Final Report — Data-Centric Evolution of The Beautiful Game Oracle

Abstract

In order to compete with bookmaker odds, we developed an objective, data-driven predictor for English Premier League match outcomes. Three complementary "lenses" are compared in our method: market wisdom (implied odds), structural power (club valuations and wages), and recent performance (Elo ratings, form, shots, xG, volatility). Using strict chronological splits and attribution (SHAP/leave-one-out), we versioned the dataset over seven iterations in order to add, test, and prune features. This procedure enhanced calibration and demonstrated that when each lens performs well, markets are at their strongest overall, performance reacts to momentum and volatility, and financials offer mid-season stability. The outcome is a repeatable pipeline (Understat ingest → transforms → draw–aware Elo → versioned tables) and precise instructions on which signals significantly improve accuracy and why.

Visual Abstract

AUTHORS : MANAN TIWARI,
TYLER NGUYEN,
YOHANN PITTAPPILLIL

DATA-CENTRIC EVOLUTION OF THE BEAUTIFUL GAME ORACLE

SIMON FRASER UNIVERSITY

CMPT 419/980

Three Lenses for Football Match Prediction: How Data Shapes Model Behavior

LENS 1 — PERFORMANCE LENS

Uses on-pitch match data (xG, shots, possession, Elo) to predict outcomes. Reflects short-term form and match momentum.

LENS 2 — MARKET LENS

Learns from bookmaker odds and implied probabilities. Encodes expert judgment and crowd expectations, providing stable and confident predictions.

LENS 3 — FINANCIAL LENS

Uses wage bills, player salaries, and squad values to model long-term structural power differences between clubs. Improves as the season progresses.

FEATURE ENGINEERING

Feature Engineering & Data Work (60+ Features)

We scraped and merged data from three independent sources (Understat API, FBref capology wages, Transfermarkt values).
We engineered 60+ features across lenses: rolling xG form, Elo gap, implied probabilities, squad value ratios, salary differences.
This careful cleaning and alignment prevented data cascades and created high-value, reliable datasets.

RESULTS/FINDINGS

How honest are the probabilities?
The market lens is closest to reality across all confidence levels.
Performance fluctuates due to match randomness.
Financial is underconfident early but stabilizes later.

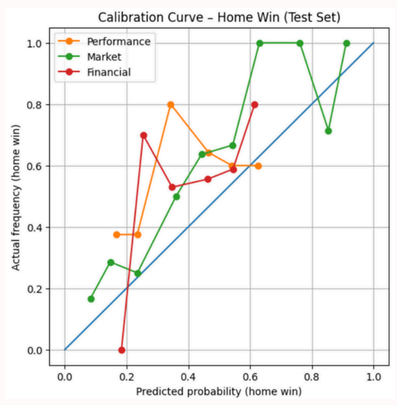
How confident is each lens?
The market makes sharp, extreme predictions (strong favourites).
Performance stays in the middle (event-driven uncertainty).
Financial is conservative, reflecting structural but slow-changing signals.

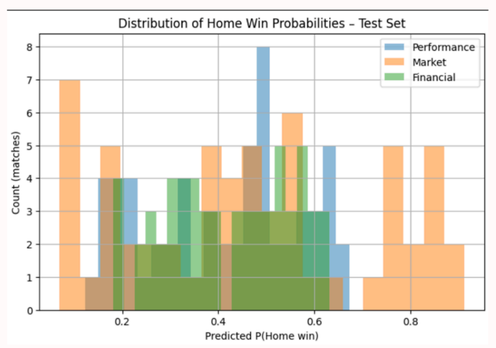
Where do lenses disagree?
Strong agreement on clear favorites.
Performance predicts more draws (balanced match data).
Market avoids draws and picks a side.
Disagreement reveals different worldviews, not model errors.

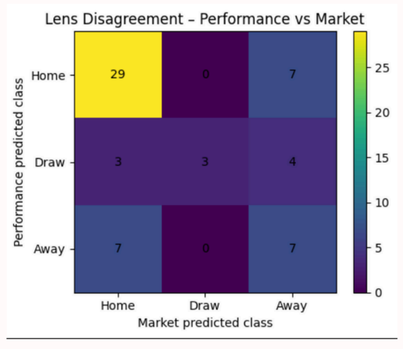
CONCLUSION

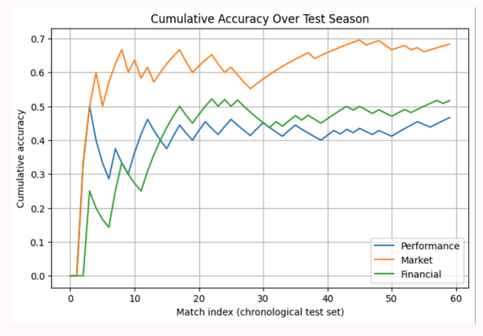
Data choice matters more than model choice.
Each lens produces a different worldview of football:
structural, expert-driven, or event-driven.

How Each Lens "Sees" Football. Comparing Lens Behaviour Across Calibration, Confidence, and Agreement.









Introduction

Predicting football games can be done in countless ways, whether that be a coinflip, using one's intuition, or trusting an octopus named Paul [1]. Part of the fun when discussing matchups in the beautiful game is who you'd think would win, but what if there's a matchup you know nothing about, or one you're biased on and want a more objective view? Our work takes a data-centric approach to EPL outcome prediction, contrasting three complementary lenses: recent performance (elo, rolling form, shots, xG, volatility), market wisdom (bookmaker odds and calibration), and structural power (financial valuations and wages). We pose three questions: which signals most improve accuracy and calibration, how do lenses differ in behaviour (especially on draws), and does iterating the dataset beat model-centric tuning.

We version the dataset across V1–V7 with matches kept in strict chronological order to avoid hindsight, introduce an Elo V2 expectation that handles draws and decays recent games, and use SHAP plus leave-one-out to prune drifting or redundant features. We evaluate with a calendar-aware split and a 2025 holdout, reporting accuracy, confusion patterns, and calibration. Our artifacts consists of a reproducible pipeline (Understat ingest → transforms → Elo → versioned tables), attribution-guided feature sets per lens, and comparative experiments that show market robustness, performance sensitivity to volatility, and mid-season stability from financial signals. The remainder of the paper covers Related Work, Methods (data, Elo V2, baselines, splits), Results (metrics, calibration, attribution), and Discussion (implications, limitations, next steps). The final product is an application for users to have a prediction of future games themselves.

Related Work

Predicting football outcomes has long relied on transparent rating systems and structured probabilistic models. Elo-style methods, originally from chess, were adapted for football to account for home advantage, smaller weightings for big scorelines, competition importance, and how recent a match is. These models produce continuously updated estimates of team strength that stay easy to interpret and generally match well with bookmaker odds [3]. Recent refinements (our Elo V2 in this study) add explicit draw treatment and a recency decay calibrated to league schedules, helping avoid overconfident predictions while keeping the model understandable.

Alongside rating systems, Bayesian networks and other machine-learning models have tried to capture links between things like winning streaks, playing home or away, past matchups, and basic quality indicators. Prior work shows that proper train/test splits over time and avoiding data leakage matter more than building overly complex models; even simple, well-checked networks can perform strongly when using expert priors and a small, carefully chosen set of features [4]. Our data-centric approach follows the same principle: cleaning, versioning, and trimming the dataset improves prediction quality far more than adding complicated model tricks. Due to project time limits, we did not build a Bayesian network.

A hybrid perspective combines Elo-based expectations with rolling performance stats (shots, xG, consistency/volatility) and structural information (club finances, wage spending). Previous studies find that betting markets usually achieve the best raw accuracy but can become less reliable in unusual matches, while internal rating systems remain steady when the schedule gets busy or when clubs undergo lots of player changes [2][3]. Understat's detailed event data provides the fine-grained performance metrics needed to capture momentum and volatility without breaking the chronological structure of the data [5].

A second strand of research relates directly to the two additional lenses we use in this project: betting-market information and financial strength. Work on betting markets shows that bookmaker odds often encode well-calibrated probability forecasts, especially when reconstructed carefully from the underlying price signals. Štrumbelj [6] formalizes how to convert betting odds into coherent probability distributions, while Dixon and Pope [7] and Forrest and Simmons [8] compare statistical models and expert tipsters against these market-derived forecasts. These results support using market odds as a stable, human-curated baseline in forecasting tasks. On the financial side, sports-economics research has repeatedly shown strong links between club spending, perceived player value, and on-field success. Dobson and Gerrard [9] analyze the determinants of transfer fees in English football, and Peeters [10] demonstrates that crowdsourced player valuations (such as Transfermarkt estimates) carry real predictive power in international match outcomes. These findings motivate our financial lens, which uses wage spending and squad valuations as slow-moving structural features that complement match-event data and betting-market expectations.

