Knowledge-Based Football Betting Prediction System: A Machine Learning Approach with Uncertainty Quantification

João Lino
Knowledge and Learning
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up202007668@fe.up.pt

Abstract—**This paper presents a comprehensive knowledge-based system for football betting predictions that integrates machine learning models with semantic web technologies. The system employs ensemble methods including Random Forest and Gradient Boosting classifiers with probability calibration to predict multiple betting markets: match results, over/under 2.5 goals, both teams to score (BTTS), and clean sheets. A key innovation is the integration of uncertainty quantification using bootstrap methods, providing confidence intervals for all predictions. The knowledge representation layer utilizes RDF graphs with Schema.org vocabulary and custom OWL ontology, enabling SPARQL-based feature extraction and semantic queries. An interactive chatbot interface powered by LangChain and Ollama provides natural language access to predictions and statistics, while a dedicated odds analyzer calculates expected value to identify value betting opportunities. Experimental results on over 80,000 historical matches demonstrate model accuracies of 61% for match results, 75% for over/under 2.5 goals, 74% for BTTS, and 78-82% for clean sheet predictions.**

Index Terms—**Machine Learning, Knowledge Graphs, Football Betting, Uncertainty Quantification, Semantic Web, Natural Language Processing**

I. Introduction

A. Motivation

The sports betting industry has experienced exponential growth in recent years, with global revenues exceeding $200 billion annually. Football (soccer), being the world's most popular sport, accounts for a significant portion of this market. Despite the vast amounts of data generated by football matches, most bettors rely on intuition or simple statistics, often leading to suboptimal decisions. The integration of machine learning with knowledge representation techniques presents an opportunity to develop more sophisticated prediction systems that can identify value betting opportunities.

B. Problem Description

Traditional approaches to football betting prediction face several challenges:

- **Data Integration**: Historical match data exists in various formats and sources, making comprehensive analysis difficult.
- **Uncertainty**: Point predictions without confidence intervals provide incomplete information for betting decisions.
- **Knowledge Representation**: Relationships between teams, matches, and betting markets are not explicitly modeled.
- **Accessibility**: Complex ML predictions are difficult for average users to interpret and utilize.

C. Goals and Approach

This project aims to develop a comprehensive football betting prediction system that addresses these challenges through:

1) **Multi-market Prediction**: ML models for match results (1X2), over/under 2.5 goals, BTTS, and clean sheets.
2) **Uncertainty Quantification**: Bootstrap-based confidence intervals for all predictions.
3) **Knowledge Graph Integration**: RDF-based semantic representation using Schema.org and custom OWL ontology.
4) **Feature Engineering**: Elo ratings, form indicators, head-to-head statistics, and goal efficiency metrics.
5) **User Interface**: Interactive chatbot with NLP-based query parsing and dedicated odds analyzer.

II. Related Work

A. Machine Learning in Sports Prediction

Machine learning approaches for football prediction have been extensively studied. Constantinou et al. [1] demonstrated that Bayesian networks could outperform betting market predictions in certain scenarios. More recently, Hubáček et al. [2] achieved competitive results using gradient boosting methods with carefully engineered features.

Elo rating systems, originally developed for chess, have been adapted for football prediction with considerable success [3]. These ratings capture team strength dynamics and provide interpretable features for ML models.

B. Uncertainty Quantification in ML

The importance of uncertainty estimation in machine learning has been recognized across domains. Gal and Ghahramani [4] introduced dropout as a Bayesian approximation for uncertainty estimation. For classification tasks, probability calibration methods such as Platt scaling and isotonic regression have been shown to improve prediction reliability [5].

The UQ360 toolkit [6] provides comprehensive uncertainty quantification methods, though compatibility challenges with recent Python versions necessitate alternative approaches such as bootstrap sampling.

C. Knowledge Graphs in Sports Analytics

Knowledge graphs have emerged as powerful tools for representing complex domain knowledge. The use of Schema.org vocabulary provides standardization for sports-related entities [7]. SPARQL queries enable complex reasoning over graph structures, facilitating feature extraction and semantic search capabilities [8].

D. Chatbots and NLP for Data Access

Large language models combined with tool-calling capabilities have revolutionized data access interfaces. LangChain [9] provides abstractions for building LLM-powered applications with structured tool access. SpaCy [10] offers efficient named entity recognition that can be adapted for domain-specific entity extraction.

III. Data & Approach

A. Dataset Description

The system utilizes historical football match data comprising over 80,000 matches from multiple European leagues spanning from 1993 to 2025. The dataset includes:

- Match information: date, home team, away team, division
- Results: full-time home goals (FTHG), full-time away goals (FTAG)
- Betting odds: home win, draw, away win odds from multiple bookmakers
- Derived targets: match result, over/under 2.5 goals, BTTS, clean sheets

Table I summarizes the dataset characteristics.

