Football Matchs Result Prediction Using AI And N8N Automation

By :

ED-DAHMANI Soulaimane

AKOUDAD Karim

CHARKAOUI Soufiane

DOUMI Mohamed Taha

TP : B

Encadré par :

Prof. RABEH Ayoub

AI & NoSql : Project

Table des matières

1. Introduction

The continuous evolution of digital technologies has profoundly transformed the way data is collected, processed, and exploited across various domains, including sports analytics. Football, as one of the most popular and data-rich sports worldwide, generates massive volumes of information at different levels, such as match results, team statistics, player performances, tactical formations, seasonal trends, and historical confrontations. With the increasing accessibility of online sports data platforms and public datasets, football analytics has become an important research and industrial field, attracting both academic researchers and professional developers. However, the sheer volume and complexity of football data make manual analysis inefficient and unreliable, especially when the objective is to predict match outcomes with a high level of accuracy.

Traditional football match prediction methods are generally based on expert opinions, intuition, or simple statistical indicators such as win percentages, goal differences, or league rankings. While these approaches may provide a general overview of team performance, they fail to capture the complex and dynamic relationships that exist between football entities. Football matches are influenced by a wide range of interdependent factors, including historical rivalries, player injuries, home and away performance, seasonal momentum, and tactical changes. These factors interact in non-linear ways, making prediction a challenging task that goes beyond classical statistical models.

In recent years, Artificial Intelligence (AI) and Machine Learning (ML) techniques have demonstrated strong capabilities in handling complex prediction problems by learning patterns from large datasets. More recently, the emergence of Large Language Models (LLMs), such as the latest version of ChatGPT, has further expanded the potential of AI systems by enabling advanced data understanding, contextual reasoning, and explainability. LLMs can assist not only in prediction tasks but also in data preprocessing, feature interpretation, and decision explanation, which are critical for building trustworthy AI systems.

This engineering project proposes the design and development of an online football match prediction system that leverages advanced AI techniques, including LLM-assisted processing, automated data pipelines, and graph-based data storage. Football data is collected from Kaggle datasets and football match history websites, where it is originally available in CSV format. This data is then cleaned, transformed, and converted into a graph-based representation using the Neo4j database, enabling the modeling of complex relationships between teams, matches, players, and competitions. The entire data pipeline is automated using the n8n workflow automation platform, ensuring continuous data updates and system reliability.

The final system integrates data automation, intelligent prediction models, and a dynamic web interface to provide accurate, real-time football match predictions. By combining AI, LLMs, automation, and graph databases, this project demonstrates a modern, scalable, and intelligent approach to sports analytics, suitable for real-world deployment and future extensions.

2. Problematic

One of the main challenges addressed in this project is the complexity and heterogeneity of football data. Football-related information originates from multiple sources, including public datasets from Kaggle and specialized football history websites. These sources provide data in tabular CSV format, which is convenient for storage but limited in its ability to represent relationships between entities. For example, a single match involves multiple teams, players, competitions, and contextual conditions, all of which are interrelated. Representing such complexity using traditional relational or flat data structures leads to redundancy, poor scalability, and limited analytical capabilities.

Another major problem lies in data quality and consistency. Real-world football data often contains missing values, inconsistencies, duplicate entries, and formatting issues. These problems can significantly affect the performance of AI models if not handled properly. Manual data cleaning is time-consuming and error-prone, especially when data is updated frequently. Therefore, ensuring automated, reliable, and repeatable data preprocessing is a critical challenge for any large-scale prediction system.

From a modeling perspective, predicting football match outcomes is inherently difficult due to the uncertainty and randomness associated with sports events. Traditional machine learning models may struggle to capture long-term historical dependencies and contextual patterns if the data representation is not sufficiently expressive. Moreover, selecting appropriate models, tuning hyperparameters, and ensuring generalization across different leagues and seasons require careful design and continuous evaluation. The integration of LLMs introduces additional challenges, such as aligning textual reasoning with numerical prediction tasks and managing computational complexity.

Finally, system-level challenges must be addressed. Since the application is deployed online, it must support real-time predictions, frequent data updates, and multiple concurrent users. Performance, scalability, and fault tolerance are essential requirements. Additionally, users must trust the predictions generated by the system. Black-box models that provide results without explanation may reduce user confidence. Therefore, improving interpretability and transparency, especially through LLM-generated explanations, is a key challenge in the system design.

3. Objectives

The primary objective of this project is to design and implement an intelligent football match prediction system capable of delivering accurate and reliable predictions based on historical and real-time data. This involves developing AI models that can learn from complex football data and adapt to new information as it becomes available. The system aims to move beyond simplistic prediction methods and provide a data-driven solution that reflects the real dynamics of football competitions.

