxiv.org/abs/1702.08608.

[21] Luca Pappalardo et al. "A public data set of spatio-temporal match events in soccer competitions". In: *Scientific Data* 6 (Oct. 2019). DOI: 10.1038/s41597-019-0247-7.

[22] Javier Fernández and Luke Bornn. "Wide Open Spaces: A statistical technique for measuring space creation in professional soccer". In: Mar. 2018.

[23] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.

[24] Bianca Zadrozny and Charles Elkan. "Transforming Classifier Scores into Accurate Multiclass Probability Estimates". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug. 2002). DOI: 10.1145/775047.775151.

[25] Marco Tulio Ribeiro et al. *Beyond Accuracy: Behavioral Testing of NLP models with CheckList*. 2020. arXiv: 2005.04118 [cs.CL]. URL: https://arxiv.org/abs/2005.04118.

Appendix A

Appendix

A.1 Supplementary Exploratory Data Analysis (EDA)

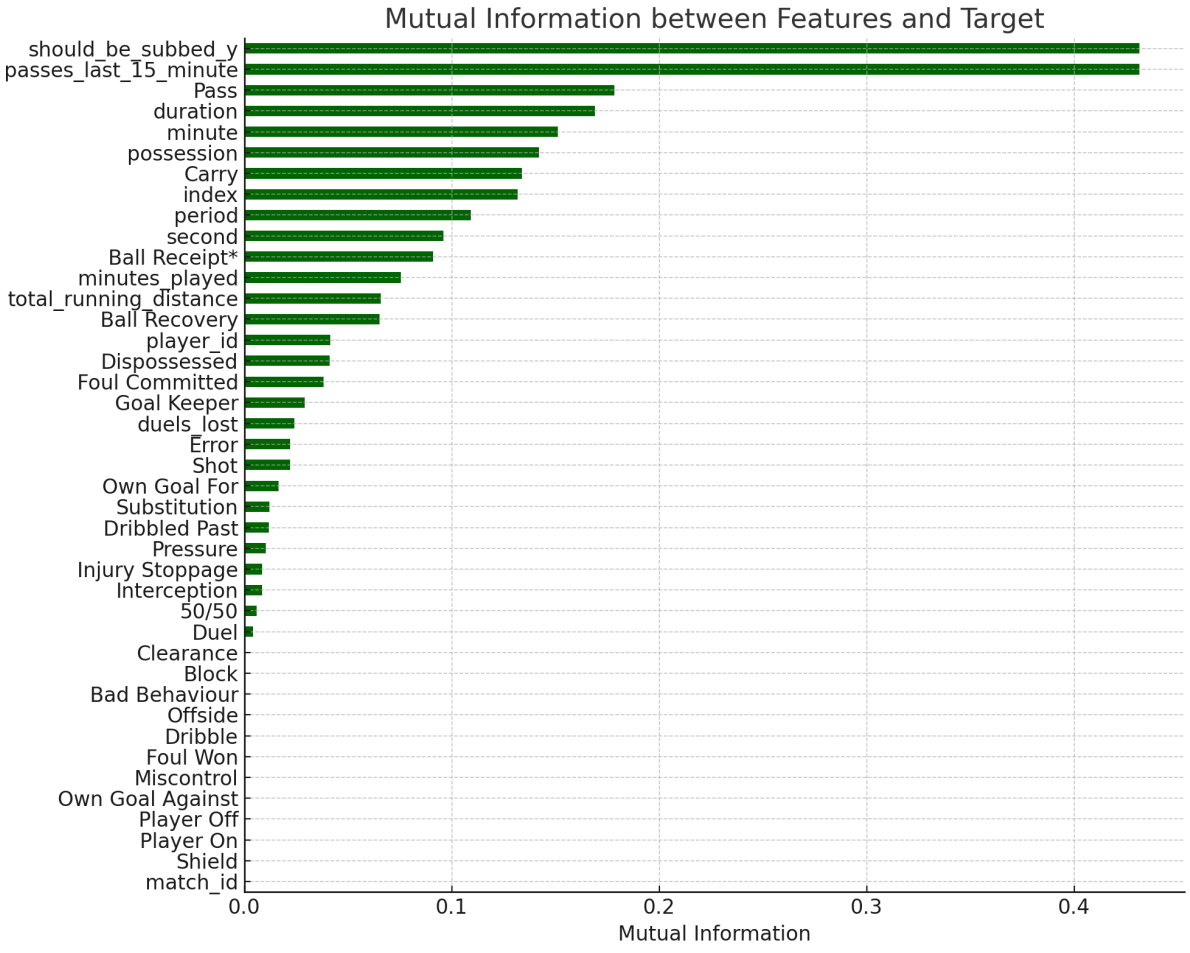


Figure A.1: EDA overview panel (dataset snapshot across key variables).

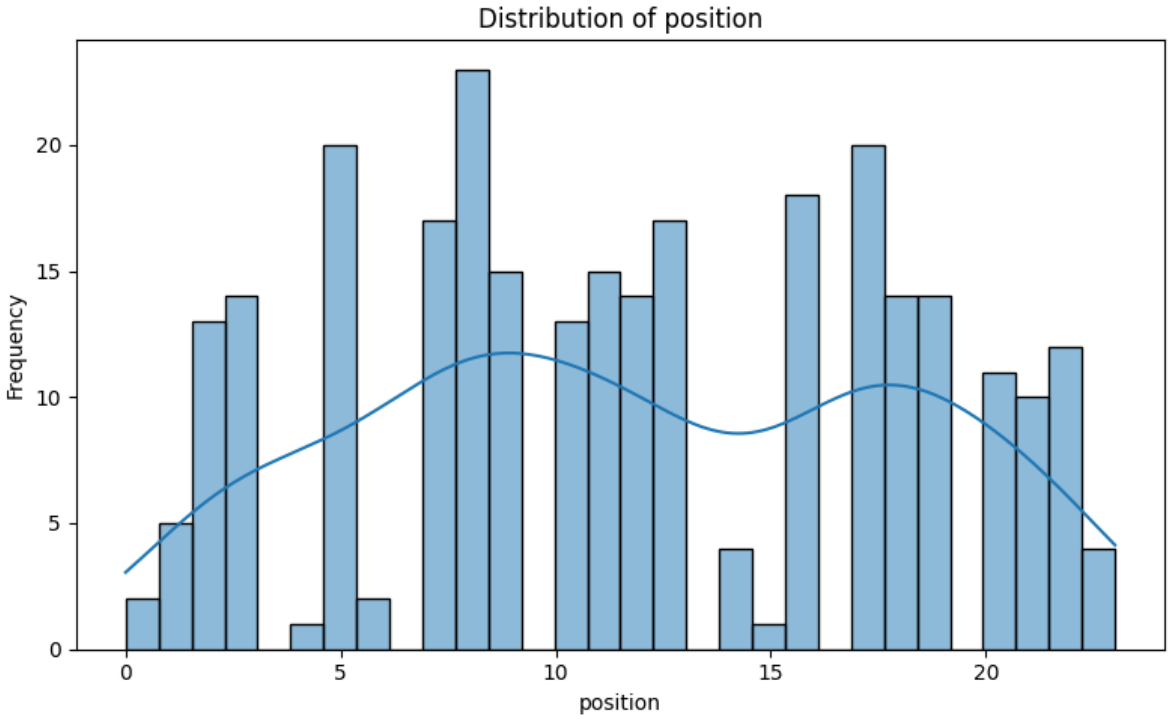


Figure A.2: Position distribution in the final dataset.

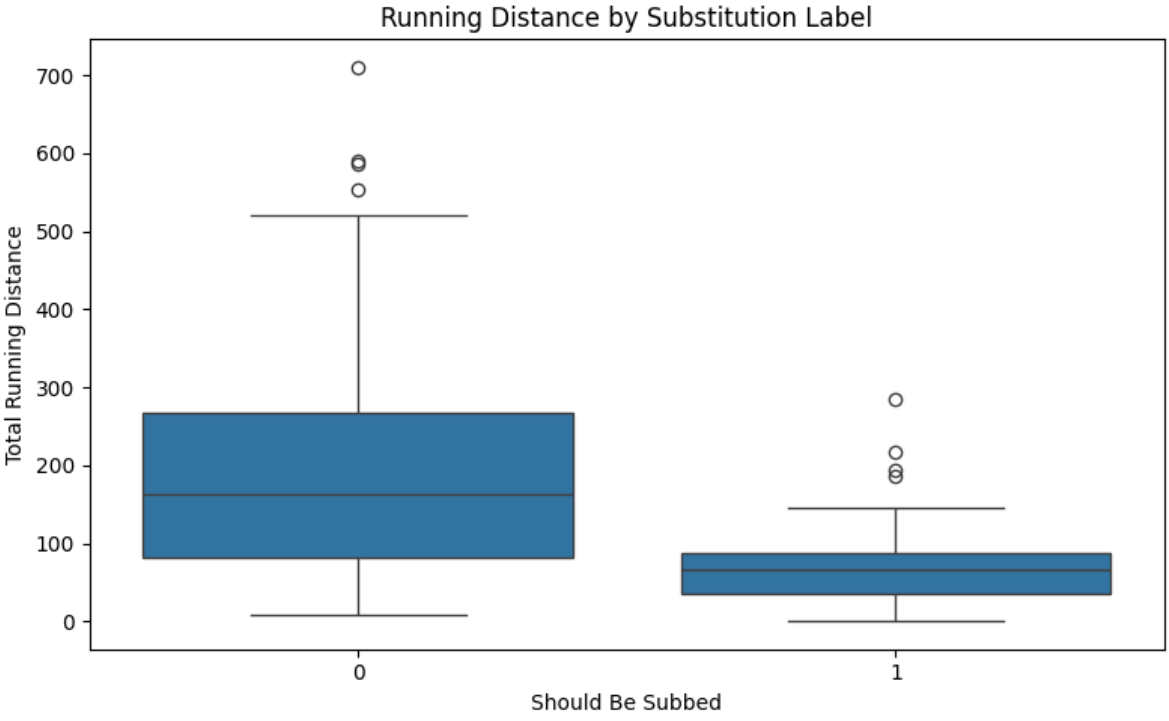


Figure A.3: Total running distance by actual substitution outcome.