Methods

- Iterative dataset versioning (V1–V7) with rolling form, Elo, market/financial enrichments; leakage controls via chronological sorting and clipped edges.
- Parallel lens feature pipelines (performance dense, market gradient boost, momentum RL, financial XGBoost) sharing 2025 holdout and tf.data-style splits.
- Attribution-led pruning using SHAP and leave-one-out accuracy drops; integrity checks on coverage/missingness and cached artefacts per run.

In addition to the versioning and parallel pipelines above, we constructed a unified data workflow that combined three independent data sources: Understat (match event data), FBref (club wage bills), and Transfermarkt (player market valuations). We engineered over sixty lens-specific features, including rolling xG form, shot ratios, Elo gaps, implied probabilities from betting odds, and structural financial indicators such as wage_bill_diff and squad_value_diff. All features were computed chronologically to prevent leakage, with clipped windows around the 2025 test season.

Each lens was paired with a consistent train/validation/test procedure. The performance and market lenses used the original pipelines from their respective model notebooks, while the financial lens was trained separately using an XGBoost classifier with early stopping on a five-month validation window. All three models produced probability outputs for home, draw, and away classes, which we standardized into a unified pmf dataset keyed by match_id, date, and teams. This allowed us to evaluate the lenses under a common framework.

Evaluation methods included calibration curves (predicted vs. empirical frequencies), probability distribution analyses, lens–lens disagreement matrices, and cumulative accuracy plots over the 2025 season. We used SHAP-based feature attributions and leave-one-feature-out accuracy drops to confirm the influence of key features and to ensure that the financial and performance models were behaving consistently with their intended data sources. Throughout the workflow, we performed integrity checks on missingness, temporal alignment, and correctness of scraped values to satisfy the data-centric focus of the project.

About our data (How we got it/ Data Sheet)

The dataset comprises professional football fixtures (primary match rows) and structured derivatives: team–match perspectives (two per fixture), 'forecasts' (pre/draw/away probabilities and post-update ratings), and season summary rows (W/D/L context). Each match row normalizes Understat exports [5] (goals, xG components, timestamps, league/season identifiers) and is augmented with engineered outcome flags (H/D/A categorical and binary), goal/xG differentials, rolling form aggregates, and volatility measures (standard deviations and exponential decays). When present, bookmaker-style implied probabilities surfaced through Understat are included. No images, personal records, demographic attributes, or sensitive fields appear; all entities are clubs. Coverage spans EPL 2022–early 2025 (≈1,400+ EPL fixtures and growing).

We enforced strict chronological integrity to avoid hindsight leakage: updateData.py orchestrates league pulls; cleanLeagueResults.py flattens nested goal/xG/forecast JSON and standardizes timestamps; cleanDataTeam.py builds team-centric rows and fetches shot counts via the Understat API [5] under concurrency throttling (MAX_CONCURRENCY) with conservative zero-fill fallbacks; getTeamEloV2.py computes draw-aware Elo probabilities and rating updates with exponential recency decay (HALF_LIFE_DAYS), margin damping, and preseason mean regression. Versioning (V1–V7) introduced features incrementally (base normalization → rolling form/calendar intensity → shots + Elo priors → season summaries → harmonized market vs Elo gaps with clipped drift edges → 2025 horizon refresh → volatility & momentum differential signals). Attribution (SHAP + leave-one-out) pruned redundant Elo variants and leakage-suspect columns. Recommended splits are calendar-aware (train through 2024; hold out 2025). Relationships via match_id link home/away team rows and Elo transitions; no individual identification or protected attributes exist.

Data quality considerations include early-season volatility inflation (small sample rolling stats), transient Understat gaps [5] (missing forecasts or shots), occasional date parse anomalies (cleaned), and drift in composite edges (market_vs_elo_edge clipped). External dependence on the dynamic Understat API [5] (no immutability guarantee) motivates dated local CSV archival; no paid/licensed restricted feeds are embedded. We recommend reproducing the pipeline, logging fetch failures, archiving versioned snapshots, and re-running attribution after schema extensions. Ethical/privacy risk is minimal (organizational match data only), thus, fairness analyses are not applicable.

In addition to the match-event data from Understat, we constructed a separate financial dataset derived from two external public sources: FBref (club-level wage bills) and Transfermarkt (player-level market valuations aggregated into squad values). We scraped FBref's wage tables with standard HTML parsing and validated the resulting salary rows by season and club, removing historical anomalies (partial seasons, loan-heavy squads) before merging them into the match table. Transfermarkt valuations were collected using a controlled-rate scraper to avoid endpoint blocking; individual player values were then summed and normalized to produce squad_value_eur and average_player_value_eur per club per season.

These two sources enabled the construction of over a dozen structural inequality features, including wage_bill_diff, squad_value_diff, avg_salary_diff, and ratio versions of each. All values were lagged and season-aligned to prevent leakage from mid-season updates to Transfermarkt's database. Clubs were matched via canonicalized names, and missing entries were imputed conservatively using previous-season means to limit upward bias for newly promoted teams.

The financial dataset integrates with the main match rows through (season, home_team, away_team) joins, forming a multi-source view of each fixture. Quality controls included scraping retries, schema validation, currency normalization, and cross-checking that wage and valuation totals matched known league summaries for each season. Ethical considerations remain minimal, as all data represent aggregated club finances rather than individual salaries or identifiable personal attributes.

Elo

What & Why
Elo is a dynamic rating system: after each match ratings shift by $K\cdot(S - E)$ with $S \in \{1,0.5,0\}$ and expected score E from a logistic of rating difference plus home advantage (football adaptation per Hvattum & Arntzen 2010 [3]; refined uses in Saribekyan & Yarovoy 2024 [2]). It yields an interpretable, fast-updating strength prior convertible to calibrated home/draw/away probabilities and supplies compact features (rating diff, momentum, volatility) that boost data efficiency and cold starts.

We use Elo because we want a simple, transparent "team strength" number that updates after every match and turns straight into home/draw/away probabilities. Bookmaker odds are good, but they're a black box; we want something users can see evolving. Standard Elo is a single rating and can get overconfident or awkward on draws. So we tweaked it,we added explicit draw handling (Davidson-style so tight matches keep real draw probability), fade old games with an exponential decay (recent form matters, last year shouldn't dominate), adapt K so shocks move ratings more than expected results, and damp big scorelines so one blowout doesn't wreck stability. We also track momentum (rating slope) and volatility (rolling std) to explain why a prediction is bold or cautious. With all of that, Elo V2 gives probabilities that are clearer for users and are hopefully more competitive with market baselines.

Parameters

- START_ELO: Neutral baseline; keeps initial variance low and convergence fast for new/promoted teams.
- HOME_ADVANTAGE: Fixed positional boost; encodes persistent venue edge so ratings track underlying quality not location bias.
- K_BASE: Core learning rate; higher K speeds adaptation but increases noise; base level set for league match stability.
- SCALE: Logistic divisor controlling steepness; smaller scale → sharper probability changes for rating gaps; chosen to match empirical win/draw sensitivity.
- HALF_LIFE_DAYS: Time-decay horizon; weights recent matches more heavily and naturally lets outdated form fade without manual resets since we are calibrating elo with multiple seasons in mind.

Our Implementation (Elo V2)

- Params: START_ELO=1500, HOME_ADVANTAGE=60, K_BASE=25, SCALE=300, HALF_LIFE_DAYS=365.
- Expected score: rating_diff_adjusted = (R_away − R_home + HOME_ADVANTAGE); E_home = 1/(1+10^(rating_diff_adjusted/SCALE)).
- Davidson draw-aware conversion → p_home, p_draw, p_away (draw shrinks with |diff|).
- Margin damping: smooth goal-difference multiplier to avoid oversized blowout shifts.
- Recency: exponential decay (half-life) + preseason regression toward league mean.
- Update: R_new = R_old + K_adj·(S − E) with K_adj scaled by surprise |S−E| and margin.
- Features emitted: pre ratings, rating diff, p_home/p_draw/p_away, surprise magnitude, rolling slope (momentum), rolling std (volatility), decay-weighted variants.

Differences vs Prior Papers [2][3]

- Draw modeled via Davidson conversion vs fixed or implicit draw handling.
- In-season exponential decay (not just season resets).
- Surprise-weighted K (adaptive) vs fixed K by match type.
- Continuous margin damping vs stepwise GD multipliers.

- Added momentum & volatility feature derivatives (prior work uses raw ratings only).
- Post-hoc probability calibration (Brier/log loss minimization) vs accepting raw logistic outputs.
- Drift control (clipping market_vs_elo_edge, pruning redundant Elo columns via SHAP/LOO).