A second major objective is to build a complete and automated data pipeline that handles data collection, preprocessing, transformation, and storage with minimal human intervention. By using the n8n automation platform, the project seeks to ensure that football data is continuously updated, cleaned, and integrated into the system. Automation reduces operational errors, improves data freshness, and enhances overall system reliability, which is essential for an online prediction platform.

Another important objective is to exploit advanced data modeling techniques by transforming CSV-based football data into a graph-based representation using Neo4j. This objective focuses on capturing relationships such as team confrontations, seasonal participation, match outcomes, and historical trends. Graph modeling enables richer data analysis and supports more powerful feature extraction for AI models.

Additionally, the project aims to integrate Large Language Models, specifically the latest version of ChatGPT, to enhance both prediction performance and interpretability. LLMs are used to assist in data understanding, contextual reasoning, and explanation generation, making the system more transparent and user-friendly. Finally, the project seeks to develop a dynamic and interactive web application that allows users to access predictions, statistics, and insights in real time, ensuring a high-quality user experience.

4. Tools and Technologies Used

This project relies on a combination of modern tools and technologies, each playing a specific role within the system. Artificial Intelligence models form the core of the prediction engine. These models are trained on structured features extracted from historical football data and are responsible for predicting match outcomes. To enhance the intelligence and interpretability of the system, Large Language Models (LLMs), specifically the latest version of ChatGPT, are integrated into the processing pipeline. LLMs assist in feature interpretation, contextual analysis, and the generation of human-readable explanations for predictions.

For data automation and orchestration, the n8n platform is used extensively. n8n is a workflow-based automation tool that enables the design of complex data pipelines through interconnected nodes. In this project, n8n workflows are responsible for collecting football data from Kaggle datasets and football match history websites, preprocessing CSV files, triggering data transformation processes, updating the Neo4j database, and initiating AI model retraining when new data becomes available. The use of n8n significantly improves system efficiency and reliability.

Data storage is handled using Neo4j, a graph-based database that is particularly well suited for modeling complex relationships. Football entities such as teams, players, matches, competitions, and seasons are represented as nodes, while relationships capture interactions such as match participation, historical confrontations, and competition membership. This graph-based approach enables efficient querying and advanced analytics that are difficult to achieve with traditional relational databases.

Finally, a dynamic web application is developed to serve as the user interface. The frontend communicates with backend APIs to display predictions, statistics, and analytical insights in real time. Together, these tools and technologies form a cohesive ecosystem that supports automation, intelligence, scalability, and usability.

5. Architecture and Operation

The proposed system follows a layered and modular architecture designed to ensure scalability, maintainability, and robustness. At the foundation of the system lies the automation and data ingestion layer, implemented using n8n. This layer orchestrates the entire data pipeline through workflows that are triggered either on a scheduled basis or manually. These workflows collect football data from Kaggle and football history websites, handle CSV file preprocessing, and ensure data consistency before further processing.

Once the data is cleaned and normalized, it is passed to the data transformation layer, where tabular CSV data is converted into a graph-based structure. This transformation involves mapping rows to nodes and defining relationships that represent football dynamics, such as matches played between teams, seasonal participation, and historical outcomes. This process preserves the semantic richness of football data and enables advanced relationship-based analysis.

The transformed data is then stored in the Neo4j graph database, which serves as the central data repository. Neo4j enables efficient traversal of historical relationships and supports complex queries required for feature extraction and analysis. On top of this storage layer, the AI and LLM-assisted processing layer operates. Machine learning models extract features from the graph and generate match predictions, while LLMs provide contextual reasoning and explanation generation.

The backend layer acts as an intermediary between the database, AI models, and frontend application. It exposes APIs for prediction requests, data retrieval, and system monitoring. Finally, the frontend layer presents predictions and insights to users through a dynamic and interactive web interface. The entire system is monitored continuously, with automated updates and retraining ensuring that predictions remain accurate and relevant over time.

6. Neo4J Database

6.1 Data Collection from football-data.co.uk

As part of this project, the first step consisted of collecting reliable and structured football data. The website ''football-data.co.uk'' was selected as the main data source because it provides high quality CSV files containing detailed historical football match statistics, particularly for the English Premier League.

The downloaded CSV files cover multiple seasons and include essential information such as match dates, home and away teams, final scores, and several statistical indicators related to match outcomes. These datasets were downloaded locally to enable further processing and integration into a database system.