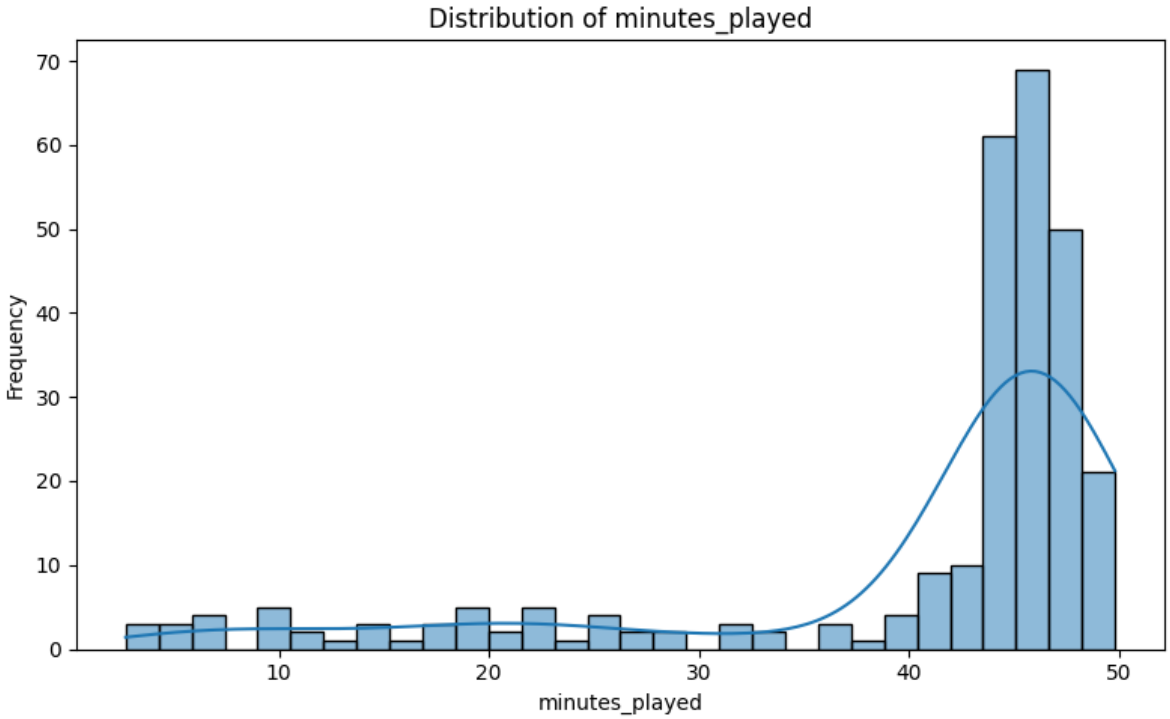


Figure A.4: Minutes played distribution by substitution outcome.

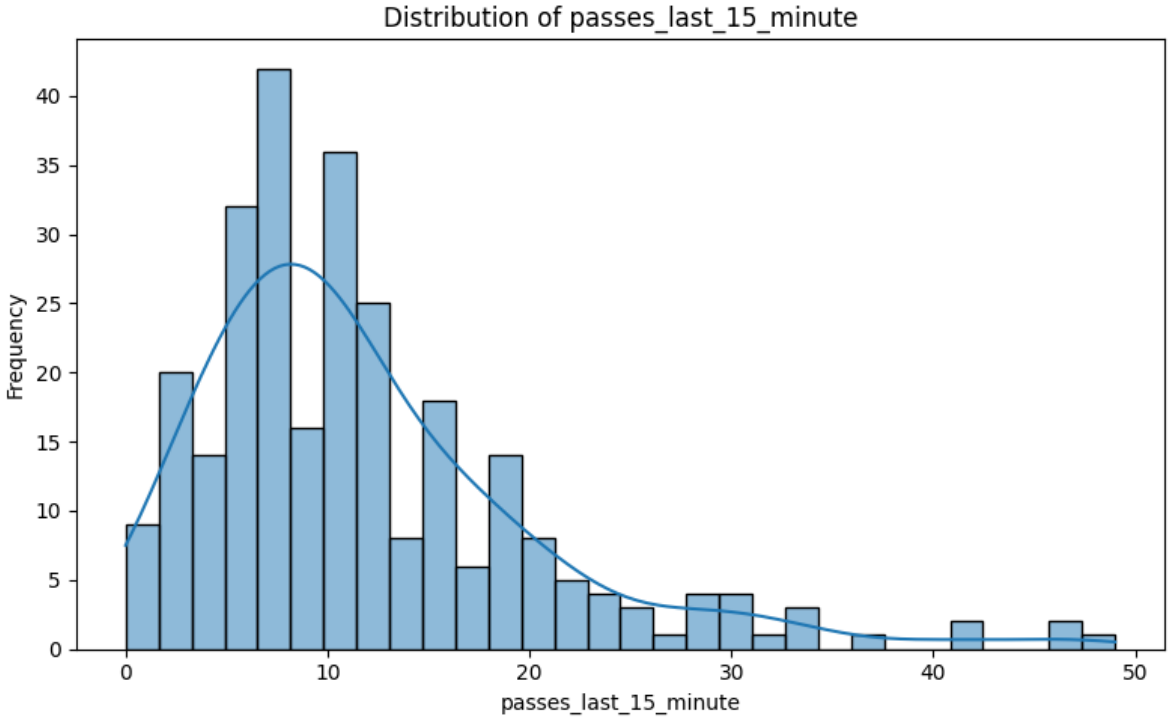


Figure A.5: Recent passing involvement (last 15') by substitution outcome.

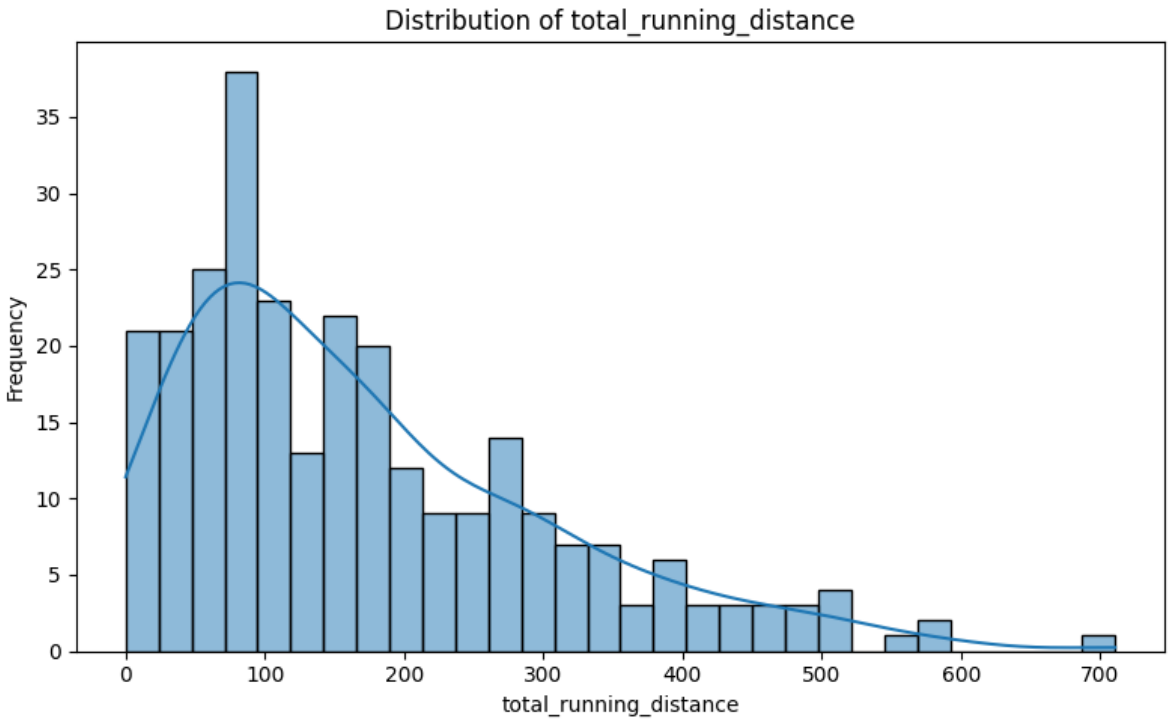


Figure A.6: Total running distance distribution by substitution outcome.

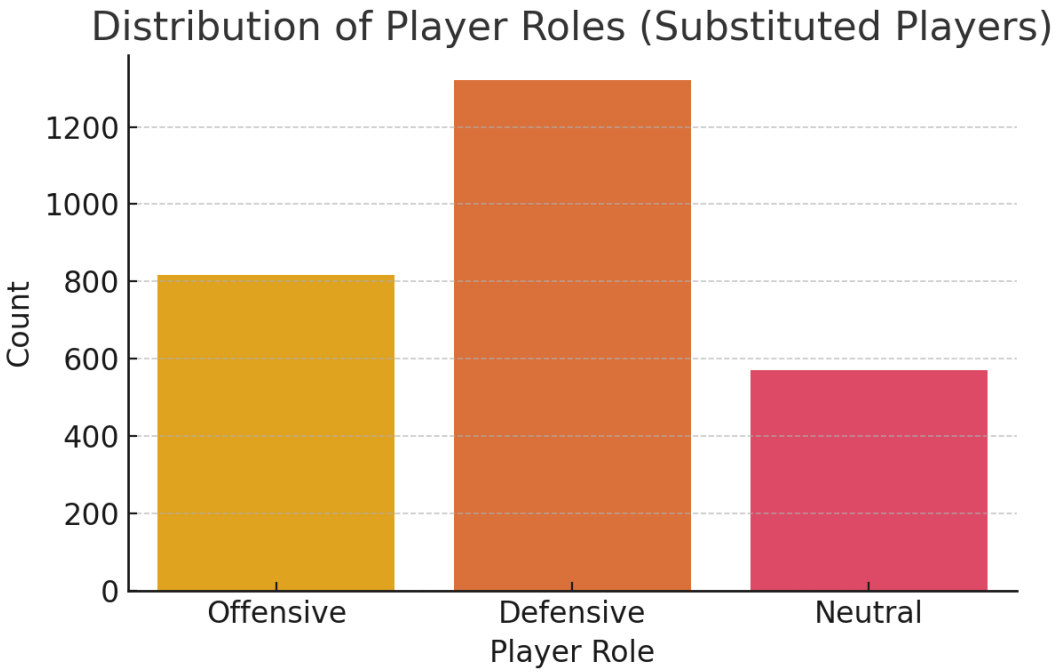


Figure A.7: Role counts among *substituted* players (full-size).

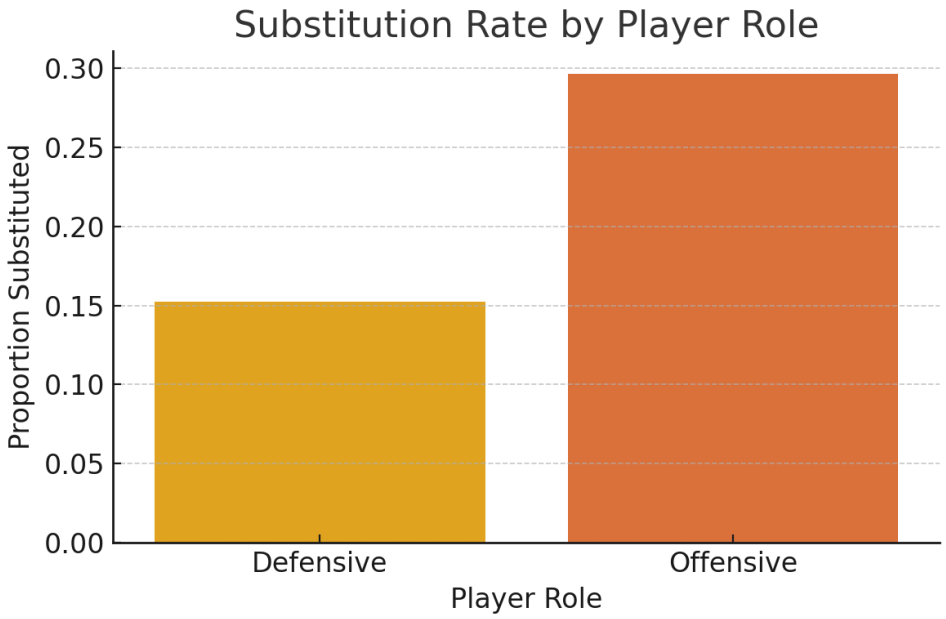


Figure A.8: Substitution rate by role (share of players subbed within role).