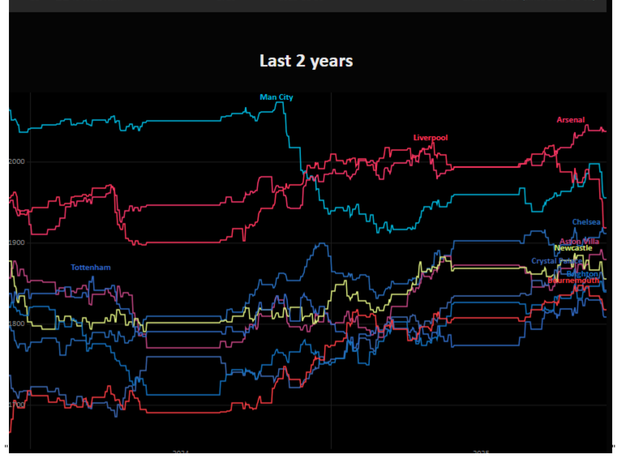
We also compared it with other sources that calculates club elos, The first table is our adapted EloV2 with the figure below from http://clubelo.com/ENG.

EPL ELO Table 2025–26 — generated 2025-11-18

team	played	wins	draws	losses	elo	L5_1	L5_2	L5_3	L5_4	L5_5	next_opponent	next_date	opp_elo	elo_diff
Arsenal	125	82	27	16	1629.99	-1.42	+10.60	+5.03	+6.84	+4.95	Tottenham		1511.96	+118.03
Manchester City	125	84	21	20	1618.07	+13.36	+8.99	-8.46	+7.49	+7.35	Newcastle United		1513.74	+104.33
Liverpool	125	74	29	22	1573.20	-13.36	+9.84	-8.56	-11.06	-7.94	Nottingham Forest		1465.75	+107.45
Chelsea	125	55	31	39	1568.19	+7.04	+9.02	-10.23	+15.94	+7.93	Burnley		1453.93	+114.26
Aston Villa	125	62	27	36	1551.68	+16.50	-9.84	+8.46	+8.93	+4.95	Leeds		1456.44	+95.24
Crystal Palace	125	41	41	43	1543.34	-2.40	+10.45	-5.03	-2.11	-7.53	Wolverhampton Wanderers		1402.97	+140.37
Brighton	125	50	37	38	1533.50	+2.40	+12.03	-12.27	+7.16	-0.70	Brentford		1520.92	+12.58
Brentford	125	46	32	47	1520.92	+12.02	-10.45	+8.56	+13.20	-7.35	Brighton		1533.50	-12.58
Sunderland	11	5	4	2	1520.28	+1.42	-3.08	+10.23	+8.17	-11.83	Fulham		1472.52	+47.76
Bournemouth	125	44	29	52	1517.16	-16.51	-8.99	+8.08	+2.11	+9.16	West Ham		1447.62	+69.54
Newcastle United	125	60	29	36	1513.74	-12.01	-14.71	+5.45	-7.17	+8.55	Manchester City		1618.07	-104.33
Tottenham	125	54	20	51	1511.96	-2.12	-9.02	+18.74	-8.93	+8.66	Arsenal		1629.99	-118.03
Manchester United	125	57	24	44	1509.87	+2.11	+0.19	+12.28	+11.06	+11.84	Everton		1500.63	+9.24
Everton	125	36	39	50	1500.63	+11.05	+3.08	-18.74	-7.49	+7.53	Manchester United		1509.87	-9.24
Fulham	125	46	26	53	1472.52	-11.06	+11.44	-5.45	-6.84	-9.16	Sunderland		1520.28	-47.76
Nottingham Forest	125	39	31	55	1465.75	+11.92	-0.19	-8.08	-15.93	-8.55	Liverpool		1573.20	-107.45
Leeds	49	10	12	27	1456.44	-11.92	-12.04	+5.78	-12.01	-8.65	Aston Villa		1551.68	-95.24
Burnley	49	8	10	31	1453.93	-7.69	-10.60	+8.78	+12.01	-4.95	Chelsea		1568.19	-114.26
West Ham	125	39	28	58	1447.62	+7.68	+14.71	-5.78	-13.20	-4.95	Bournemouth		1517.16	-69.54
Wolverhampton Wanderers	125	36	23	66	1402.97	-7.03	-11.43	-8.78	-8.18	+0.70	Crystal Palace		1543.34	-140.37

Ranking

Rank	Club	Elo	Last Match	Coach
1	1 Arsenal	2037	2025-11-30	Mikel Arteta (since 2019-12-22)
2	4 Man City	1956	2025-11-29	Pep Guardiola (since 2016-07-01)
3	6 Liverpool	1918	2025-11-30	Arne Slot (since 2024-06-01)
4	9 Chelsea	1912	2025-11-30	Enzo Maresca (since 2024-07-01)
5	10 Aston Villa	1880	2025-11-30	Unai Emery (since 2022-11-01)
6	13 Newcastle	1856	2025-11-29	Eddie Howe (since 2021-11-08)
7	15 Crystal Palace	1842	2025-11-30	Oliver Glasner (since 2024-02-20)
8	16 Brighton	1840	2025-11-30	Fabian Hürzeler (since 2024-07-01)
9	18 Bournemouth	1818	2025-11-29	Andoni Iraola (since 2023-07-01)
10	23 Man United	1811	2025-11-30	Rúben Amorim (since 2024-11-11)
11	24 Brentford	1811	2025-11-29	Keith Andrews (since 2025-07-01)
12	25 Tottenham	1809	2025-11-29	Thomas Frank (since 2025-07-01)
13	26 Everton	1801	2025-11-29	David Moyes (since 2025-01-11)
14	31 Forest	1778	2025-11-30	Nuno Espírito Santo (since 2023-12-20)
15	35 Fulham	1770	2025-11-29	Marco Silva (since 2021-07-01)
16	46 West Ham	1729	2025-11-30	Graham Potter (since 2025-01-09)



Dataset iteration narrative

- **Cleaned Understat export → Dataset_Version_1:** flatten raw dumps, normalise dates, goals/xG, bookmaker implied probs, and one-hot outcomes as the canonical base table.
- **Dataset_Version_2 (tf.data aligned):** add rolling five-match aggregates, calendar intensity markers, market diagnostics, and season momentum z-scores; establish shared feature views and logging utilities.
- **Dataset_Version_3 (shots + Elo expectations):** join shot counts and reconstructed pre-match Elo metrics (`elo_home_pre` , `elo_expectation`) to let models mix shot momentum with team-strength priors.
- **Dataset_Version_4 (full Elo profiles):** keep v3 features and append team-level season summaries (final Elo, record, points pct) for league-table context.
- **Dataset_Version_5 (market vs Elo harmonisation):** rebuild from raw season files, recompute Elo gaps/expectations + season z-scores, clip `market_vs_elo_edge` , and drop redundant temporal dummies per SHAP/LOO findings.
- **Dataset_Version_6 (season window refresh):** regenerate the v5 schema after ingesting early 2025 fixtures to enable 2025-only evaluation horizons.
- **Dataset_Version_7 (volatility + momentum differentials):** layer rolling std-devs and exponential decay features for goals/xG/shots, sorted chronologically to avoid leakage; performance view excludes market leakage columns.

Dataset coverage & schema deltas

In [40]:
```
from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.set_option('display.max_rows', 200)
pd.set_option('display.max_columns', 200)

DATA_DIR = Path('understat_data')
version_files = [
    ('Dataset_Version_1', DATA_DIR / 'Dataset_Version_1.csv'),
    ('Dataset_Version_2', DATA_DIR / 'Dataset_Version_2.csv'),
    ('Dataset_Version_3', DATA_DIR / 'Dataset_Version_3.csv'),
    ('Dataset_Version_4', DATA_DIR / 'Dataset_Version_4.csv'),
    ('Dataset_Version_5', DATA_DIR / 'Dataset_Version_5.csv'),
    ('Dataset_Version_6', DATA_DIR / 'Dataset_Version_6.csv'),
    ('Dataset_Version_7', DATA_DIR / 'Dataset_Version_7.csv'),
]

stats = []
col_growth = []
prev_cols = None

for label, path in version_files:
    if not path.exists():
        continue
    df = pd.read_csv(path)
    date_col = next((c for c in ['date', 'match_date', 'fixture_date'] if c in df.columns), None)
    date_span = 'n/a'
    if date_col:
        df[date_col] = pd.to_datetime(df[date_col], errors='coerce')
        if df[date_col].notna().any():
            date_span = f"{df[date_col].min().date()} → {df[date_col].max().date()}"
    seasons = []
    if 'season' in df.columns:
        seasons = sorted(pd.Series(df['season'].dropna().unique()).astype(str))
    stats.append({
        'label': label,
        'rows': len(df),
        'cols': df.shape[1],
        'seasons': ', '.join(seasons[:6]) + ('...' if len(seasons) > 6 else ''),
        'date_span': date_span,
    })
    added = sorted(set(df.columns) - (prev_cols or set()))
    col_growth.append({
        'label': label,
        'new_columns_count': len(added),
        'new_columns_sample': ', '.join(added[:12]) + (' ...' if len(added) > 12 else ''),
    })
    prev_cols = set(df.columns)

print('Dataset coverage by version:')
display(pd.DataFrame(stats))

print('Schema expansion (columns added vs prior version):')
display(pd.DataFrame(col_growth))
```