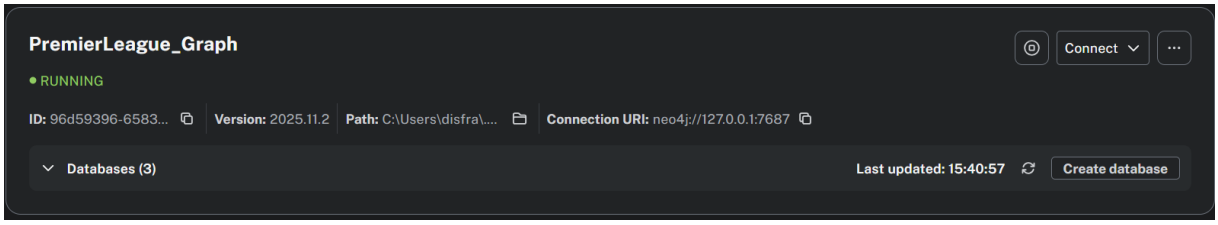
6.2 Local Storage and Modeling Using Neo4j

After collecting the CSV files, the data was imported into a local Neo4j database. Neo4j was chosen due to its-graph based model, which is well suited for representing relationships between football entities such as teams and matches.

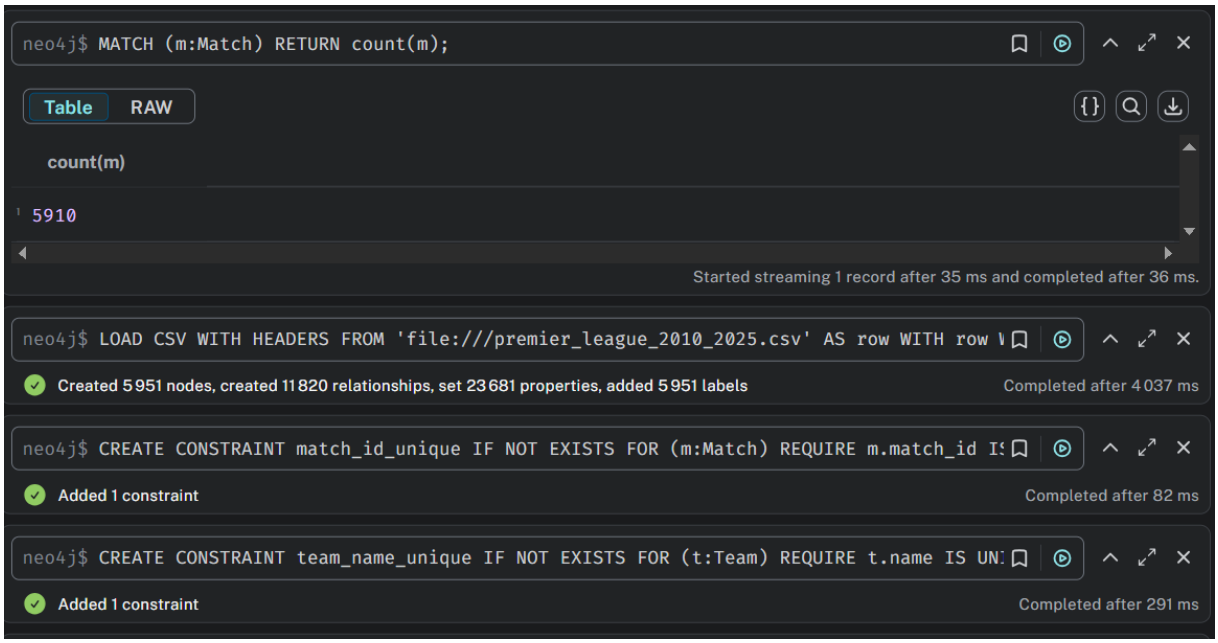
The import process was performed using Cypher Queries, mainly through the ''LOAD CSV WITH HEADERS'' command. The data was modeled using :

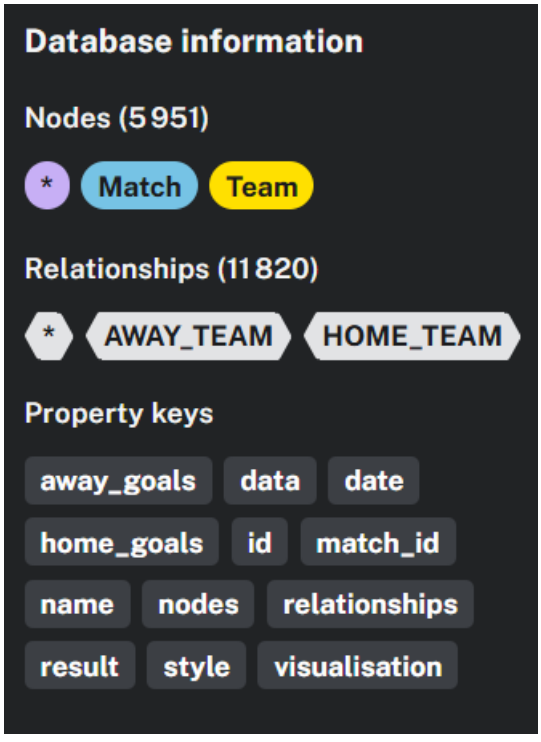
- Match nodes, representing individual football matches.
- Team nodes, representing football teams.
- Relationships connecting teams to matches, such as home and away participation.