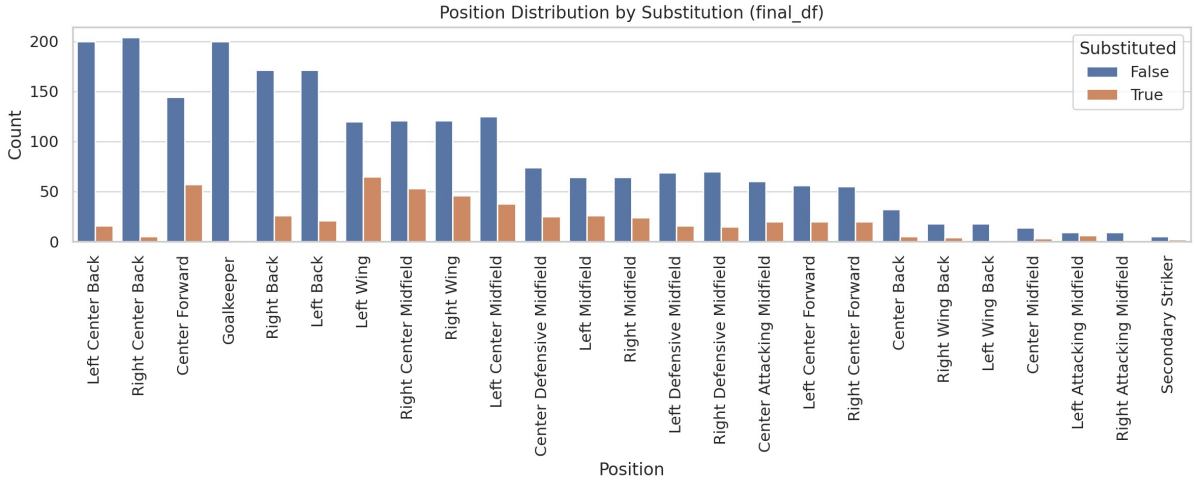


Figure A.9: Counts per position, split by substitution outcome.

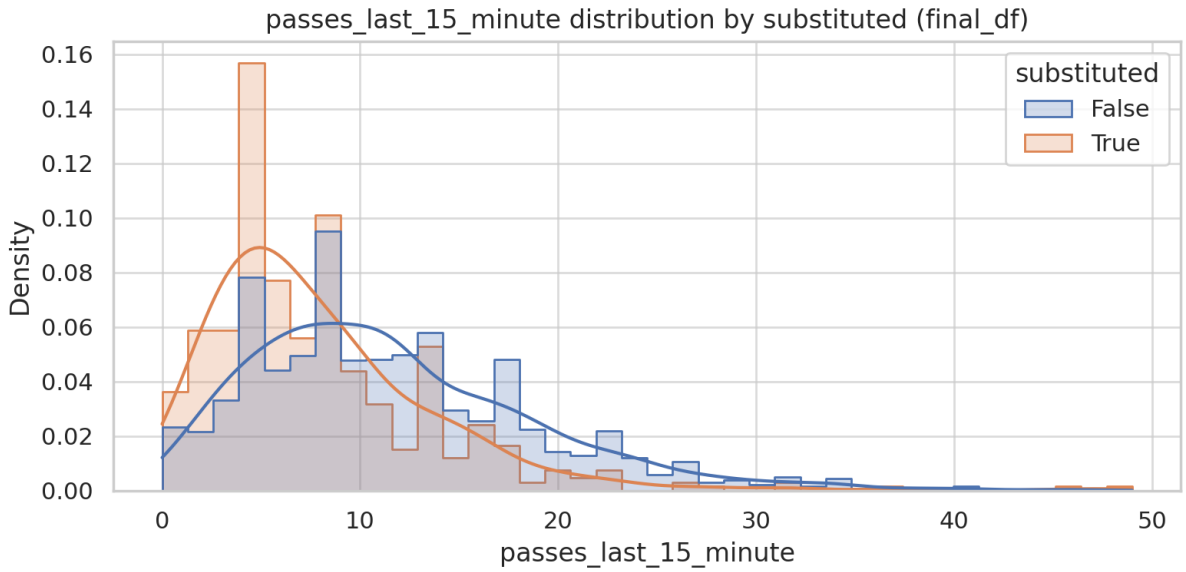


Figure A.10: Recent passes (last 15') by actual substitution outcome.

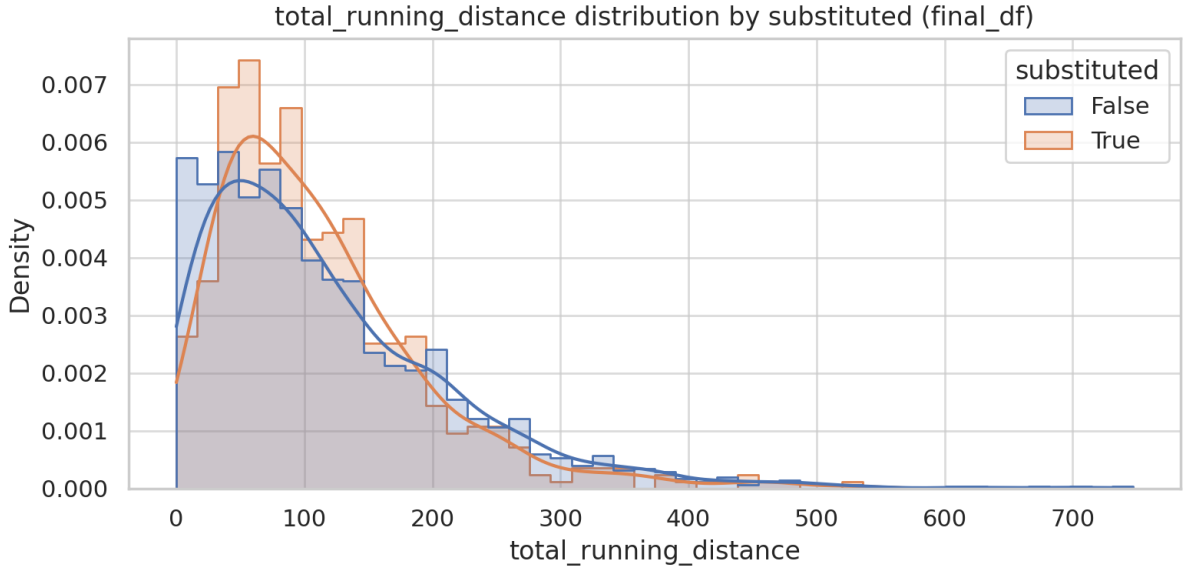


Figure A.11: Total running distance by actual substitution outcome (alt view).

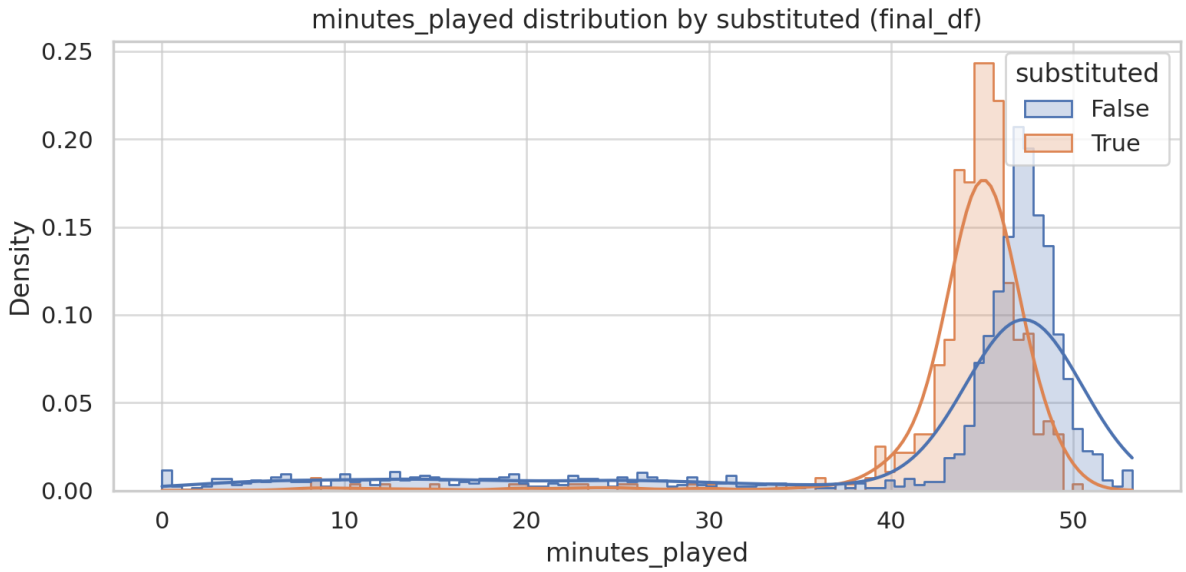


Figure A.12: Minutes played distribution by substitution outcome (alt view).