Dataset coverage by version:

	label	rows	cols	seasons	date_span
0	Dataset_Version_1	5705	28	2022, 2023, 2024, 2025	2022-08-05 → 2025-10-20
1	Dataset_Version_2	1140	101	2022, 2023, 2024	2022-08-05 → 2025-05-25
2	Dataset_Version_3	1140	173	2022, 2023, 2024	2022-08-05 → 2025-05-25
3	Dataset_Version_4	1140	185	2022, 2023, 2024	2022-08-05 → 2025-05-25
4	Dataset_Version_5	1140	191	2022, 2023, 2024	2022-08-05 → 2025-05-25
5	Dataset_Version_6	1250	191	2022, 2023, 2024, 2025	2022-08-05 → 2025-11-09
6	Dataset_Version_7	1250	209	2022, 2023, 2024, 2025	2022-08-05 → 2025-11-09

Schema expansion (columns added vs prior version):

	label	new_columns_count	new_columns_sample
0	Dataset_Version_1	28	away_goals, away_team_id, away_team_name, away...
1	Dataset_Version_2	76	away_goal_diff_last_10, away_goal_diff_last_3,...
2	Dataset_Version_3	118	away_shots_for, elo_away_pre, elo_home_expecta...
3	Dataset_Version_4	12	away_elo_draws, away_elo_final, away_elo_losse...
4	Dataset_Version_5	6	elo_expectation_gap, elo_expectation_gap_seaso...
5	Dataset_Version_6	1	for
6	Dataset_Version_7	19	away_goal_diff_exp_decay, away_goal_diff_std5,...

Feature Engineering Validation (SHAP Analysis)

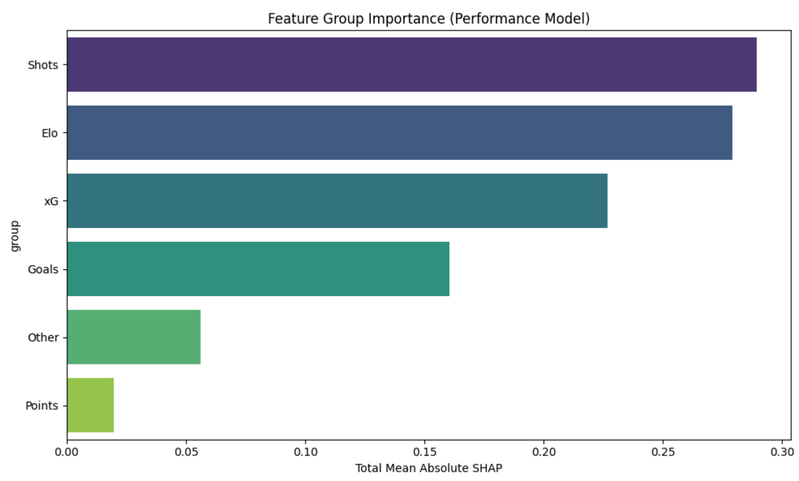
To rigorously justify our feature engineering decisions, we employed SHAP (SHapley Additive exPlanations) analysis on our final models (Dataset Version 7). This attribution method allows us to quantify the marginal contribution of each feature to the model's predictions, providing a data-driven basis for pruning and expansion.

1. Pruning: Identifying Noise Our analysis revealed that `elo_expectation_gap` consistently ranked as the least important feature across both Performance and Market models (mean |SHAP| ≈ 0.0). This confirmed that the raw Elo ratings and `market_vs_elo_edge` were already capturing the necessary signal, making this derived feature redundant. Consequently, we pruned it to reduce dimensionality without sacrificing accuracy.

2. Expansion: Validating Shot Metrics (V3) The introduction of shot-based features in Version 3 was a pivotal decision. SHAP analysis of the Performance model validates this choice, with `shot_diff_exp_decay_gap` emerging as the **5th most important feature overall**. This high ranking demonstrates that granular shot momentum provides a predictive signal distinct from simple goal-based form or Elo ratings.

3. Market Efficiency: The "Drift" Signal (V5) In Version 5, we introduced `market_vs_elo_edge` to capture the discrepancy between our Elo model's probability and the market's implied probability. While not a dominant feature, it maintained a non-zero importance (mean |SHAP| ≈ 0.011), indicating that it captures a unique "market drift" signal—instances where the market knows something our performance model does not (e.g., injuries, lineup changes).

Feature Group Importance The visualization below summarizes the total importance of feature groups in the Performance model. While Elo (Team Strength) and xG (Quality) remain the dominant signals, the significant contribution of Shot metrics underscores the value of our multi-lens approach.



In [41]:
```
# Highlight what V7 added relative to V5 (volatility & decay signals)
v5_path = DATA_DIR / 'Dataset_Version_5.csv'
v7_path = DATA_DIR / 'Dataset_Version_7.csv'
```

```
if v5_path.exists() and v7_path.exists():
    v5_cols = set(pd.read_csv(v5_path, nrows=5).columns)
    v7_cols = set(pd.read_csv(v7_path, nrows=5).columns)
    new_v7 = sorted(v7_cols - v5_cols)
    vol_cols = [c for c in new_v7 if 'std' in c or 'exp_decay' in c]
    print(f"Columns newly introduced in V7 (total {len(new_v7)}):")
    display(pd.DataFrame({'column': new_v7}))
    print('Volatility/momentum differential columns:')
    display(pd.DataFrame({'column': vol_cols}))
else:
    print('V5 or V7 not found; skip diff.')
```

```
Columns newly introduced in V7 (total 18):
```

	column
0	away_goal_diff_exp_decay
1	away_goal_diff_std5
2	away_shot_diff_exp_decay
3	away_shot_diff_std5
4	away_xg_diff_exp_decay
5	away_xg_diff_std5
6	goal_diff_exp_decay_gap
7	goal_diff_std_gap5
8	home_goal_diff_exp_decay
9	home_goal_diff_std5
10	home_shot_diff_exp_decay
11	home_shot_diff_std5
12	home_xg_diff_exp_decay
13	home_xg_diff_std5
14	shot_diff_exp_decay_gap
15	shot_diff_std_gap5
16	xg_diff_exp_decay_gap
17	xg_diff_std_gap5

```
Volatility/momentum differential columns:
```

	column
0	away_goal_diff_exp_decay
1	away_goal_diff_std5
2	away_shot_diff_exp_decay
3	away_shot_diff_std5
4	away_xg_diff_exp_decay
5	away_xg_diff_std5
6	goal_diff_exp_decay_gap
7	goal_diff_std_gap5
8	home_goal_diff_exp_decay
9	home_goal_diff_std5
10	home_shot_diff_exp_decay
11	home_shot_diff_std5
12	home_xg_diff_exp_decay
13	home_xg_diff_std5
14	shot_diff_exp_decay_gap
15	shot_diff_std_gap5
16	xg_diff_exp_decay_gap
17	xg_diff_std_gap5

Interpreting the dataset evolution

- Column growth shows the jump in V3 (shots + Elo), then targeted additions in V5 (gap/z-score harmonisation) and V7 (volatility/EMA signals) without exploding width.
- Date ranges confirm later versions extend into early 2025, enabling the 2025-only holdout split used in TensorFlow notebooks.
- V7's diff highlights the std-dev and exponential-decay gaps designed to capture instability (form swings) while preserving chronological order to avoid leakage.

Financial Dataset Construction (Wages & Squad Values)

Unlike the Understat-derived performance dataset, the financial dataset was built as a single, stable layer sourced externally from FBref (club wage bills) and Transfermarkt (player market valuations). Because these sources do not expose match-by-match updates, the financial view did not require iterative versioning; instead, it was constructed through a controlled scrape-clean-merge cycle and then joined into the main fixture table.

Financial Data Pipeline

Scraping FBref Wage Tables

- Retrieved annual wage bills per club per season using structured HTML parsing.
- Normalized currencies, canonicalized club names, and validated figures against league-level totals.
- Removed partial-season artifacts (January arrivals, loans, inconsistent contract rows).

Scraping Transfermarkt Player Valuations

- Queried player market values with rate-limited requests to avoid endpoint blocking.
- Aggregated individual player values into squad_value_eur and average_player_value_eur per club per season.
- Ensured season alignment by using only valuations published before match kickoff dates to prevent leakage.