Visualization :



AI & NoSql : Project





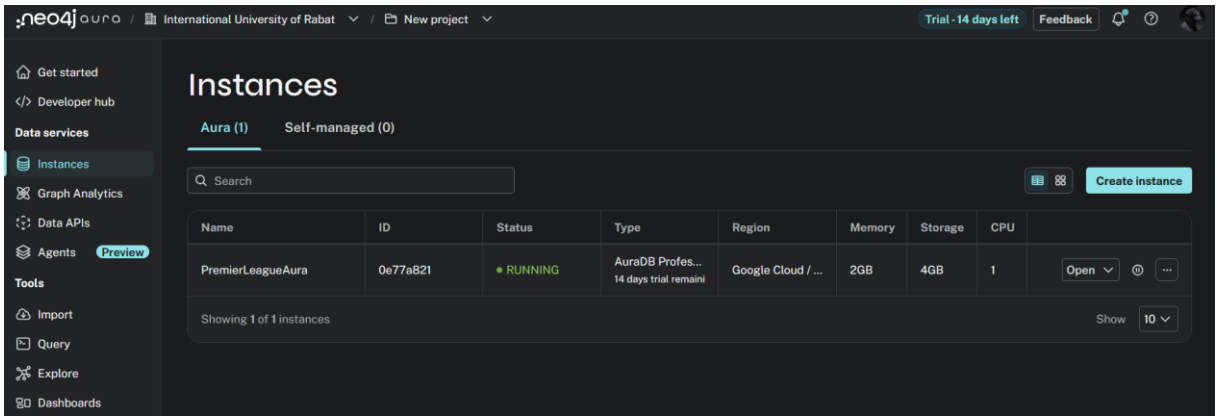
6.3 Migration to Neo4j Aura (Cloud Environment)

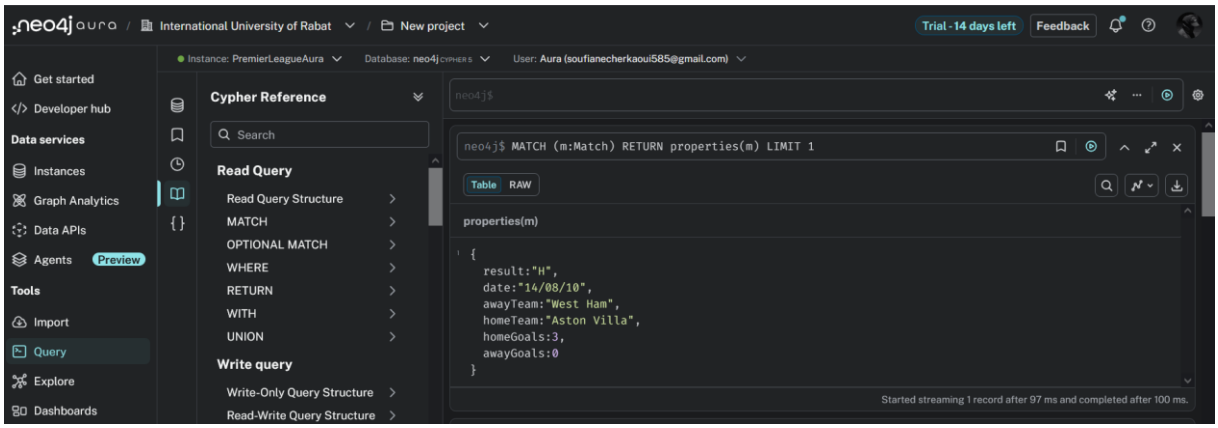
Once the local Neo4j database was successfully populated and validated, the final step consists of migrating the data to Neo4j Aura, a managed cloud platform. The aim of this migration was to ensure remote accessibility, scalability, and easier collaboration.

The migration was performed by re-importing the CSV data, hosted online, directly into Neo4j Aura using Cypher Queries compatible with the cloud environment. This approach made it possible to reproduce the same graph structure defined in the local database while taking advantage of the reliability and security provided by Neo4j Aura.