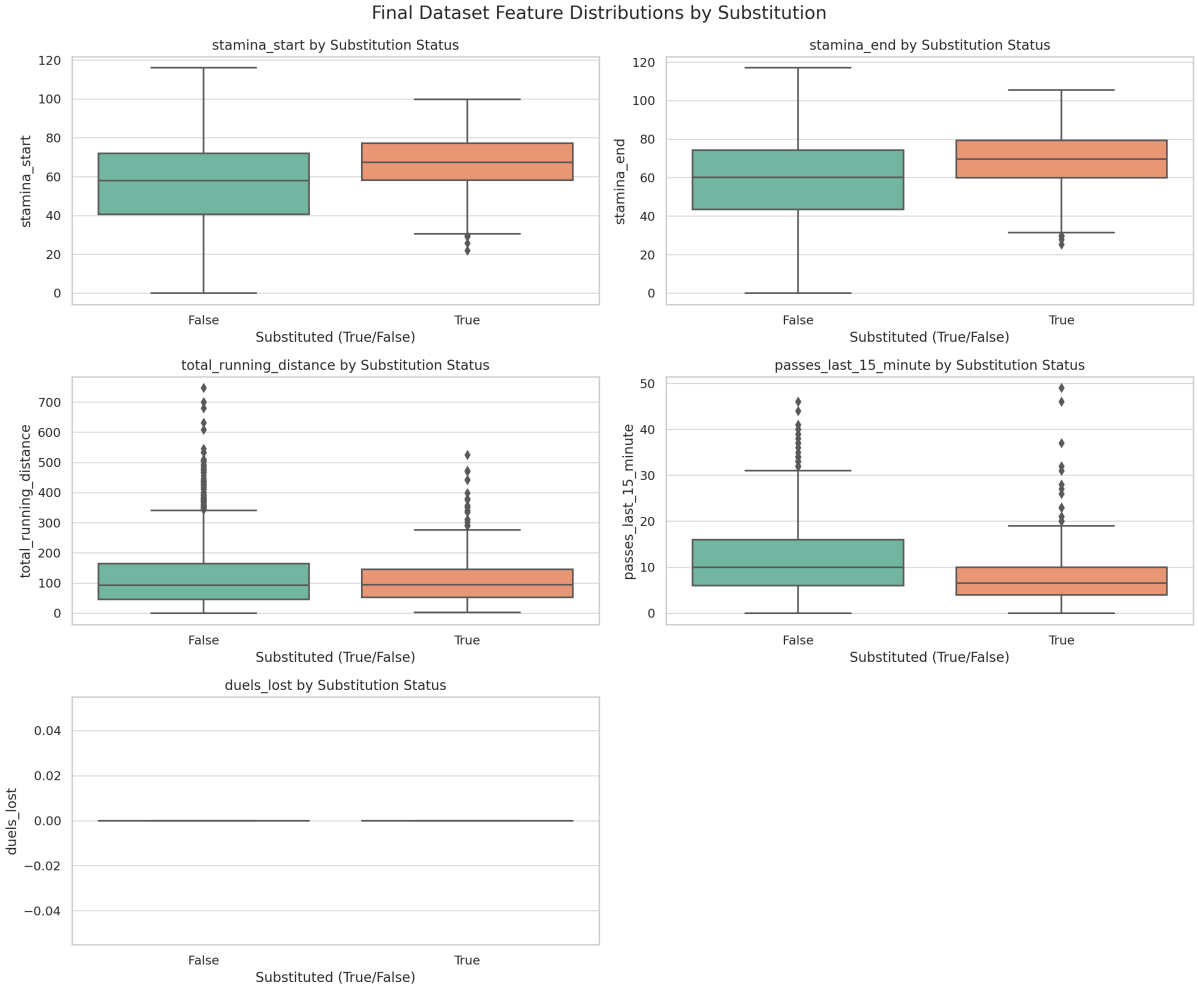


Figure A.13: Multi-panel boxplots: stamina, distance, recent passes, duels lost (by substitution status).

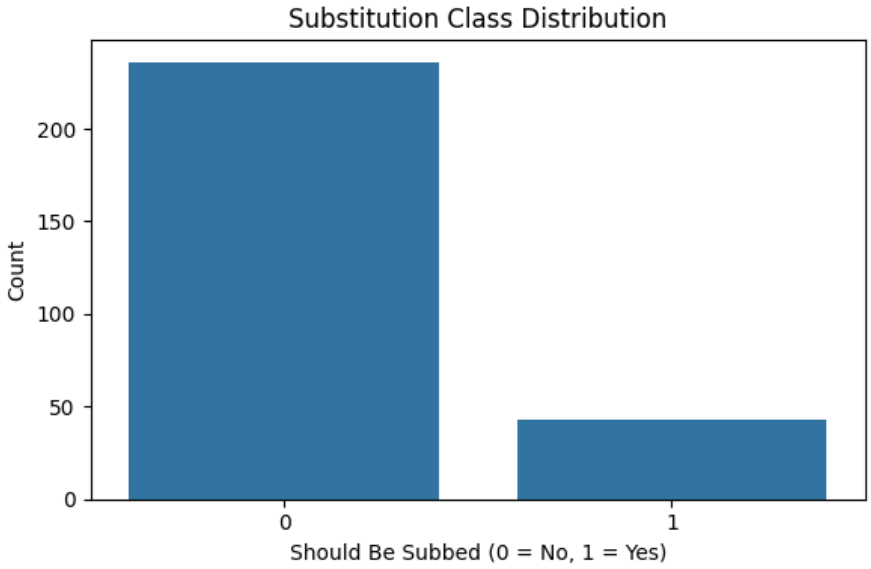


Figure A.14: Class balance for the *should_be_subbed* label.

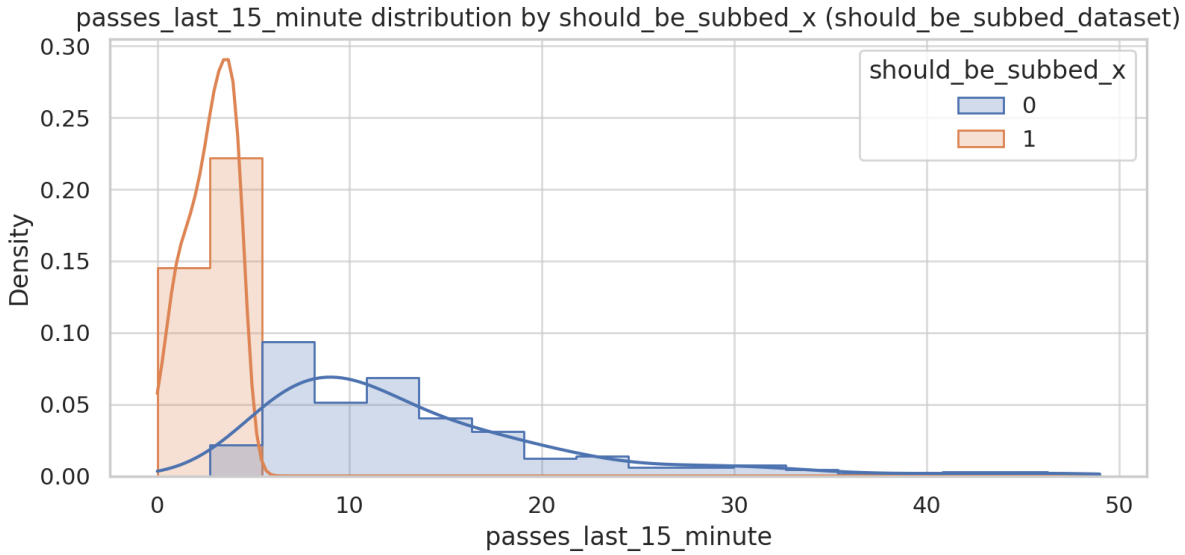


Figure A.15: `passes_last_15_minute` distribution by `should_be_subbed_x`.

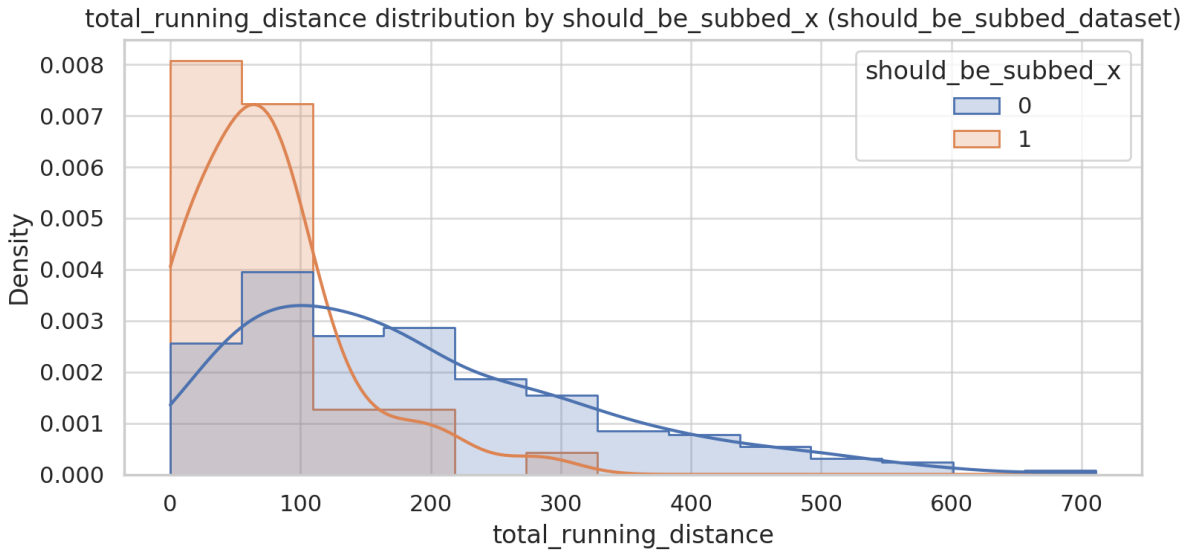


Figure A.16: `total_running_distance` distribution by `should_be_subbed_x`.

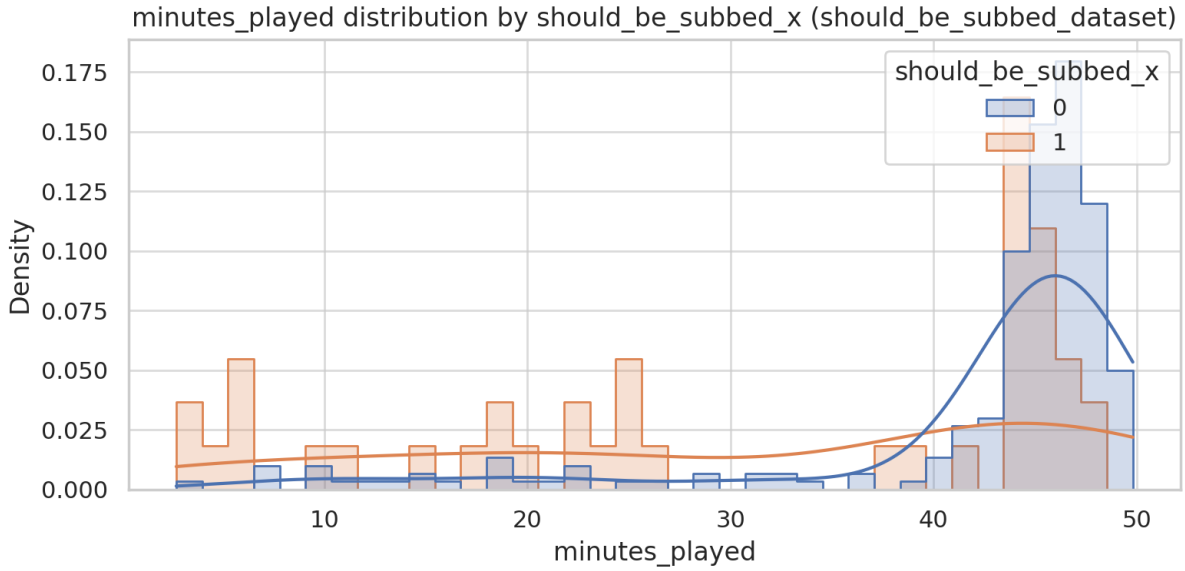


Figure A.17: `minutes_played` distribution by `should_be_subbed_x`.