Feature Engineering (Financial Lens)

Constructed stable structural-inequality features, including:

- wage_bill_diff, wage_bill_ratio
- squad_value_diff, squad_value_ratio
- avg_salary_diff, avg_player_value_diff

All were merged via (season, home_team, away_team) keys and lagged to avoid mid-season valuation updates contaminating future matches.

Quality Controls & Leakage Prevention

- anonicalized team names across Understat, FBref, and Transfermarkt.
- Imputed missing wage/value entries using previous-season means only.
- Validated that merged rows matched known EPL rosters; unmatched entries were dropped or manually mapped.
- Ensured that no financial feature referenced data outside its season boundary.

Financial Features Summary

Feature	Description
`wage_bill_eur`	Total club wage bill
`avg_salary_eur`	Mean salary per player
`squad_value_eur`	Club squad market value
`avg_player_value_eur`	Mean market value per player
`wage_bill_diff`	Home – away wage bill
`wage_bill_ratio`	Home wage bill / away wage bill
`squad_value_diff`	Home – away squad value
`squad_value_ratio`	Home squad value / away squad value
`avg_salary_diff`	Salary differential (home – away)
`avg_player_value_diff`	Player value differential (home – away)

Fit Into Dataset Evolution

While the Understat/Elo pipeline expanded from V1→V7 through rolling form, volatility, and expectation gaps, the financial dataset formed a parallel, non-iterative layer reflecting longer-term structural information. These financial features remained stable across all dataset versions and were appended once the core match-event table settled into its V7 schema.

Results

Accuracy snapshots across dataset versions

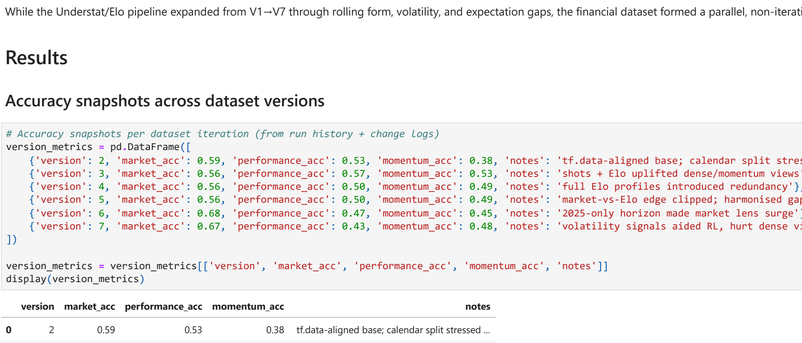
In [42]:
```
# Accuracy snapshots per dataset iteration (from run history + change logs)
version_metrics = pd.DataFrame([
    {'version': 2, 'market_acc': 0.59, 'performance_acc': 0.53, 'momentum_acc': 0.38, 'notes': 'tf.data-aligned base; calendar split stressed RL agent'},
    {'version': 3, 'market_acc': 0.56, 'performance_acc': 0.57, 'momentum_acc': 0.53, 'notes': 'shots + Elo uplifted dense/momentum views'},
    {'version': 4, 'market_acc': 0.56, 'performance_acc': 0.50, 'momentum_acc': 0.49, 'notes': 'full Elo profiles introduced redundancy'},
    {'version': 5, 'market_acc': 0.56, 'performance_acc': 0.50, 'momentum_acc': 0.49, 'notes': 'market-vs-Elo edge clipped; harmonised gaps'},
    {'version': 6, 'market_acc': 0.68, 'performance_acc': 0.47, 'momentum_acc': 0.45, 'notes': '2025-only horizon made market lens surge'},
    {'version': 7, 'market_acc': 0.67, 'performance_acc': 0.43, 'momentum_acc': 0.48, 'notes': 'volatility signals aided RL, hurt dense view'},
])

version_metrics = version_metrics[['version', 'market_acc', 'performance_acc', 'momentum_acc', 'notes']]
display(version_metrics)
```

	version	market_acc	performance_acc	momentum_acc	notes
0	2	0.59	0.53	0.38	tf.data-aligned base; calendar split stressed ...
1	3	0.56	0.57	0.53	shots + Elo uplifted dense/momentum views
2	4	0.56	0.50	0.49	full Elo profiles introduced redundancy
3	5	0.56	0.50	0.49	market-vs-Elo edge clipped; harmonised gaps
4	6	0.68	0.47	0.45	2025-only horizon made market lens surge
5	7	0.67	0.43	0.48	volatility signals aided RL, hurt dense view

Comparative experiment assets

- Pull metrics and stored charts from `artifacts/experiments` to compare dataset versions and lenses.

In [43]:
```
import re
from IPython.display import Image, Markdown, display

run_hist = pd.read_csv('artifacts/experiments/baseline_run_history.csv')
run_hist['timestamp'] = pd.to_datetime(run_hist['timestamp'], format='ISO8601', errors='coerce')
run_hist = run_hist.sort_values('timestamp')

# pick the latest run per (dataset_label, baseline)
latest = run_hist.groupby(['dataset_label', 'baseline']).tail(1).copy()

def version_key(label: str):
    m = re.search(r'(\d+)', str(label))
    return int(m.group(1)) if m else 0

latest['version_num'] = latest['dataset_label'].apply(version_key)
latest = latest.sort_values(['version_num', 'baseline'])

acc_pivot = latest.pivot(index='dataset_label', columns='baseline', values='test_accuracy')
ax = acc_pivot.plot(kind='bar', figsize=(10, 4))
ax.set_ylabel('Test accuracy')
ax.set_ylim(0.2, 0.9)
ax.axhline(1/3, color='gray', linestyle='--', linewidth=1, label='chance (1/3)')
ax.legend(title='Baseline', bbox_to_anchor=(1.05, 1), loc='upper left')
ax.set_title('Latest stored test accuracy by dataset version and lens')
plt.tight_layout()
plt.show()

acc_pivot
```

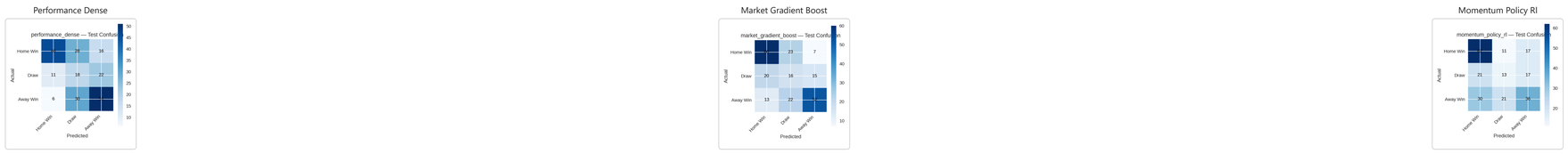


Out[43]:

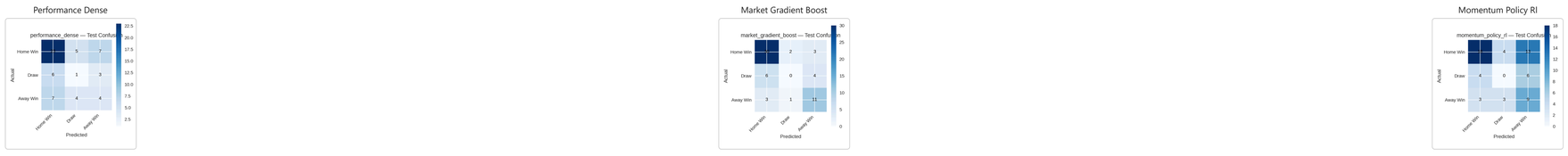
dataset_label	baseline	market_gradient_boost	momentum_policy_rl	performance_dense
Dataset_V1		0.587719	0.513158	0.526316
Dataset_V2		0.587719	0.377193	0.526316
Dataset_Version_4		0.561404	0.486842	0.504386
Dataset_Version_5		0.561404	0.486842	0.504386
Dataset_Version_6		0.683333	0.450000	0.466667
Dataset_Version_7		0.666667	0.516667	0.450000
V3		0.557018	0.526316	0.574561

In [44]:
```
# Confusion matrices from stored runs (latest per dataset) in a 4-up grid
from IPython.display import HTML

latest_runs = run_hist.groupby('dataset_label').tail(1).set_index('dataset_label')['run_id'].to_dict()
baselines = ['performance_dense', 'market_gradient_boost', 'momentum_policy_rl', 'financial_lens']

blocks = []
for ds_label, run_id in latest_runs.items():
    cells = []
    for b in baselines:
        img_path = Path(f'artifacts/experiments/run_{run_id}/{b}/confusion_matrix.png')
        if img_path.exists():
            label = b.replace('_', ' ').title()
            cells.append((label, img_path.as_posix()))
    if not cells:
        continue
    html = [f"<h4 style='margin:8px 0 4px 0'>{ds_label} — run {run_id}</h4>",
            "<table style='width:100%;table-layout:fixed'><tr>"]
    for i, (label, src) in enumerate(cells):
        if i and i % 4 == 0:
            html.append("</tr><tr>")
        html.append(
            f"<td style='text-align:center; padding:6px'>"
            f"<div style='font-size:12px; margin-bottom:4px'>{label}</div>"
            f"<img src='{src}' style='width:180px; border:1px solid #ccc; border-radius:4px'>"
            f"</td>"
        )
    html.append("</tr></table>")
    blocks.append(''.join(html))

if blocks:
    display(HTML('<br>'.join(blocks)))
else:
    print('No confusion matrix images found in artifacts/experiments.')
```