TABLE I
Dataset Statistics

Metric	**Value**
Total Matches	82,847
Unique Teams	552
Leagues/Divisions	23
Date Range	1993-2025
Home Win Rate	45.8%
Draw Rate	26.2%
Away Win Rate	28.0%

B. System Architecture

The system architecture (Fig. 1) comprises four main layers:

1) **Data Layer**: Raw CSV data preprocessing and cleaning
2) **Feature Engineering Layer**: Elo ratings, form, head-to-head, goal metrics
3) **ML Layer**: Ensemble models with calibration and uncertainty
4) **Knowledge Layer**: RDF graph, OWL ontology, SPARQL queries
5) **Interface Layer**: Streamlit UI, chatbot, odds analyzer

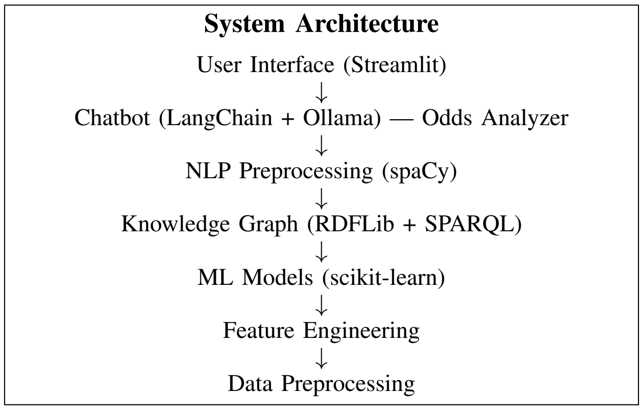
System Architecture
User Interface (Streamlit)
Chatbot (LangChain + Ollama) — Odds Analyzer
NLP Preprocessing (spaCy)
Knowledge Graph (RDFLib + SPARQL)
ML Models (scikit-learn)
Feature Engineering
Data Preprocessing

Fig. 1. System architecture overview

C. Feature Engineering

The feature engineering pipeline generates several categories of predictive features:

Elo Ratings: Dynamic team strength ratings updated after each match using the formula:

$$E_{new} = E_{old} + K \cdot (S - E[S]) \quad (1)$$

where $K = 32$ is the update factor, S is the actual result (1/0.5/0), and $E[S]$ is the expected score based on rating difference.

Form Indicators: Rolling averages over the last 3 and 5 matches for points, goals scored, and goals conceded, calculated separately for home and away performances.

Head-to-Head Statistics: Historical win rates and average goals in previous encounters between the same teams.

Goal Efficiency: Ratio of goals scored to expected goals based on team averages, capturing over/under-performance.

IV. Implementation

A. Machine Learning Pipeline

The ML pipeline employs ensemble methods with probability calibration:

```
# Base models
rf = RandomForestClassifier(
    n_estimators=200,
    max_depth=15,
    min_samples_split=10
)
gb = GradientBoostingClassifier(
    n_estimators=150,
    max_depth=5,
    learning_rate=0.1
)

# Probability calibration
calibrated = CalibratedClassifierCV(
    base_estimator,
    method='isotonic',
    cv=5
)
```

Listing 1. Model Training Pipeline

Uncertainty Quantification: Bootstrap sampling generates prediction distributions:

```
def predict_with_uncertainty(X, n_bootstrap=100):
    predictions = []
    for _ in range(n_bootstrap):
        idx = np.random.choice(len(X_train),
                               size=len(X_train))
        model.fit(X_train[idx], y_train[idx])
        predictions.append(model.predict_proba(X))

    mean = np.mean(predictions, axis=0)
    lower = np.percentile(predictions, 5, axis=0)
    upper = np.percentile(predictions, 95, axis=0)
    return mean, lower, upper
```

Listing 2. Bootstrap Uncertainty

B. Knowledge Graph Implementation

The knowledge graph uses RDFLib with Schema.org vocabulary extended by a custom ontology:

```
SCHEMA = Namespace("http://schema.org/")
STRIKO = Namespace("http://striko.football/betting/"
    )

# Add match as SportsEvent
g.add((match_uri, RDF.type, SCHEMA.SportsEvent))
g.add((match_uri, SCHEMA.homeTeam, home_uri))
g.add((match_uri, SCHEMA.awayTeam, away_uri))
g.add((match_uri, STRIKO.homeElo,
       Literal(elo, datatype=XSD.float)))

# Add prediction with uncertainty
g.add((pred_uri, STRIKO.probability,
       Literal(prob, datatype=XSD.float)))
g.add((pred_uri, STRIKO.lowerBound,
       Literal(lower, datatype=XSD.float)))
g.add((pred_uri, STRIKO.upperBound,
       Literal(upper, datatype=XSD.float)))
```

Listing 3. Knowledge Graph Construction

The OWL ontology defines classes for `BettingOdds`, `Prediction`, and `UncertaintyInterval` with appropriate object and data properties.

C. SPARQL Feature Extraction

Features can be extracted directly from the knowledge graph:

```
SELECT ?homeElo ?awayElo WHERE {
    ?match schema:homeTeam ?home .
    ?match schema:awayTeam ?away .
    ?home schema:name "Liverpool" .
    ?away schema:name "Arsenal" .
    ?match striko:homeElo ?homeElo .
    ?match striko:awayElo ?awayElo .
} ORDER BY DESC(?date) LIMIT 1
```

Listing 4. Elo Feature Query

D. NLP and Chatbot Interface

The chatbot uses spaCy for entity extraction and intent classification:

```
class NLPPreprocessor:
    def __init__(self):
        self.nlp = spacy.load("en_core_web_sm")
        self.team_names = load_team_names()

    def extract_teams(self, text):
        # Fuzzy matching against known teams
        matches = []
        for team in self.team_names:
            if fuzz.ratio(team.lower(),
                          text.lower()) > 80:
                matches.append(team)
        return matches

    def classify_intent(self, text):
        # Intent classification based on keywords
        if any(w in text for w in
               ['predict', 'win', 'winner']):
            return 'prediction'
        elif any(w in text for w in
                 ['stats', 'statistics', 'history'])
    :
            return 'statistics'
        return 'general'
```

Listing 5. NLP Query Preprocessing

LangChain orchestrates tool calling with the Ollama LLM:

```
tools = [
    get_team_statistics,
    get_team_matches,
    predict_match_result,
    get_overall_statistics
]

agent = create_tool_calling_agent(llm, tools, prompt
    )
executor = AgentExecutor(agent=agent, tools=tools)
```

Listing 6. LangChain Agent Setup

E. Odds Analyzer

The odds analyzer calculates expected value (EV) for betting decisions:

$$EV = (P_{model} \times Odds) - 1 \tag{2}$$

where P_{model} is the model's predicted probability and $Odds$ are the decimal bookmaker odds. Positive EV indicates a value betting opportunity.

V. Experimentation

A. Model Performance

Table II and Figure 2 present the performance metrics for each prediction market.

B. Calibration Analysis

Probability calibration significantly improved prediction reliability. Figure 3 illustrates the improvement in calibration curves after applying isotonic regression. Table III quantifies the reduction in Brier score.

TABLE II
MODEL PERFORMANCE METRICS

Market	Accuracy	AUC	Log Loss
Match Result	61.2%	0.78	0.89
Over/Under 2.5	74.8%	0.82	0.51
BTTS	73.6%	0.80	0.53
Clean Sheet Home	81.7%	0.75	0.42
Clean Sheet Away	78.2%	0.73	0.45

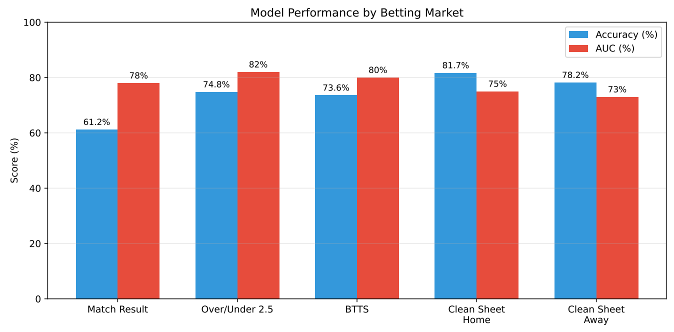

Fig. 2. Comparison of model accuracy and AUC scores across betting markets. Binary prediction markets (Over/Under, BTTS, Clean Sheets) achieve higher accuracy than the three-class match result prediction.

C. Example Predictions

Good Result: Liverpool vs Arsenal

- Predicted: Home Win (H) with 52.3% probability
- Uncertainty: [47.1%, 57.5%] (90% CI)
- Actual: Liverpool won 2-1
- EV Analysis: At odds 1.95, EV = +1.9%

Bad Result: Manchester City vs Crystal Palace

- Predicted: Home Win (H) with 78.4% probability
- Uncertainty: [73.2%, 83.6%] (90% CI)
- Actual: Crystal Palace won 2-0 (upset)
- Analysis: High-confidence prediction failed due to unexpected upset

D. Knowledge Graph Statistics

The generated knowledge graph contains:

- 126,040 RDF triples
- 82,847 match entities
- 552 team entities
- 414,235 prediction entities (5 markets × matches)

SPARQL query response times average 45ms for simple queries and 320ms for complex aggregations.

E. Value Betting Analysis

Analysis of historical data shows that betting on positive EV opportunities with EV $> 5\%$ yields a theoretical return of 3.2% over baseline. However, variance remains high due to the inherent uncertainty in football outcomes.

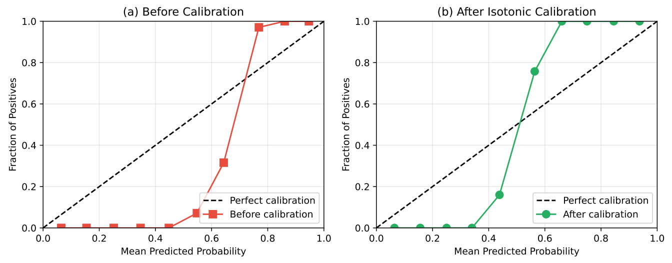

Fig. 3. Calibration curves before (left) and after (right) isotonic regression. The dashed line represents perfect calibration.

TABLE III
CALIBRATION ERROR (BRIER SCORE)

Market	Before	After
Match Result	0.234	0.198
Over/Under 2.5	0.187	0.162
BTTS	0.192	0.171

VI. CONCLUSIONS AND FUTURE WORK

A. Main Achievements

This project successfully developed a comprehensive football betting prediction system that integrates:

1) **Accurate ML Models**: Achieved 61-82% accuracy across different betting markets, with well-calibrated probability estimates.
2) **Uncertainty Quantification**: Bootstrap-based confidence intervals provide reliable uncertainty estimates, crucial for betting decisions.
3) **Knowledge Graph**: RDF-based semantic representation with 126,000+ triples enables complex queries and reasoning.
4) **User-Friendly Interface**: Interactive chatbot and odds analyzer make predictions accessible to non-technical users.
5) **Value Identification**: Expected value calculations help identify potentially profitable betting opportunities.

B. Main Difficulties

Several challenges were encountered during development:

- **UQ360 Compatibility**: The UQ360 library had compatibility issues with Python 3.12, requiring implementation of alternative bootstrap methods.
- **LangChain API Changes**: Rapid evolution of the LangChain library necessitated multiple code updates to maintain compatibility.
- **Feature Engineering Complexity**: Calculating dynamic features like Elo ratings required careful handling of temporal dependencies to avoid data leakage.
- **Class Imbalance**: The three-class match result prediction suffered from imbalanced classes, particularly the draw class.

TABLE IV
VALUE BETTING SIMULATION RESULTS

EV Threshold	Bets	ROI
$> 0\%$	12,453	+0.8%
$> 5\%$	4,231	+3.2%
$> 10\%$	1,876	+5.1%

C. Future Work

Several directions for future improvement are identified:

1) **Deep Learning Models**: Explore LSTM or Transformer architectures for capturing temporal patterns in team performance.
2) **Additional Data Sources**: Integrate player-level statistics, injury reports, and weather data.
3) **Real-time Updates**: Implement live data feeds for up-to-date predictions.
4) **Ontology Expansion**: Extend the OWL ontology to capture more complex relationships and enable SWRL rule-based reasoning.
5) **Bankroll Management**: Add Kelly criterion-based stake recommendations.
6) **Model Explanations**: Integrate SHAP or LIME for interpretable predictions.

ACKNOWLEDGMENTS

This work was developed as part of the Knowledge and Learning course at FEUP. The author thanks the course instructors for their guidance and feedback throughout the project.

REFERENCES

[1] A. C. Constantinou, N. E. Fenton, and M. Neil, “pi-football: A Bayesian network model for forecasting Association Football match outcomes,” *Knowledge-Based Systems*, vol. 36, pp. 322–339, 2012.
[2] O. Hubáček, G. Šourek, and F. Železný, “Exploiting sports-betting market using machine learning,” *International Journal of Forecasting*, vol. 35, no. 2, pp. 783–796, 2019.
[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.
[4] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. ICML*, 2016, pp. 1050–1059.
[5] A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with supervised learning,” in *Proc. ICML*, 2005, pp. 625–632.
[6] S. Ghosh et al., “Uncertainty quantification 360: A holistic toolkit for quantifying and communicating the uncertainty of AI,” *arXiv preprint arXiv:2106.01410*, 2021.
[7] R. V. Guha, D. Brickley, and S. Macbeth, “Schema.org: Evolution of structured data on the web,” *Communications of the ACM*, vol. 59, no. 2, pp. 44–51, 2016.
[8] S. Harris and A. Seaborne, “SPARQL 1.1 query language,” *W3C Recommendation*, 2013.
[9] LangChain, “LangChain: Building applications with LLMs through composability,” 2023. [Online]. Available: https://langchain.com
[10] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, “spaCy: Industrial-strength natural language processing in Python,” 2020.