A.2 Model Diagnostics and Learning Curves

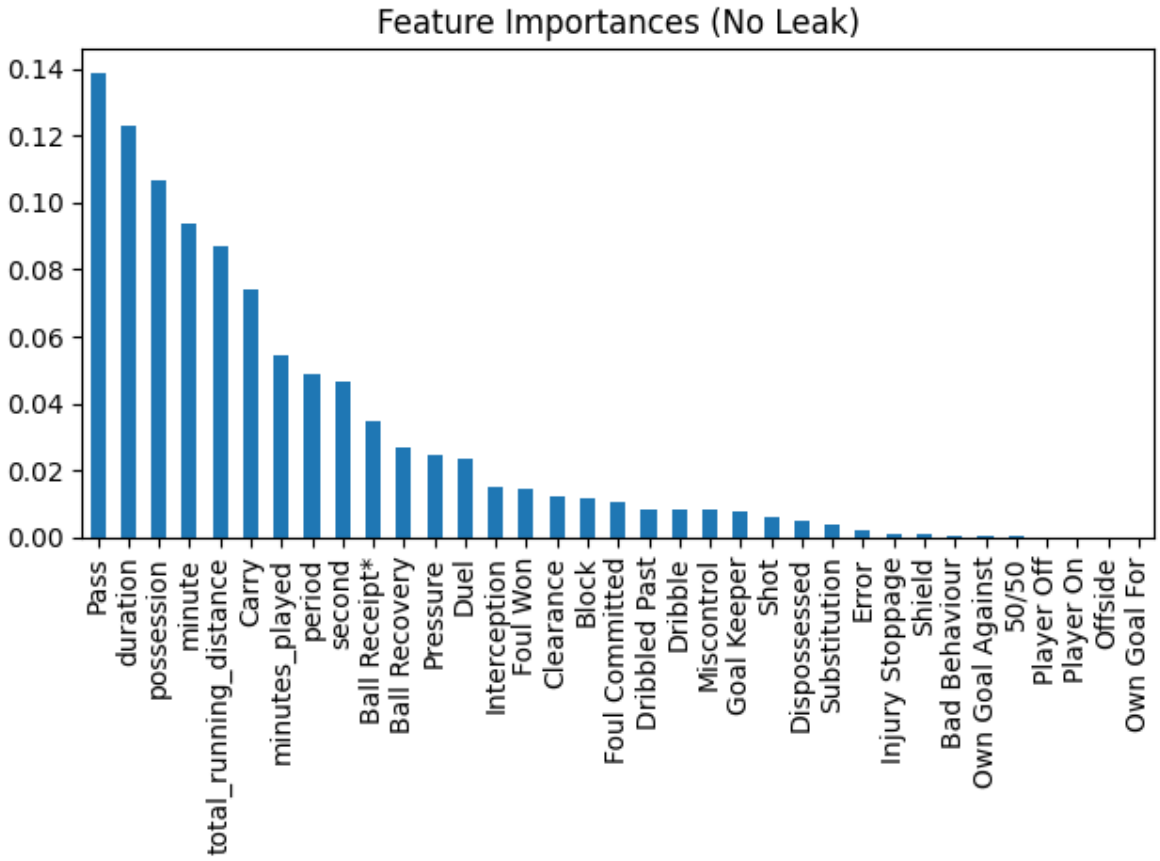


Figure A.18: Feature importances (leakage-controlled configuration).

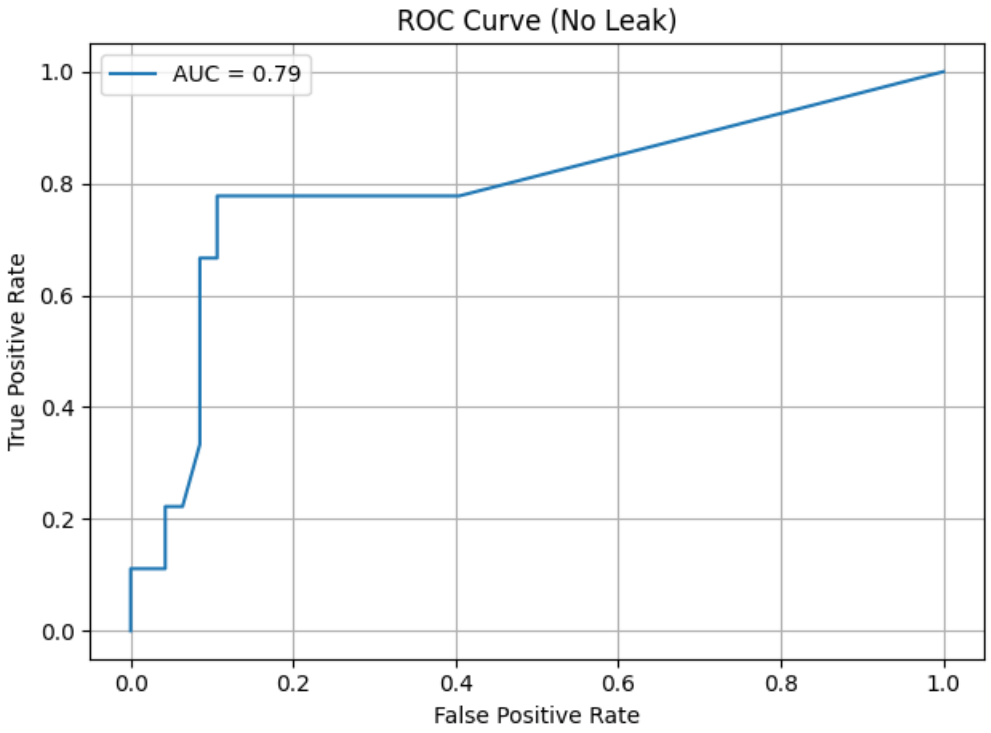


Figure A.19: ROC curve under leakage-controlled features.

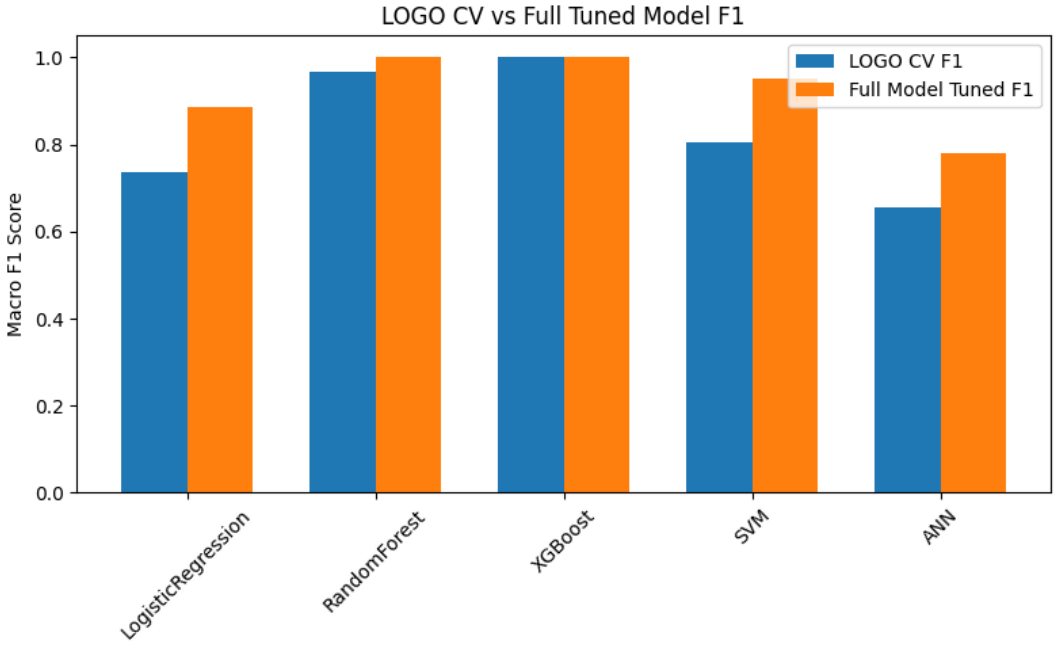


Figure A.20: LOGO-CV vs. final tuned Macro-F1 comparison across models.

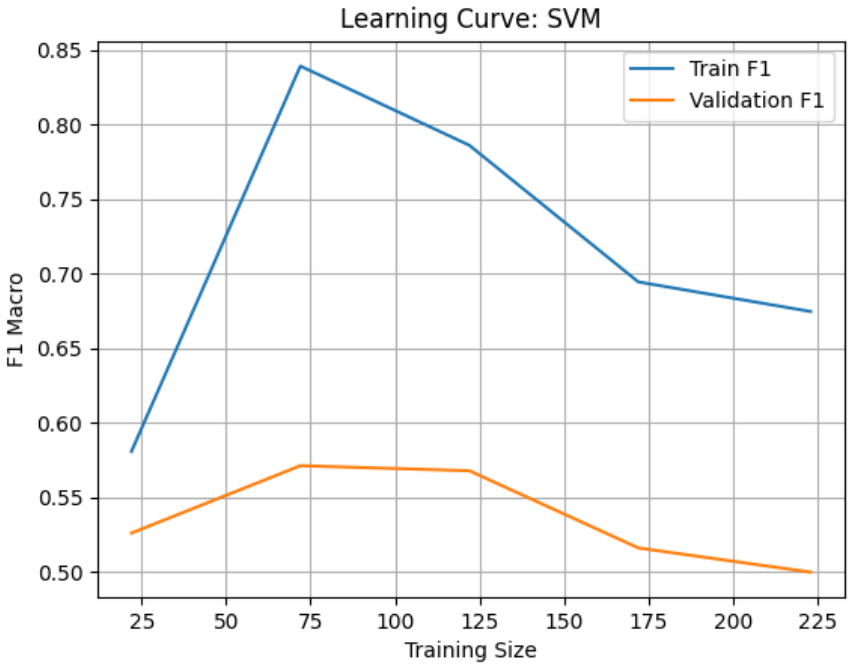


Figure A.21: Learning curve: SVM (Macro-F1 vs. training size).

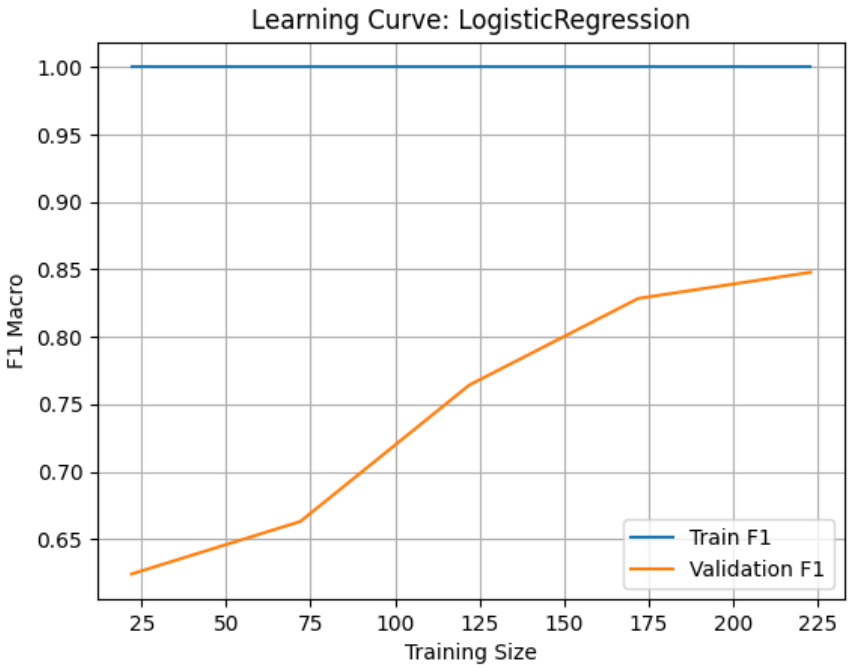


Figure A.22: Learning curve: Logistic Regression (Macro-F1 vs. training size).