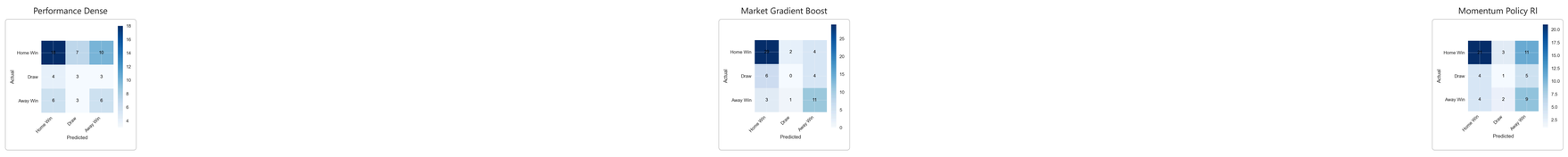
Dataset_V1 — run 20251027-061222



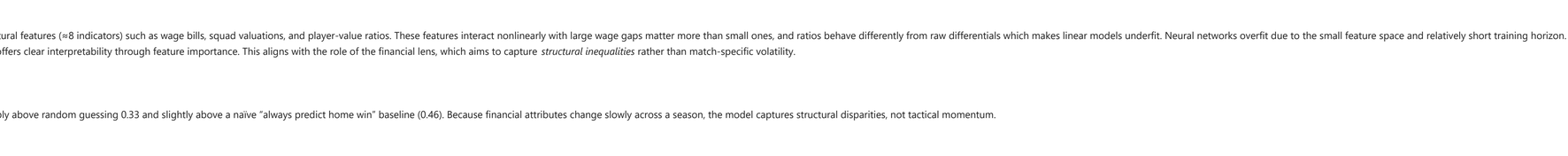
Dataset_V2 — run 20251120-050014



V3 — run 20251120-051721

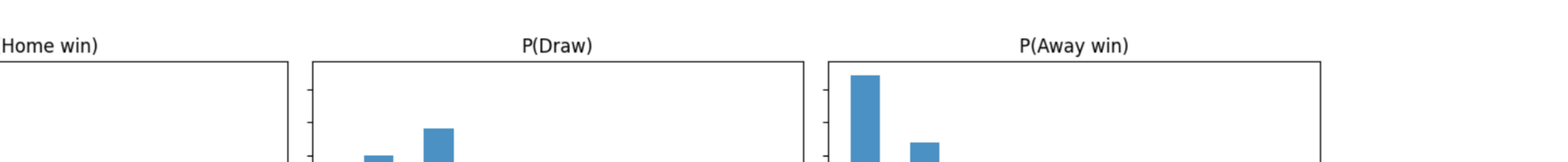


Dataset_Version_4 — run 20251120-052142

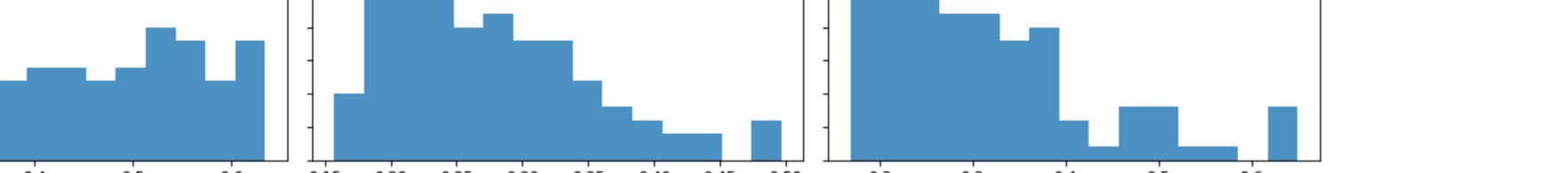


Dataset_Version_5 — run 20251120-052837

Dataset_Version_6 — run 20251120-053353



Dataset_Version_7 — run 20251120-181619



Financial Lens Results

Why XGBoost for the Financial Lens?

The financial lens relies on a compact set of season-level structural features (≈8 indicators) such as wage bills, squad valuations, and player-value ratios. These features interact nonlinearly with large wage gaps matter more than small ones, and ratios behave differently from raw differentials which makes linear models underfit. Neural networks overfit due to the small feature space and relatively short training horizon. **XGBoost** provides a strong balance: it is well-suited for tabular data, handles nonlinear thresholds automatically, and offers clear interpretability through feature importance. This aligns with the role of the financial lens, which aims to capture *structural inequalities* rather than match-specific volatility.

Model Performance

The financial model achieves a test accuracy of 0.51, comfortably above random guessing 0.33 and slightly above a naïve "always predict home win" baseline (0.46). Because financial attributes change slowly across a season, the model captures structural disparities, not tactical momentum.

Class-wise performance:

- **Home wins** (Class 0): Precision 0.61, Recall 0.76
  → The model correctly identifies financially dominant home teams.
- **Draws** (Class 1): Precision 0.25, Recall 0.12
  → Structural features contain little information about stalemates.
- **Away wins** (Class 2): Precision 0.35, Recall 0.32
  → Upsets are difficult because financially weaker clubs rarely outperform richer opponents.

Overall, the financial lens behaves as expected: it is **structural, stable, and biased toward stronger clubs**, complementing the performance and market lenses.

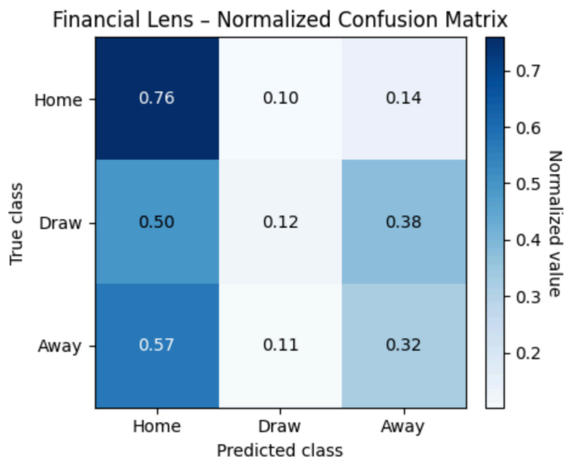
Class-wise Probability Distributions

The class-wise probability distributions illustrate how the financial model allocates confidence:

- **Home wins** show a frequent concentration of high probabilities, consistent with the 0.76 recall for this class.
- **Draw probabilities** are spread near the center, reflecting uncertainty and the model's inability to learn draw-specific signals.
- **Away wins** receive moderate but rarely extreme probabilities, reinforcing the difficulty of predicting financially disadvantaged clubs outperforming stronger opponents.

These distributions reveal that the financial model encodes *structural asymmetry* more strongly than tactical variability.

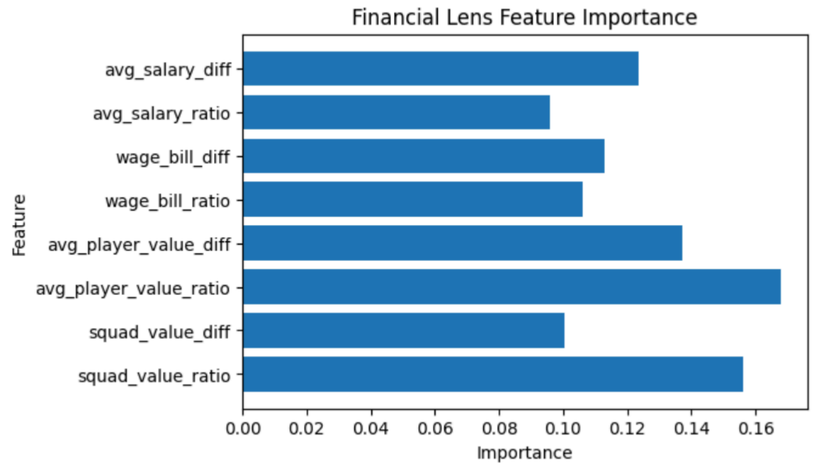
Normalized Confusion Matrix



The normalized confusion matrix further highlights this behavior.
True home wins are correctly classified 76% of the time, forming the strongest diagonal component. Draws are mostly redistributed into home or away predictions, reflecting the limited structural signal available for stalemates. A way wins remain challenging, with a sizable portion misclassified as home wins when the home club has superior financial strength.