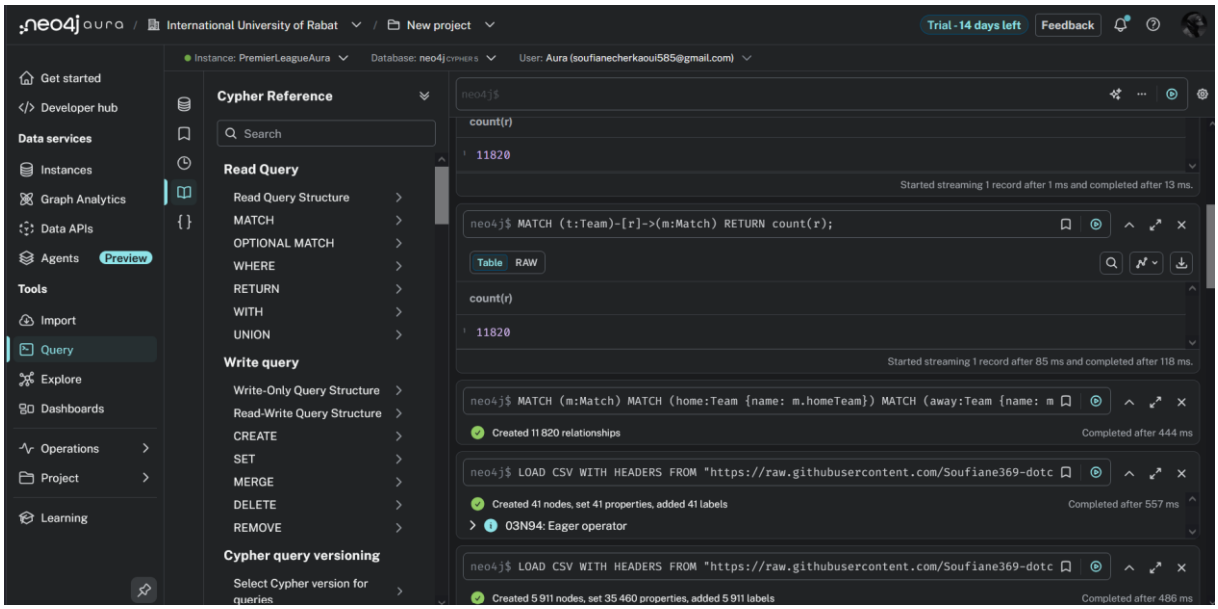
Post-migration checks were carried out to confirm that all nodes and relationships were correctly created and that the data remained fully usable.

Visualization :





