INTEGRATIVE TASK 2 - APO

Julio Prado
Alejando Mejía
Angel Ordoñez

INTRODUCTION

The goal of this project is the creation of a chatbot that indicates whether a bet is risky or not for a football match of the world cup. This chatbot tries to clear up this by offering consumers real-time knowledge on the level of danger their gamble might be.

Python was used to build the foundation of the chatbot. Two main systems contribute to its intelligence: Bayesian networks assist with handling the intrinsic probabilities and uncertainties in football matches, and the Experta framework uses particular rules and expert logic to evaluate game circumstances. Furthermore, telegram API allows to combine everything into one platform to provide consumers a simple and easy method to quickly find the answers they need.

Problem Statement and Objectives

Users betting on World Cup football matches need an **accessible way** to assess the risk of their wagers. The unpredictable nature of these high-stakes games makes it challenging to accurately gauge a bet's true danger in real-time. This project aims to provide a reliable tool to address this.
Develop a user-friendly interface on Telegram for easy input and clear risk display.
Implement an intelligent core using **Bayesian networks** to model match uncertainties.
Integrate the Experta framework to apply specific rules and expert logic for risk evaluation.
Ensure the tool is accessible, allowing users to quickly obtain insights into their bet's risk.

Requirements Analysis

The user inputs the next information via chat bot:
home_team: colombia (example)
away_team: peru (example)
match_stage: group (example)
home_conceded: 2.0 (example)
away_conceded: 1.0 (example)
home_win_percentage: 0.75 (example)
weather: good (example)
key_players_injured: both (example)
match_importance: 10 (example)
home_crowd_size: 50000 (example)
home_crowd_support: 75 (example)
last_meetings_draws: 0 (example)
match_date: 2025-06-03 (example)

Expert System Output:
The system generates a comprehensive betting recommendation including: final decision type (home_win, away_win, draw, avoid_bet, home_blowout), confidence percentage (0-100%),

detailed reasoning based on activated rules, Bayesian probability analysis, and specific betting advice with risk assessment categorized as very high, high, medium, or low confidence levels.

Knowledge Acquisition and Representation

To build the chatbot's intelligence, a vast history of football matches, spanning from 1980 to today, was collected. From this data, a Bayesian Network was created. This network acts like a map, showing how different factors influence a match's winner, helping to handle all the uncertainties.

This Bayesian Network includes these key points, or "nodes":**home_team, away_team,** neutral (if played on neutral ground), away_strength, home_strength, tournament and winner (the outcome predicted).

To determine the risk of a bet, the Bayesian Network takes the home_team and away_team as its primary evidence. After processing all the input data, it outputs a ϕ (phi) value. This ϕ directly represents the probabilities of the three possible match outcomes: a home win, an away win, or a draw.

System Design

Bot Interface: Telegram bot (bot.py) that handles user interactions using state pattern
User Session Management: Manages conversations with multiple users (UserSessionManager.py, UserSession.py)
State Machine: Series of states that guide users through prediction process (states/ folder)
Team Finder: Helps users find teams by name (Finder.py)
Prediction Engine:

- Bayesian Network: Probabilistic model for match outcome predictions
- Expert System: Rule-based system using Experta framework for betting recommendations. Takes bayesian network output as another fact.