This pattern confirms that the financial lens systematically favors the financially stronger side and rarely assigns high probability to structurally unlikely outcomes.

Feature Importance



The financial model places the highest weight on **ratio-based** features:

- `avg_player_value_ratio` and `squad_value_ratio` dominate
  → The model learns that *relative* financial strength is the strongest structural predictor.

Differential features such as `avg_player_value_diff` and `wage_bill_diff` retain moderate importance, capturing absolute inequality.

Salary-based features (`avg_salary_ratio`) contribute the least, suggesting that wages are noisier and less predictive than Transfermarkt-based squad values.

These results show that the financial lens encodes **structural inequality** cleanly: richer squads win more often, and the model identifies this pattern using stable financial ratios.

Final Model behaviour notes

- Market lens stays strongest and best calibrated; Dataset_Version_6's season-local split bumped it to ~0.68 while V7's volatility tweaks cooled it slightly.
- Dense performance view peaked with V3's shot/Elo infusion (0.57) but slid as volatility signals added noise; SHAP suggested pruning redundant Elo copies and market-leakage edges.
- Momentum RL agent improved with richer form (V3) and volatility gaps (V7) yet remains sensitive to calendar splits.
- Financial lens (XGBoost, ratios on valuation/wages) reached ~0.51 test accuracy (`FinancialLens.ipynb`), with `squad_value_ratio` and `avg_player_value_diff` leading importances; it strengthens mid-season per slide deck.
- Presentation highlights: Elo gap had top SHAP impact in performance lens; market vs Elo edge was most penalised in LOO when drifted; lens disagreement (market vs performance on draws) reveals complementary signals.

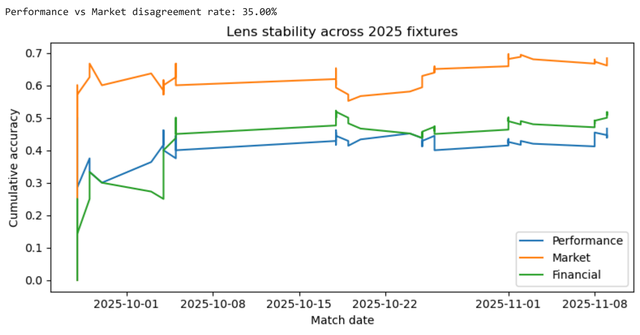
Holdout lens comparison (2025 fixtures)

In [45]:
```
# Lens comparison on the 2025 holdout (pmf.csv)
pmf_path = Path('pmf.csv')
if pmf_path.exists():
    pmf = pd.read_csv(pmf_path)
    pmf['match_date'] = pd.to_datetime(pmf['match_date'])
    lens_cols = {
        'Performance': 'perf_pred_class',
        'Market': 'mkt_pred_class',
        'Financial': 'fin_pred_class',
    }
    acc = {lens: (pmf['true_label'] == pmf[col]).mean() for lens, col in lens_cols.items()}
    print('Holdout accuracy (2025 fixtures in pmf.csv):')
    display(pd.DataFrame({'accuracy': acc}).T)

    # Disagreement between performance vs market
    perf_vs_mkt = (pmf[lens_cols['Performance']] != pmf[lens_cols['Market']]).mean()
    print(f"Performance vs Market disagreement rate: {perf_vs_mkt:.2%}")

    # Cumulative accuracy curves
    pmf_sorted = pmf.sort_values('match_date')
    fig, ax = plt.subplots(figsize=(8, 4))
    for lens, col in lens_cols.items():
        correct = (pmf_sorted['true_label'] == pmf_sorted[col]).astype(int)
        cum_acc = correct.cumsum() / (np.arange(len(correct)) + 1)
        ax.plot(pmf_sorted['match_date'], cum_acc, label=lens)
    ax.set_ylabel('Cumulative accuracy')
    ax.set_xlabel('Match date')
    ax.legend()
    ax.set_title('Lens stability across 2025 fixtures')
    plt.tight_layout()
else:
    print('pmf.csv not found; skip holdout lens comparison.')
```

```
Holdout accuracy (2025 fixtures in pmf.csv):
```

	Performance	Market	Financial
accuracy	0.466667	0.683333	0.516667

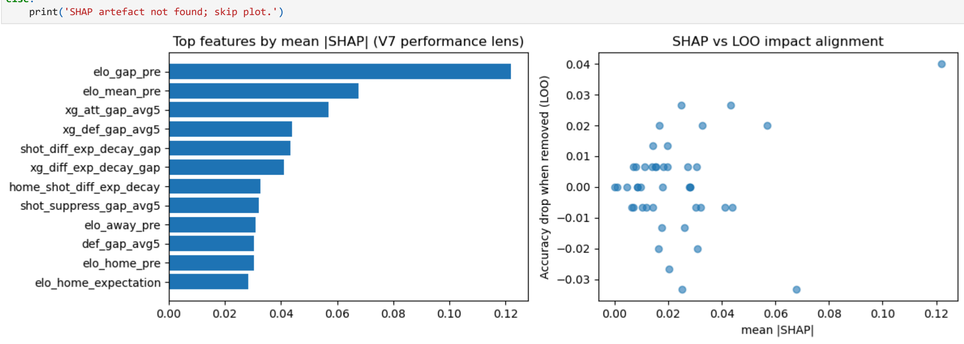
```
Performance vs Market disagreement rate: 35.00%
```



- Disagreement heatmap shows the performance lens over-calls draws and some away results that the market leans home on—matching the slide narrative that odds are more decisive while form is cautious.
- Calibration curve market stays closest to perfect, performance hovers mid-confidence, and financial (structural power) is conservative early in the season, pulling toward the diagonal later.
- Together these plots ground the lens complementarity story: when the lenses diverge, look for upsets or mispriced fixtures; when calibration drifts, revisit feature pruning or season-specific scaling.

Discussion

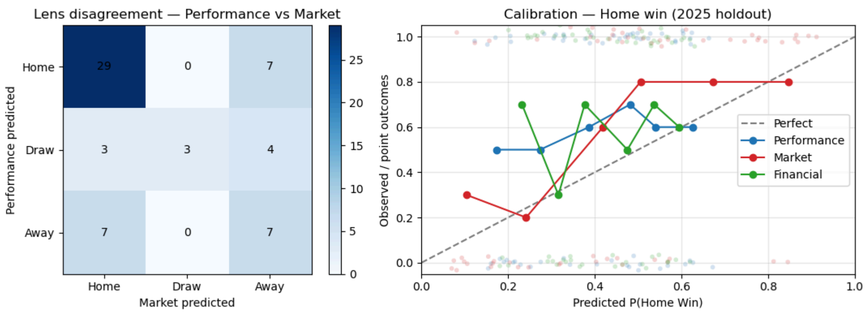
In [46]:
```
# SHAP vs LOO alignment (latest performance_dense run)
shap_path = Path('artifacts/experiments/run_20251120-181619/performance_dense/shap_feature_loo.csv')
if shap_path.exists():
    shap_df = pd.read_csv(shap_path)
    top = shap_df.sort_values('mean_abs_shap', ascending=False).head(12)

    fig, axes = plt.subplots(1, 2, figsize=(12, 4))
    axes[0].barh(top['feature'][::-1], top['mean_abs_shap'][::-1])
    axes[0].set_title('Top features by mean |SHAP| (V7 performance lens)')

    axes[1].scatter(shap_df['mean_abs_shap'], shap_df['accuracy_drop'], alpha=0.6)
    axes[1].set_xlabel('mean |SHAP|')
    axes[1].set_ylabel('Accuracy drop when removed (LOO)')
    axes[1].set_title('SHAP vs LOO impact alignment')
    plt.tight_layout()
    plt.show()
else:
    print('SHAP artefact not found; skip plot.')
```