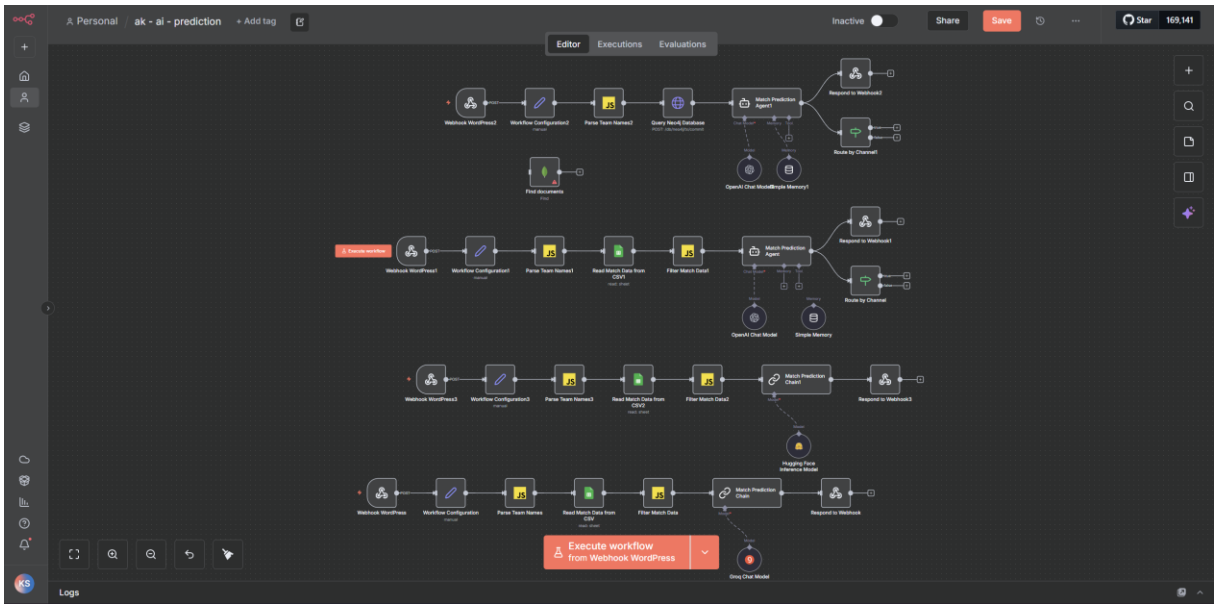
AI & NoSql : Project



This part of the project focused on the acquisition and preparation of football match data for graph-based storage and future analysis. Data was collected in CSV format from the ‘’football-data.co.uk’’ website, providing structured historical information on Premier League matches. The collected data was then stored in a local Neo4j database, where it was organized using a graph model representing teams, matches, and their relationships. This modeling approach allowed the data to be structured in a way that reflects real-world interactions between football entities.Finally, the data was migrated to Neo4j Aura, enabling cloud based access to the same dataset.

7. Workflow Description and Explanation

7.1 General Workflow Overview



The workflow is designed to predict the outcome of a football match requested by a user through a website.

The user sends a match in text form (for example: "Arsenal vs Chelsea").

The workflow processes this request through several independent paths, each using a different data source or reasoning method.

All results are then merged and analyzed by a final decision-making model, which produces one final prediction.

The workflow follows these main steps:

1. Receive the user request from the website.
2. Extract the two team names from the user input.
3. Run multiple analysis paths in parallel.
4. Merge all results.
5. Generate a final prediction and return it to the website.

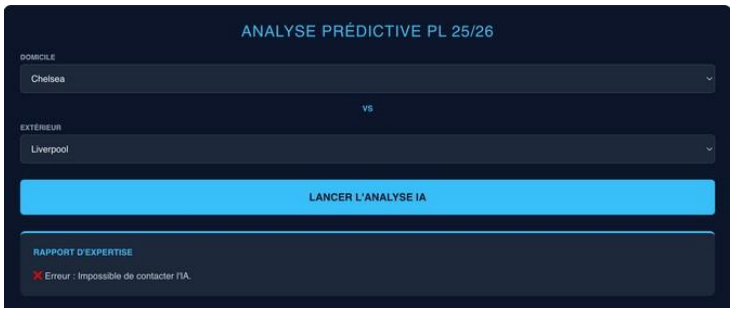
7.2 Webhook (Website Input)

This is the entry point of the workflow.

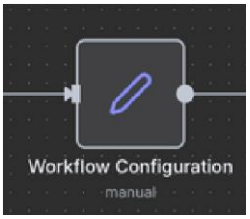
It receives an HTTP POST request from the website containing:

- The match description (example: "Team A vs Team B") A
- session identifier (optional)

The webhook allows the workflow to be triggered automatically whenever a user submits a request on the website.



7.3 Workflow Configuration



This component standardizes the incoming data.

It extracts and stores:

- userMessage: the text sent by the user sessionId:
- used to track or group requests

This step ensures that all later components receive clean and consistent input fields.

7.4 Parse Team Names

This component analyzes the user's text input and extracts the two teams involved in the match.

It uses pattern matching to detect common formats such as:

- "Team A vs Team B"
- "Team A v Team B"
- "Team A - Team B"

The output of this step

- is: team1 team2
- These two variables are then used by all subsequent paths.

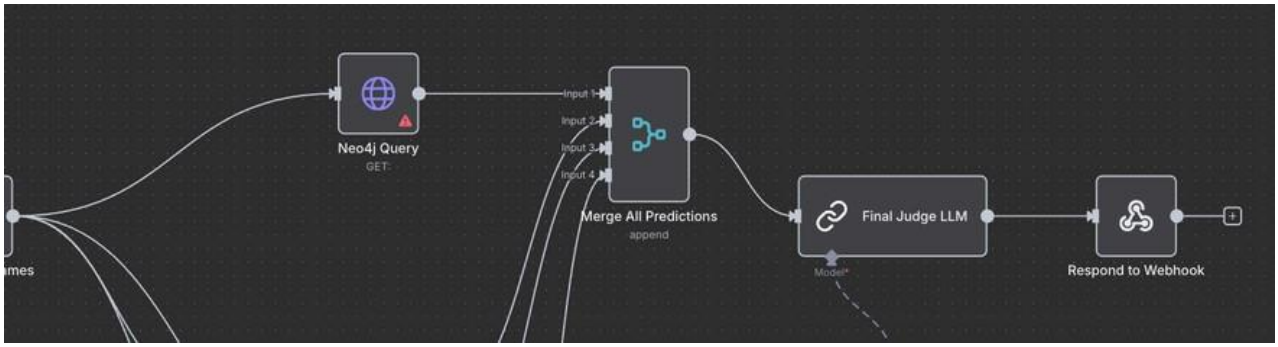
In this project, four machine learning models studied in class were explored.

7.5 Parallel Analysis Paths

After extracting the teams, the workflow splits into four parallel paths.

Each path analyzes the match independently using a different approach.

Path 1 – Neo4j Analysis (Graph Database)



Purpose

This path queries a Neo4j graph database to retrieve historical relationships between teams.

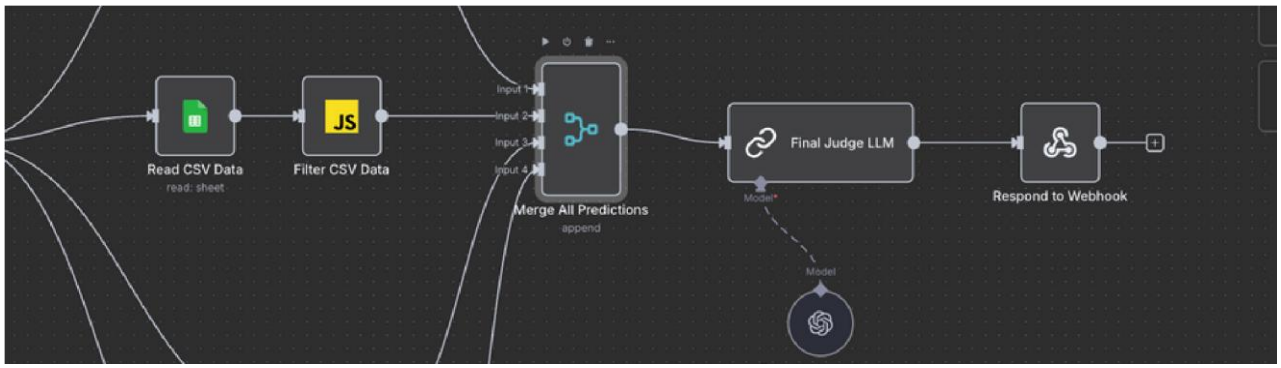
How it works

- The workflow sends a request to Neo4j using the two team names.
- Neo4j searches for past matches where these teams played against each other. The graph
- structure makes it easy to analyze relationships such as head-to-head history and recurring patterns.

Role in the workflow

This path provides relationship-based historical data, which is useful for understanding long-term trends between the two teams.

Path 2 – CSV / Google Sheets Analysis



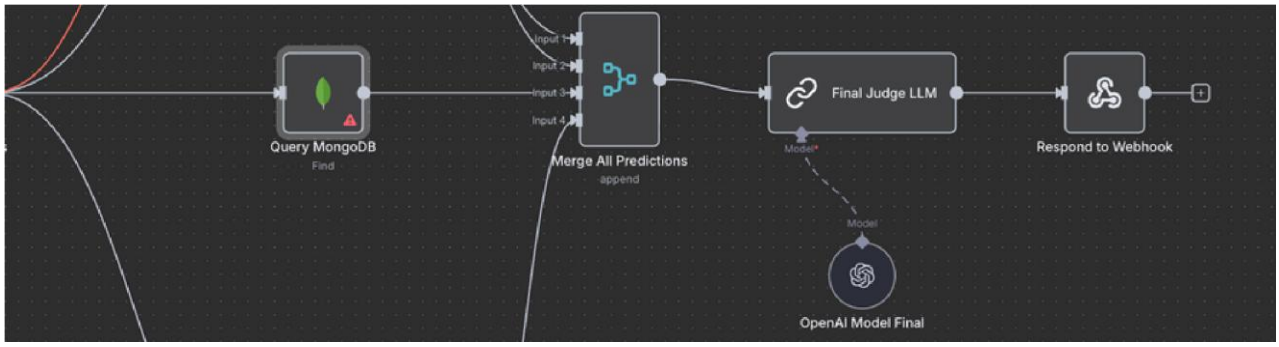
Purpose
This path uses structured historical match data stored in a CSV file hosted in Google Sheets.

How it works

- The workflow reads all rows from the dataset.
- It filters the rows to keep only matches involving team1 and team2.
- A limited number of relevant matches (for example, the most recent ones) are kept.

Role in the workflow
This path provides reliable, tabular historical data such as previous scores and match outcomes.

Path 3 – MongoDB Analysis



Purpose
This path retrieves stored insights from a MongoDB database.