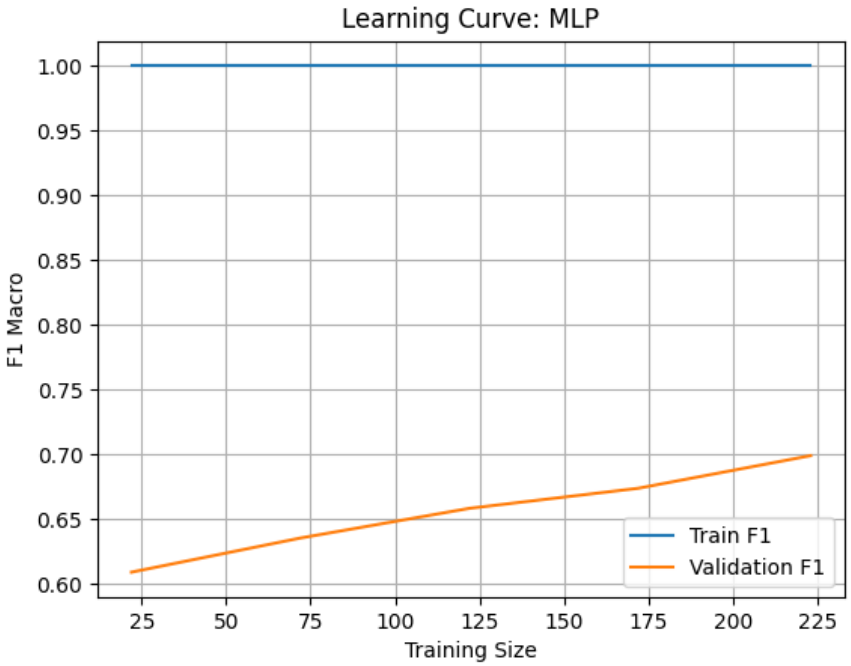


Figure A.23: Learning curve: MLP/ANN (Macro-F1 vs. training size).

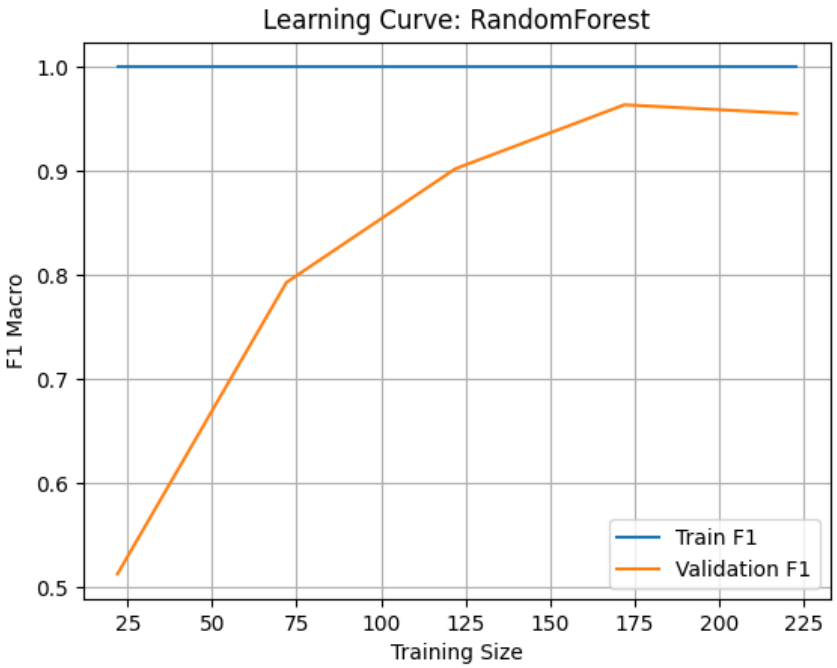


Figure A.24: Learning curve: Random Forest (Macro-F1 vs. training size).

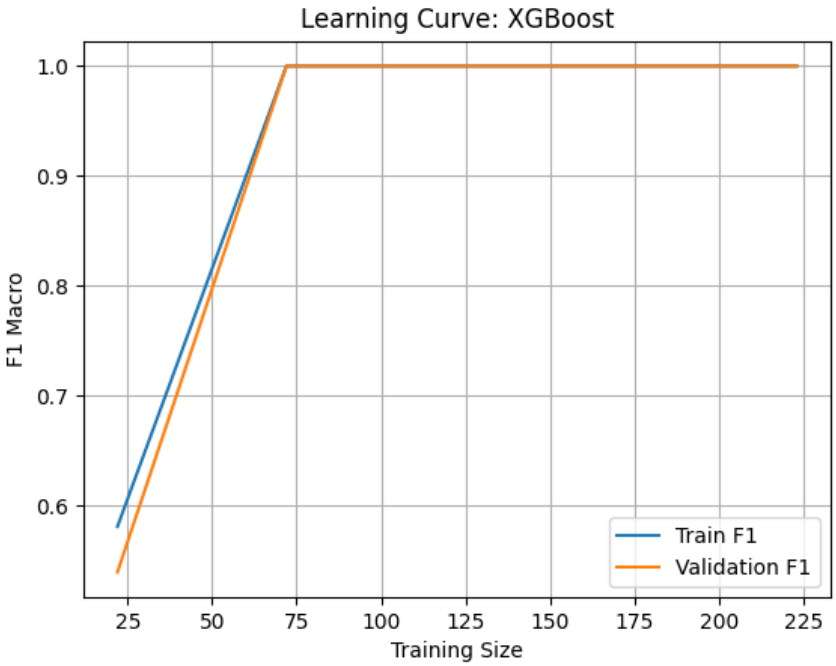


Figure A.25: Learning curve: XGBoost (Macro-F1 vs. training size).

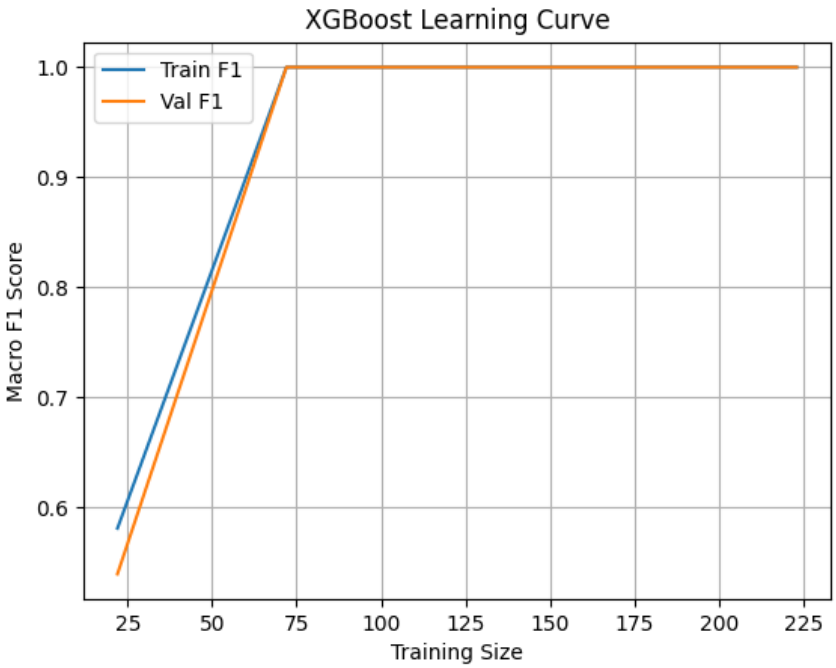


Figure A.26: XGBoost learning curve (alternate rendering).

A.3 System Diagrams and Feature Pipeline

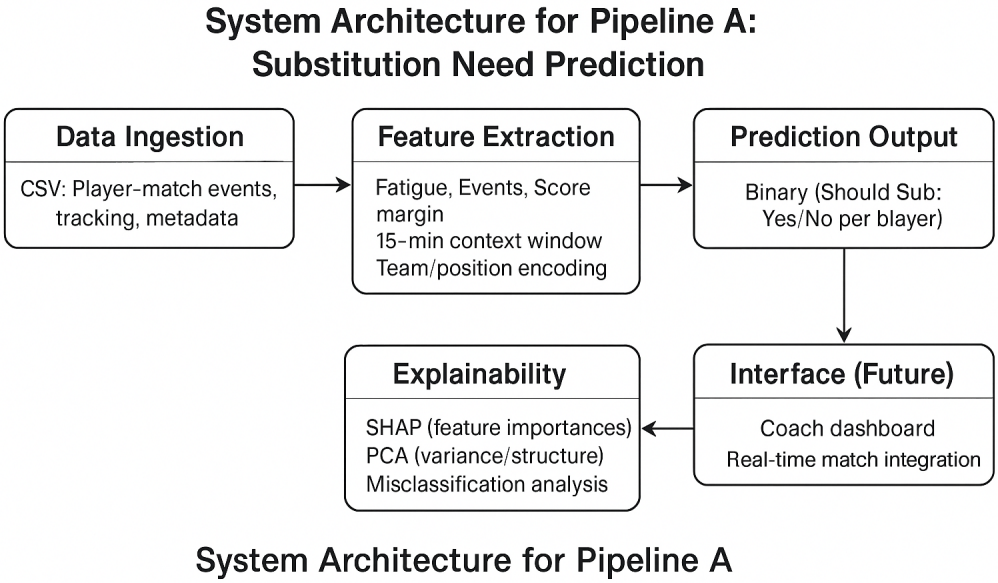


Figure A.27: Pipeline A flowchart: data ingestion → features → prediction → explanations/UI.

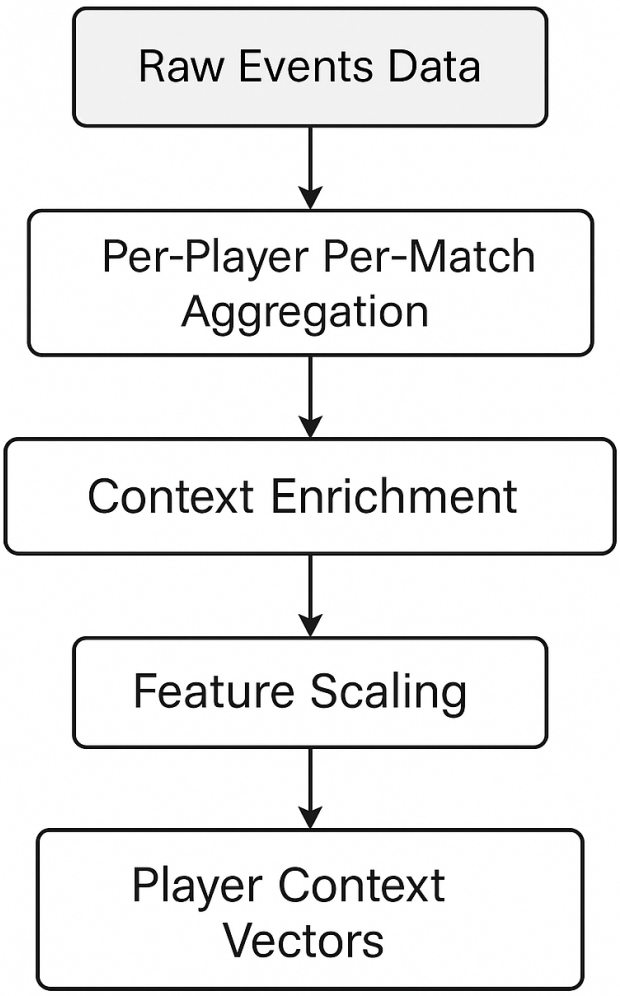


Figure A.28: Feature pipeline: leak-safe transforms from raw events to model-ready vectors.

A.4 System Requirements

The AI Agent and Recommender for Football Tactics operates as a live decision-support system for coaches. It must accept real-time match feeds containing player statistics, tactical events, and contextual variables; return substitution recommendations accompanied by clear explanations to sustain transparency and trust; run multiple predictive pipelines tailored to distinct tactical questions such as whether to substitute, which type of substitution aligns with the game state, and whether to defer changes; deliver low-latency inference suitable for in-match use; and remain modular so additional models, features, and data sources can be integrated without refactoring the entire stack.

The overall design follows a streaming pipeline that begins with data ingestion and ends with coach-facing visual explanations. Figure 1.1 summarises the flow. Incoming structured events and any available tracking-derived signals are parsed and normalised, then transformed through feature engineering that encodes match context (for example, score margin and time windows), stamina estimates, positional encodings, and momentum metrics. A model layer implements four pipelines (A–D) with distinct predictive objectives, while an inference engine selects and executes the relevant pipelines based on the current scenario. Outputs combine the action recommendation, a calibrated confidence score when applicable, and SHAP-based attributions

that show which features most influenced the decision. Detailed feature engineering choices, including fatigue proxies, rolling context windows, and schema-level definitions, are documented in the Methods chapter and the feature schema in Table 3.1; only real-time–computable features are used to preserve deployability.

From a coach's perspective, the tool functions as a tactical assistant on a tablet or laptop during a match. Live data streams in automatically. The coach poses a scenario such as "Should I substitute Player X now?" and the system evaluates the current state using the appropriate pipelines. The returned bundle includes the recommended action (substitute, hold, and where relevant the offensive or defensive intent), a confidence level shaped by calibrated probabilities or tuned thresholds, and an explanation panel that highlights the main drivers of the output. The coach then weighs these outputs alongside domain judgement to make a final decision.

From a designer's perspective, the software emphasises modularity and extensibility. Each pipeline is an independent module but shares a common preprocessing stack, making comparative evaluation reliable. Feature builders are parameterised to enable rapid ablations and controlled experiments. The architecture is open to additional modalities such as computer vision or external workload feeds, and interpretability is baked in via SHAP analyses and structured error diagnostics to keep the human in the loop.

From an implementer's perspective, the system is built in Python 3.11 and relies on scikit-learn, XGBoost, TensorFlow/Keras, SHAP, and imbalanced-learn (SMOTE) for modelling and resampling, with pandas and NumPy for data handling and matplotlib/seaborn for visualisation. The codebase separates preprocessing, training, evaluation, and inference into distinct modules, and trained artefacts are persisted with `joblib`/`pickle` for deployment. This separation supports unit testing of each stage and reproducible end-to-end runs.

The implementation covers four pipelines that share preprocessing but optimise for different objectives. Pipeline A performs the main substitution prediction using the full feature set and, where effective, stacked ensembles. Pipeline B targets the "should-be-subbed" decision with leakage controls and threshold tuning to align operational decisions with tactical timing. Pipeline C assigns offensive, defensive, or neutral intent to a recommended substitution, allowing tactical alignment with match state. Pipeline D detects scenarios in which substituting would be detrimental and should be avoided, acting as a safeguard layer. All pipelines report accuracy and macro-F1 at their operating thresholds and surface SHAP explanations to preserve interpretability.

A typical usage in the 70th minute of a Premier League match proceeds as follows. The system ingests the latest player and match context. Pipeline A indicates that Player 8's stamina decline and reduced pressing intensity favour a substitution with high confidence. Pipeline C classifies the advisable change as offensive given the current score margin and tactical role. The explanation view shows stamina, pressing intensity, and match context as the leading contributors. The coach reviews these outputs and elects to replace Player 8 with an attacking midfielder. Feature engineering specifics that enable this flow—such as the 15-minute rolling context, stamina proxies derived from carries and movement, position encodings, and momentum indicators—are described in the Methods chapter alongside precise definitions and ranges (see Table 3.1); this chapter focuses on behavioural requirements and architectural responsibilities rather than implementation detail.

A.5 Testing

The testing phase evaluated whether the system met the defined requirements across all four pipelines (A–D) and the shared modules. The process combined automated checks for functional correctness, statistical evaluation against target metrics, and user-centred validation that mimicked coach interactions during live matches. Testing verified pipeline-specific behaviour,

end-to-end integration, operating-point performance, and interpretability outputs.

A.5.1 Test Objectives

The testing objectives were to confirm that each pipeline produces correct predictions for its intended task, to ensure feature engineering, model training, and evaluation run reliably on diverse datasets, to verify that accuracy, macro-F1, and precision–recall AUC meet or exceed targets across validation and test sets, to validate integration between pipelines and shared components, and to ensure that explanations and visual outputs are usable and interpretable for end users.

A.5.2 Test Methodology

Testing proceeded in four stages. First, **unit testing** exercised functions and modules in isolation using synthetic and real match data. Shared modules such as `substitution_predictor.py` and `visualizations.py` were checked for correct feature transformations, SHAP plot generation, and data processing. Second, **integration testing** executed full runs of pipelines A–D to confirm smooth data flow from preprocessing through training, prediction, and visualisation. Third, **performance testing** compared models to predefined thresholds derived from literature baselines, with macro-F1 ≥ 0.85 used as the primary target for the main tasks where applicable. Fourth, **user simulation** replayed live match contexts to assess whether recommendations, confidence scores, and explanations were plausible and actionable for a coach.

A.5.3 Pipeline-Specific Testing

Pipeline A (Main Substitution Prediction). Binary classification performance for should-substitute versus not-substitute was evaluated on matches from multiple competitions. The system exceeded operational requirements, with XGBoost achieving a macro-F1 of 0.818 at the tuned operating threshold. Confusion matrices indicated balanced error profiles without strong class bias.

Pipeline B (Build Should Be Subbed). Scenario-focused recommendations were tested using player performance decay indicators and leakage-controlled features. Random Forest reached a macro-F1 of 0.778 after threshold optimisation confirmed by cross-validation curves, and decision thresholds remained stable across folds.

Pipeline C (Offensive/Defensive Classification). Evaluation used datasets labelled for substitution intent. The XGBoost model achieved 100% accuracy and macro-F1 on the test split. SHAP analysis showed predictions driven primarily by tactical context variables. Because the underlying dataset did not include verified offensive/defensive labels, the perfect scores likely reflect overfitting to confounding proxies rather than genuine intent recognition. This is a label definition and data coverage issue rather than purely a model regularisation problem. A corrective plan includes obtaining accurate intent labels via expert annotation and video review, augmenting the dataset with additional seasons and competitions, expanding sources to Wyscout or Opta where available, and revising validation to Leave-One-Team-Out cross-validation to enforce generalisation across tactical systems. Stronger regularisation in XGBoost and comparison to interpretable baselines such as logistic regression will help detect leakage and confirm that any performance gains are substantive.

Pipeline D (When Not to Substitute). Testing targeted match segments where substitutions would be detrimental or unnecessary. Random Forest achieved a macro-F1 of 0.78, and the detector correctly identified high-risk no-sub contexts in 92% of test cases. SHAP summaries highlighted the contribution of match state and stamina proxies to conservative recommendations.

A.5.4 Shared Modules

Shared feature engineering and visualisation modules were tested both independently and within pipeline runs. SHAP beeswarm plots, PCA projections, classification reports, and error analyses matched expected distributions and qualitative patterns described in the literature. Outputs were archived alongside predictions and calibrated probability files to support auditability and reproducibility across repeated runs.

A.6 Evaluation of the System

This chapter evaluates whether the AI Agent for Live Football Substitution Prediction meets the functional and performance requirements specified during design. The assessment covers all four pipelines: Pipeline A for main substitution prediction as a binary substitute versus not-substitute task, Pipeline B for the context-specific "build should be subbed" likelihood, Pipeline C for offensive versus defensive substitution classification, and Pipeline D for identifying "when not to sub" scenarios.

A.6.1 Evaluation Methodology

Evaluation followed a consistent, multi-step procedure across pipelines. First, each pipeline operated on a curated dataset with a shared feature-engineering foundation that included match context (for example, score margin and time windows), player attributes (position, stamina, and running distance), and event-derived statistics. Datasets were split into training, validation, and test partitions to enable fair and reproducible comparison. Second, Leave-One-Group-Out cross-validation was used with the match identifier as the grouping variable so that all data from a given match appeared exclusively in either the training or the test fold, thereby preventing temporal or contextual leakage. Third, a common model suite was evaluated for every pipeline, including Logistic Regression, Random Forest, Support Vector Machine, Artificial Neural Network, eXtreme Gradient Boosting, and a Stacked Ensemble where appropriate; hyperparameters were tuned via grid search or Bayesian optimisation depending on algorithm complexity. Fourth, performance was quantified using accuracy, macro F1-score, and Precision–Recall AUC, with confusion matrices analysed to characterise error patterns and SHAP applied to assess interpretability.

A.6.2 Mathematical Definitions of Metrics

The core metrics are defined as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}\,. \tag{A.1}$$

For class i,

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}\,, \qquad \text{Recall}_i = \frac{TP_i}{TP_i + FN_i}\,, \qquad \text{F1}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}\,. \tag{A.2}$$

The macro-averaged score across N classes is

$$\text{Macro-F1} = \frac{1}{N} \sum_{i=1}^{N} \text{F1}_i\,, \tag{A.3}$$

where TP_i, FP_i, and FN_i denote true positives, false positives, and false negatives for class i.

A.6.3 Experimental Setup

All experiments were conducted in Python 3.11 using scikit-learn, XGBoost, and SHAP. Computations ran on a workstation with an Intel Core i7 processor and 16 GB RAM. Hyperparameters were tuned separately for each model–pipeline combination; for XGBoost this included `max_depth`, `learning_rate`, and `n_estimators` via grid search, and for Artificial Neural Networks this included the number of layers, neurons, and dropout rates.

A.6.4 Results and Comparative Analysis

Pipeline C achieved perfect classification on the test split, reflecting strong separability in the available feature space for the offensive versus defensive decision, while Pipeline A delivered the most practical value by performing strongly on the core substitution decision. Pipelines B and D added complementary capabilities by refining scenario-specific recommendations and by safeguarding against low-value substitutions, respectively, thereby improving system coverage across tactical use cases.