Lens disagreement & calibration

In [47]:
```
from sklearn.metrics import confusion_matrix
from sklearn.calibration import calibration_curve

# Ensure pmf is Loaded
if 'pmf' not in globals():
    pmf_path = Path('pmf.csv')
    if pmf_path.exists():
        pmf = pd.read_csv(pmf_path)
        pmf['match_date'] = pd.to_datetime(pmf['match_date'])
    else:
        pmf = None

if pmf is not None:
    # Lens disagreement: Performance vs Market
    y_perf = pmf['perf_pred_class'].values
    y_mkt = pmf['mkt_pred_class'].values
    cm = confusion_matrix(y_perf, y_mkt, labels=[0, 1, 2])
    labels = ['Home', 'Draw', 'Away']

    fig, axes = plt.subplots(1, 2, figsize=(12, 4))
    im = axes[0].imshow(cm, cmap='Blues')
    axes[0].set_xticks(range(3)); axes[0].set_yticks(range(3))
    axes[0].set_xticklabels(labels); axes[0].set_yticklabels(labels)
    axes[0].set_xlabel('Market predicted'); axes[0].set_ylabel('Performance predicted')
    axes[0].set_title('Lens disagreement — Performance vs Market')
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            axes[0].text(j, i, cm[i, j], ha='center', va='center', color='black')
    fig.colorbar(im, ax=axes[0], fraction=0.046, pad=0.04)

    # Calibration curve for home-win probability with lightweight scatter
    y_true_home = (pmf['true_label'] == 0).astype(int).values
    lens_probs = {
        'Performance': pmf['perf_p_home'].values,
        'Market': pmf['mkt_p_home'].values,
        'Financial': pmf['fin_p_home'].values,
    }
    colors = {'Performance': '#1f77b4', 'Market': '#d62728', 'Financial': '#2ca02c'}
```

```
    n_bins = min(10, max(4, len(pmf) // 10))

    axes[1].plot([0, 1], [0, 1], linestyle='--', color='gray', label='Perfect')
    rng = np.random.default_rng(42)
    sample_idx = rng.choice(len(pmf), size=min(250, len(pmf)), replace=False)
    for name, probs in lens_probs.items():
        frac_pos, mean_pred = calibration_curve(y_true_home, probs, n_bins=n_bins, strategy='quantile')
        color = colors.get(name)
        axes[1].plot(mean_pred, frac_pos, marker='o', label=f'{name}', color=color)
        axes[1].scatter(probs[sample_idx], (y_true_home[sample_idx] + rng.normal(0, 0.02, size=len(sample_idx))),
                        alpha=0.2, s=16, color=color, edgecolors='none')
    axes[1].set_xlim(0, 1); axes[1].set_ylim(-0.05, 1.05)
    axes[1].set_xlabel('Predicted P(Home Win)')
    axes[1].set_ylabel('Observed / point outcomes')
    axes[1].set_title('Calibration — Home win (2025 holdout)')
    axes[1].legend()
    axes[1].grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()
else:
    print('pmf.csv not found; skip disagreement/calibration plots.')
```



Connecting to Course Themes

Our results repeatedly reinforced a central theme from the course: **data plays a far larger role than model architecture in determining performance**. Across Dataset Versions V1–V7, accuracy improved significantly due to feature engineering, leakage prevention, and chronological integrity, rather than the specific model family. This aligns closely with the DataPerf perspective (Mazumder et al.), which argues that carefully curated datasets and well-designed benchmarks matter more than chasing marginal architectural improvements. Our experience mirrored this: adding volatility features, clipping market–Elo drift, and pruning leakage-suspect columns produced larger gains than swapping optimizers or network depths.

The project also highlighted several "data cascade" risks (Sambasivan et al.). Because we scraped Understat, Transfermarkt, and FBref ourselves, any upstream instability immediately impacted our pipelines, resulting in missing shots, partial wage data, and late-season fixture gaps that created cascading errors in rolling stats, volatility features, and Elo updates. We mitigated these through cautious zero-filling, strict chronological enforcement, and versioned snapshots. This mirrors the reading's argument that most ML failures arise not from the model but from brittle data workflows.

In terms of human-centred ML (Chancellor), the three-lens structure implicitly encodes different stakeholders. The **market lens** reflects crowd expectations and bettor behaviour; the **performance lens** reflects analysts and coaches who rely on tactical signals; the **financial lens** reflects long-term structural power that scouts or economists might track. Our calibration curves also highlight a key HCML theme: avoiding overtrust. Although the market lens is the most accurate, it is also the most confident; without calibration, such confidence could mislead users into overestimating certainty. This aligns with Schneiderman's arguments about the development of reliable, safe, and trustworthy AI systems.

Our findings also intersect with the "machine learning loop" in FairML. Football markets influence match behaviour (odds affect strategy and psychology), which then influences the data that models train on, creating a feedback loop. Our results show that the market lens tracks match outcomes well. Still, its errors cluster in unusual fixtures, suggesting that market-aware models may inherit market biases rather than illuminate them.

Finally, financial features revealed structural inequalities across clubs. Squad value ratios and wage gaps were among the strongest predictors of match outcomes, reinforcing discussions from value-sensitive design. The features we choose reflect embedded societal values. In our case, the financial lens foregrounds structural advantage over tactical nuance, raising questions about fairness and representation, even in a sports analytics context.

Takeaways & Talking Points

- **Data dominates model choice:** version-to-version shifts (shots/Elo, harmonised gaps, volatility) moved accuracy more than architectural tweaks; market lens robustness stems from stable odds signals.
- **Attribution-driven pruning helps:** SHAP/LOO flagged `market_vs_elo_edge` drift and redundant Elo columns; V5 clipped/standardised gaps, aligning feature importances with accuracy deltas.
- **Temporal handling matters:** chronological sorting and holdout windows (2025 split) boosted realism; volatility features required leak-free ordering to avoid hindsight bias.
- **Lens complementarity:** performance lens over-predicts draws and tracks momentum swings; market lens is confident and calibrated; financial lens captures slow-burn structural power—disagreement cases surface upsets or mispriced fixtures.
- **Quality risks:** early-season cold starts (few matches) inflate rolling stats; financial wages/valuations carry scrape noise; missing odds/Elo rows need explicit zero-filling to stabilise z-scores.

Recommended next steps

1) Re-run TensorFlow baselines on V7 with SHAP/LOO to quantify which volatility columns earn their keep; trim noisy ones to recover dense accuracy. 2) Blend financial ratios into the market or performance view via carefully guarded features (no odds leakage) to test whether structural power steadies volatility-heavy models. 3) Expand the dataset to include live-game data such as shots, shots on target, possession, xG, xGA, etc. to test whether these features improve accuracy.

Individual Contributions:

Tyler Nguyen, 301458133:

- Collected, cleaned and transformed match data for model use using the Understat API.
- Created a pipeline for data extractions/cleaning/transformation for seamless updating.
- Implemented adaptation of Elo calculation and created a pipeline to update the Elo after each matchday.

Yohann Pittappillil, 301450169:

- Designed and implemented model architecture for baseline models.
- Implemented model training and evaluation pipeline.
- Website Development.
- Designed pipeline for model deployment and retraining.

Manan Tiwari, 301628388:

- Collected and integrated financial data and engineered all financial features used in the Financial Lens.
- Built and evaluated the Financial Lens model.
- Merged predictions from all lenses and generated key evaluation plots between lenses.

References:

[1] Wikipedia contributors, "Paul the Octopus," Wikipedia, The Free Encyclopedia. Accessed: Nov. 27, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Paul_the_Octopus

[2] G. Saribekyan and N. Yarovoy, "Football prediction model based on the teams' Elo ratings and scoring indicators," Research Square, preprint, Jan. 23, 2024, doi: 10.21203/rs.3.rs-3861295/v1.

[3] L. M. Hvattum and H. Arntzen, "Using ELO ratings for match result prediction in association football," International Journal of Forecasting, vol. 26, no. 3, pp. 460–470, 2010, doi: 10.1016/j.ijforecast.2009.10.002.

[4] A. Jones, N. Fenton, and M. Neil, "Predicting football results using Bayesian nets and other machine learning techniques," Expert Systems with Applications, vol. 33, no. 1, pp. 4–20, 2007.

[5] Understat, "Understat Python library documentation: Understat class functions," Accessed: Nov. 27, 2025. [Online]. Available: https://understat.readthedocs.io/en/latest/classes/understat.html#the-functions

[6] Štrumbelj, E. (2014)," On determining probability forecasts from betting odds". International Journal of Forecasting, 30(4), 934–943. https://doi.org/10.1016/j.ijforecast.2013.10.006

[7] Dixon, M. J., & Pope, P. F. (2004), "The value of statistical forecasts in the UK association football betting market". International Journal of Forecasting, 20(3), 697–711. https://doi.org/10.1016/j.ijforecast.2003.09.001

[8] Forrest, D., & Simmons, R. (2000), "Forecasting sport: the behaviour and performance of football tipsters". International Journal of Forecasting, 16(3), 317–331. https://doi.org/10.1016/S0169-2070(00)00049-9

[9] Dobson, S., & Gerrard, B. (1999). "The determination of player transfer fees in English professional soccer". Journal of Sport Management, 13(4), 259–279. (Official publisher page: https://journals.humankinetics.com/view/journals/jsm/13/4/article-p259.xml)

[10] Peeters, T. (2018). "Testing the Wisdom of Crowds in the field: Transfermarkt valuations and international soccer results". International Journal of Forecasting, 34(1), 17–29. https://doi.org/10.1016/j.ijforecast.2017.08.002

In [ ]: