Sports Betting Prediction Project - Current Code Documentation

# 1. Overview

This project is a football (soccer) betting prediction system. It collects match data from a PostgreSQL database, computes historical features (like Elo ratings, team form, head-to-head advantage, and bookmaker odds), trains an XGBoost model, and exposes predictions via a FastAPI REST API.

# 2. Components

## 2.1 api.py

Defines the FastAPI application and the /predict endpoint. Receives match details (home\_team, away\_team, league, country, date), calls prepare\_single\_match\_features from train\_and\_save.py, and returns prediction probabilities. Includes error handling.

## 2.2 train\_and\_save.py

Contains all training, feature engineering, and prediction logic.

Main responsibilities:

- Load historical matches from PostgreSQL.  
- Parse results into labels (0=Home, 1=Draw, 2=Away).  
- Compute features: Elo ratings, team form, head-to-head, bookmaker implied probabilities.  
- Train an XGBoost model on these features.  
- Save artifacts (model + encoders + features).  
- Provide functions to prepare single match features and generate predictions.

## 2.3 sample\_out.py

A simple client script to test the API. Sends a POST request with match details, prints the raw response, and displays prediction probabilities.

# 3. Data Flow

1. Historical data is stored in a PostgreSQL database (schema: bettingschema.odds).  
2. train\_and\_save.py loads this data and computes features.  
3. Model is trained and saved in ./artifacts.  
4. API loads model artifacts when serving predictions.  
5. Client (sample\_out.py) calls API with new match info.  
6. API computes features for the match, runs prediction, and returns probabilities.

# 4. Features Computed

- Elo ratings (home, away, difference).  
- Team form: wins, points in last N matches.  
- Head-to-head advantage (last meetings).  
- Implied bookmaker probabilities (from odds).  
- Encoded categorical features (country, league).

# 5. Model

Model: XGBoost multiclass classifier (3 outcomes: home win, draw, away win).  
Training uses a time-aware split (last 10% validation).  
Saved artifacts include model, country/league encoders, and feature list.

# 6. API

Endpoint: POST /predict  
Input: JSON with home\_team, away\_team, league, country, date.  
Output: JSON with success flag, probabilities (p\_home, p\_draw, p\_away), and error message if applicable.

# 7. Current Warnings & Issues

- FutureWarning messages from pandas about DataFrame concat and fillna downcasting.  
- If encoders don't recognize new leagues/countries, they return -1.  
- Feature engineering assumes structured historical data is present.