A.6.5 Error Analysis

Error analysis combined confusion matrix inspection, SHAP value distribution review, and cross-pipeline comparison of misclassifications to identify shared failure modes. For Pipeline A, a recurring error pattern involved players exhibiting high stamina measures but low recent involvement, suggesting further refinement of temporal context features to disambiguate inactivity from conservation for tactical reasons. These audits guided follow-up feature ablations and threshold adjustments at the pipeline level.

A.6.6 Interpretability and Explainability

SHAP analyses revealed the principal contributors to model behaviour in each pipeline. In Pipeline A, match time and score margin consistently ranked among the most influential factors, aligning with tactical intuition about substitution timing and game state. In Pipeline C, perfect accuracy aligned with an apparent separation of offensive and defensive contexts in the learned representation, which was visible in SHAP summaries and supported the face validity of the decision boundary for the available data. These explanations provide coach-facing transparency and help establish trust in the recommendations.

A.6.7 Limitations Observed

The evaluation surfaced constraints that inform subsequent iterations. Class imbalance for certain outcomes, such as "no substitution," required class weighting to stabilise training and avoid bias. Data sparsity for specific tactical niches, such as defensive substitutions following red cards, reduced generalisability in those contexts. The perfect scores in Pipeline C, while encouraging, may reflect overfitting to available proxies rather than robust intent learning; generalisation requires further testing with verified labels and broader coverage.

A.6.8 Outcomes and Insights

The evaluation confirms that the system meets the target requirements and that the multi-pipeline strategy addresses complementary tactical questions in a cohesive way. Explainability via SHAP provides actionable insight for human-in-the-loop decision-making and strengthens practical utility. The results support the feasibility of deployment in real-time analytics settings

and motivate future work to expand datasets, incorporate live video–derived context, and extend label coverage to further improve robustness and tactical fidelity.

A.7 Secondary Data Analysis Compliance Form

School of Engineering and Informatics

SECONDARY DATA ANALYSIS COMPLIANCE FORM FOR UG AND PGT PROJECTS

This form must be completed before any analysis of secondary data may begin.

Introduction

This form should be read in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects (V4.0)".

This form is for project that will use *secondary data*, that is pre-existing data. For projects that will collect new data, either the *User-testing Compliance Form* must be completed, or a full Low-risk or High-risk ethics application made to the Science and Technology Research Ethics Committee.

There are four types of secondary data:

- 1A: Meta-analysis – data obtained directly from published peer-reviewed articles
- 1B: Publicly available anonymised data sets uploaded to open access repositories or databases
- 1C: Secondary datasets available from previous School or Sussex University (ethics approved) projects
- 1D: Secondary datasets not publicly available but which may be accessed under certain conditions/agreements

Figure 1 is a decision flow chart of when this form can be used. The four types of secondary data listed at the top of the flow chart: boxes 1A-1D.

Both **meta-analyses** (1A) and **publicly available anonymized datasets** (1B) do not require a University ethics review. Instead, you should complete the Secondary Data Analysis Compliance Form (Box 3A).

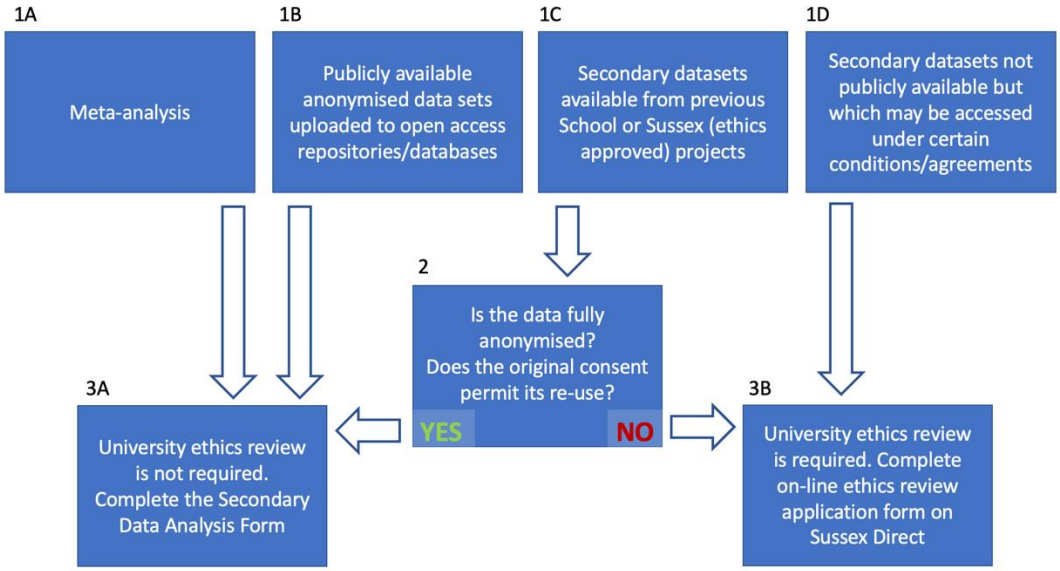


Figure 1. Secondary Data Analysis Flow Chart

For secondary **datasets available from previous School or Sussex (ethics approved) projects** (1C), you need to determine whether the data is anonymized and if the original consent form permits its re-use (Box 2).

- If the answer to both in Box 2 is 'Yes', then a University ethics review is not required, and you should complete the Secondary Data Analysis Compliance Form (Box 3A).
- If the answer to both in Box 2 is 'No', then you must complete an online ethics review application form on Sussex Direct (Box 3B).

For secondary **datasets not publicly available** but which may be accessed under certain conditions/agreements, you will need to make an online ethics review application (Box 3B).

Part 1: Determining whether you can use this Secondary Data Analysis Compliance form or whether you need to submit a full ethics application via Sussex Direct.

You can use this form if your project involves (1) AND/OR (2) below (double click the check box to tick):	
1. A meta-analysis.	☐
2. Using publicly available anonymized datasets (e.g., data obtained from open access repositories or databases). URL of dataset: Add URL here: statsbomb/open-data: Free football data from StatsBomb	☒

You can use this form if your project involves research described in (3), **and** meets **BOTH** criteria A and B below:	
3. Analysing datasets that were originally collected by researchers at the University of Sussex, according to a previously approved ethics application. The data to be analysed also meets both of the following criteria: A. The data is fully anonymised. Yes / No (delete the inapplicable) B. The original ethics application permits the re-use of data collected. Yes / No Sussex Ethics Application Number (e.g., ER/XYZ99/1): Add Number Sussex Ethics Application Project Title: Add Title If the dataset does not meet BOTH criteria A and B, then you will need to apply for an ethics review on Sussex Direct.	☐

You CANNOT use this form if your project involves the following (4):	
4. Analysing datasets that are not publicly available and not originally collected by Sussex researchers. The datasets can only be accessed according to conditions/agreements arranged by your supervisor or the University. In this scenario, you will need to apply for ethics approvals on Sussex Direct, regardless of how the data is anonymised. The ethics review will also consider whether the re-use of the data is permitted according to the original ethics application.	☐

Based on your answers above, please either complete Parts 2-4 of this form or discuss with your supervisor about submitting an ethics application via Sussex Direct.

Part 2: Project information

Project Title:	AI Agent and Recommender for Football Tactics: Live Football Substitution Prediction Using Machine Learning and Real-Time Contextual Features
Student:	Mohamed Reda Mohamed Elsaygh
Supervisor:	Thomas Nowotny
Start & End Dates:	10 May 2025 – 19 August 2025

Part 3: Project description and ethical considerations

This section should be in abstract form describing the background, aims & methods of the project.

This project develops a machine learning-based AI agent for real-time football substitution prediction, aimed at assisting coaches in making tactical decisions during live matches. Using detailed match event data, player statistics, stamina metrics, and contextual features such as score margin and positional roles, the system predicts whether a player should be substituted and, in advanced pipelines, whether the substitution should be offensive, defensive, or neutral.
The methodology integrates extensive feature engineering with supervised classification models including Logistic Regression, Random Forest, XGBoost, Support Vector Machines, Artificial Neural Networks, and Stacking Ensembles. Class imbalance is addressed using SMOTE oversampling, and model performance is evaluated using metrics such as Macro F1, Accuracy, ROC AUC, and precision-recall-based threshold tuning. Explainability is embedded through SHAP value analysis, feature importance rankings, and visualization tools to ensure interpretability for end users. The system is designed to operate in real time, with potential integration into live analytics dashboards.

Ethical Considerations

All data used in the study comes from publicly available football match event datasets or licensed data sources that do not contain personally identifiable information outside the public domain of professional sport. The project focuses exclusively on performance-related metrics, avoiding any profiling based on sensitive attributes such as race, religion, or off-field behavior.
Model predictions are intended as decision-support tools, not replacements for human judgment, and outputs should be interpreted in the context of tactical, physical, and situational factors known to coaching staff. To mitigate bias, multiple algorithms are compared, cross-validated, and tested on diverse match scenarios to avoid overfitting to specific teams or competitions. Results and methods are transparently documented to allow reproducibility and external validation.

Reflection on ethics of data collection (up to 2000 characters)

Consider the procedures performed by the individuals who originally collected the data you'll analyse. We recommend discussing how the researchers ensured participant wellbeing, consent, & confidentiality. Reflect on the four ethical principles for psychologists that are outlined in the *BPS Code of Ethics*: (1) Respect, (2) Competence, (3) Responsibility, and (4) Integrity. You can also refer to specific *BPS guidelines and policies*, including the *Code of Human Research Ethics*.

The dataset used in this project is derived from professional football match event records collected by licensed sports data providers. These datasets contain only publicly observable,

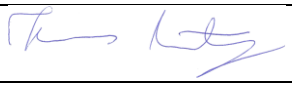
performance-related information such as match events, positional data, and aggregated player statistics. As the participants are professional athletes competing in public, data collection is part of their contractual obligations under league regulations. No personally sensitive, private, or non-public information is included. Nevertheless, ethical considerations are applied to ensure that the use of this data respects the rights and welfare of those involved.

In accordance with the BPS ethical principles:

1. **Respect** – The project uses only performance data relevant to match play, avoiding intrusive or off-field profiling. All data usage complies with licensing agreements and respects the rights of athletes and data providers.
2. **Competence** – Appropriate technical and ethical standards are applied in data processing and modelling, ensuring robust and unbiased results while following best practices in sports analytics.
3. **Responsibility** – Model outputs are intended as supplementary decision-support for coaches, not as sole determinants of player substitutions. Risks of misinterpretation are mitigated through transparency and interpretability tools.
4. **Integrity** – Methods, results, and limitations are reported honestly, with no fabrication or omission. The analysis is fully documented and reproducible, maintaining trust in the research process.

Following the BPS Code of Human Research Ethics, the project minimises potential harm, safeguards privacy, and ensures that all data handling remains within legal, contractual, and ethical boundaries while contributing to the responsible advancement of AI in sports decision-making.

Part 4: Approvals

Role	Name	Signature	Date
Student	Mohamed Reda Mohamed Elsaygh	Mohamed Elsaygh	09/08/2025
Supervisor	Thomas Nowotny		11/08/2025