Data Storage: CSV files with historical match data

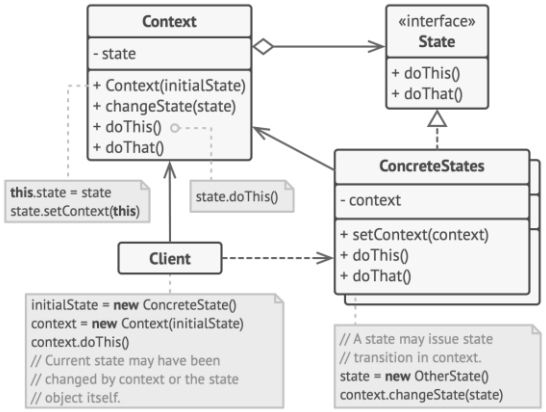
Implementation Details

State

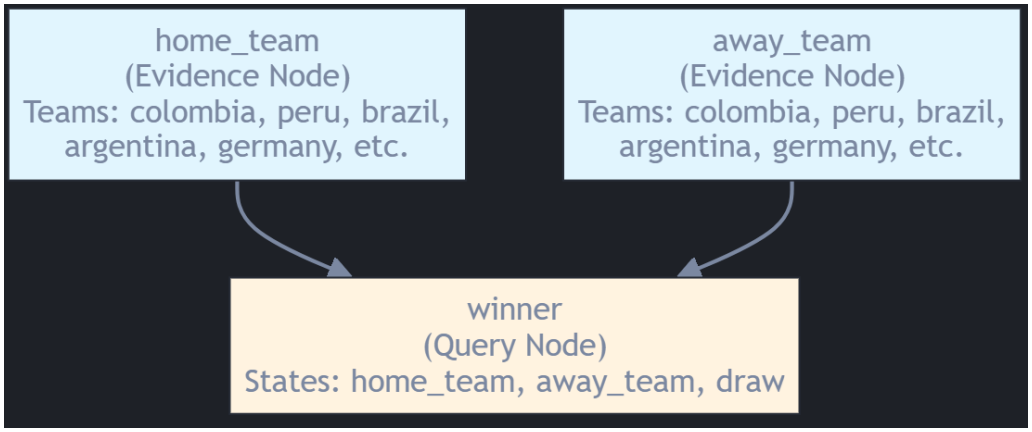
Implementing bot states is crucial for managing user interaction effectively in a chatbot. A chatbot needs to guide users through a structured conversation, collecting specific pieces of information sequentially to make a prediction. States provide the necessary framework to track the conversation's progress, store user input at each stage, and ensure the chatbot responds appropriately as it gathers all required data for a complete risk assessment.

A Base State sets core methods for all other states. **Fifteen specific states** handle various input types for predictions. **Each state manages** user messages, collects specific match data,

stores it in user sessions, and directs the conversation to the next state. **The Final State is reached** when all necessary inputs are gathered, triggering the prediction calculation.



Bayesian Network Details



Operation:

1. **Input:** The system receives as evidence the team names (e.g., colombia vs peru)
2. **Inference:** Uses conditional probabilities estimated from the historical dataset 'final_dataset.csv'
3. **Output:** Calculates probabilities for each possible outcome:
    - prob['home']: Probability of home team victory
    - prob['away']: Probability of away team victory
    - prob['draw']: Probability of draw

The network uses Maximum Likelihood Estimation to learn conditional probability tables from historical match data, and then employs Variable Elimination to make inferences about new matchups.

Testing and Validation

Bayesian Network Test

This test suite validates the soccer match prediction model, a Bayesian Network, by comparing its predictions to historical match data. For efficiency, the network instance and the historical dataset are loaded only once at the beginning of the testing process.

In each test, the model provides prediction probabilities for possible match outcomes. The outcome with the highest predicted probability is then identified.

Finally, this highest probability prediction is asserted to match the most common historical result between the specific teams being tested. This process thoroughly verifies the model's accuracy by ensuring its predictions consistently align with established historical patterns for a variety of team combinations.

Finder test
This test suite verifies the Finder class's ability to help users locate soccer teams. It sets up a Finder instance, then runs tests to confirm exact matching (e.g., "Colombia" works, case-insensitivity) and partial matching (e.g., "co" finds "Colombia"). The suite also ensures get_all_teams() returns a complete list. Overall, these tests confirm the Finder component reliably assists users in finding teams through various search methods.

Expert System Test
This test suite validates the soccer betting expert system through four key areas: historical accuracy testing against landmark matches (requiring 75% accuracy), expert prediction validation ensuring 70% agreement with human experts, usability testing for response times under 2 seconds and 85% interpretability scores, and stress testing with 100 concurrent analyses maintaining 95% success rates. The framework includes edge case testing with extreme values, invalid inputs, and missing data to ensure robustness. Additionally, concurrent processing tests verify thread-safe operation with parallel execution across multiple threads, guaranteeing system stability under high-load conditions.

Deployment
Deploying the chatbot is straightforward. Once a PC is running the bot.py script, the bot becomes active. Users can then find and interact with it on Telegram by searching for its name, 'futifu_bot', or its username, '@fubotero'.

Conclusion and Future Work

This project successfully developed a chatbot designed to provide real-time risk assessments for World Cup football bets, addressing the challenge users face in gauging wager danger. The chatbot leverages a Python foundation, integrating a Bayesian Network for probabilistic match outcome predictions and an Experta framework for rule-based risk evaluation. While effective, a more comprehensive Exploratory Data Analysis (EDA) would have been beneficial, allowing for a data-driven approach to identify key factors and refine the logical rules within the Experta system, potentially improving the precision of the risk assessments.

To further enhance the chatbot's capabilities, future efforts should focus on integrating real-time data feeds, such as live match statistics, to enable dynamic risk updates during games. Additionally, refining the expert system's rules based on continuous performance analysis and potentially incorporating advanced machine learning techniques could boost predictive accuracy. Improving user experience through more intuitive input methods and richer output visualizations would also be a key area for development.

- References

Refactoring.Guru. (n.d.). *State design pattern*. Retrieved from https://refactoring.guru/design-patterns/state