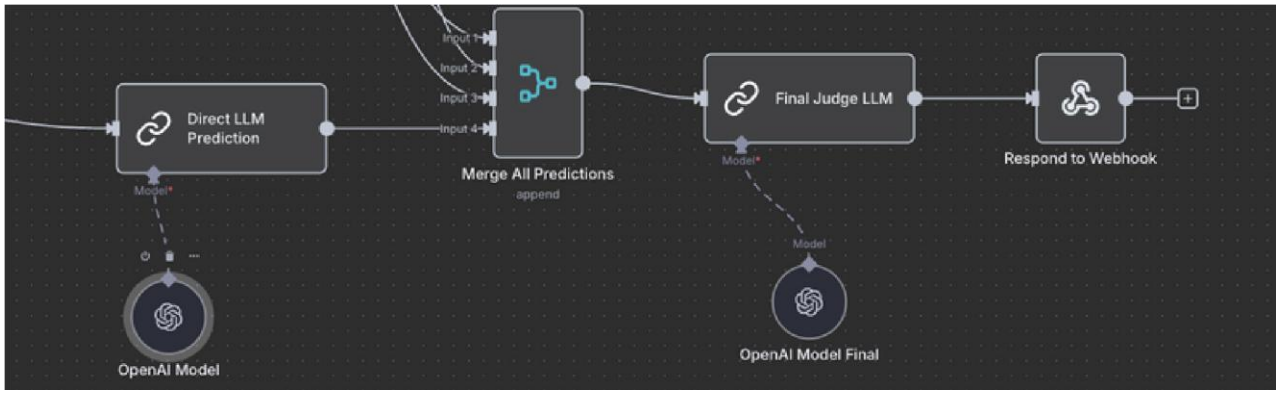
How it works

- MongoDB is queried using the team names or the session identifier.
- It may return previously stored statistics, cached predictions, or custom insights.

Role in the workflow

This path acts as a context and memory layer, allowing the system to reuse previously computed information and enrich the final decision.

Path 4 – Direct LLM Prediction (No Data)



Purpose

This path generates a prediction using only the language model's football knowledge.

How it works

- The language model receives the two team names.

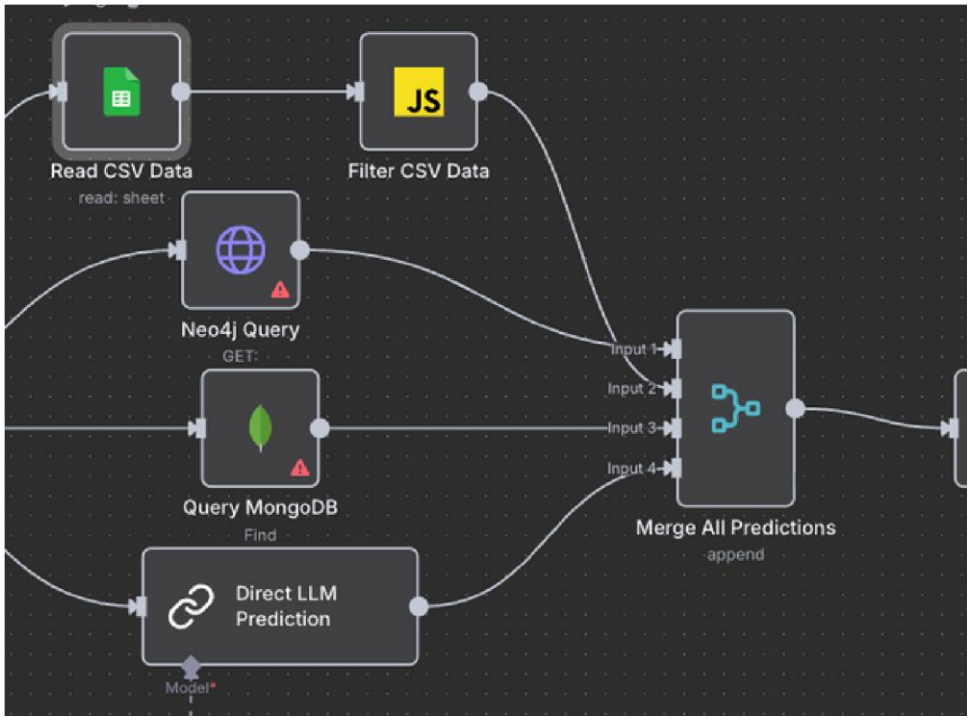
It produces a prediction based on general football reasoning, tactics, and typical team performance.

Role in the workflow

This path serves as:

- A fallback when data sources are missing or weak.
- A complementary reasoning source to compare against data-driven results.

7.6 Merging Results



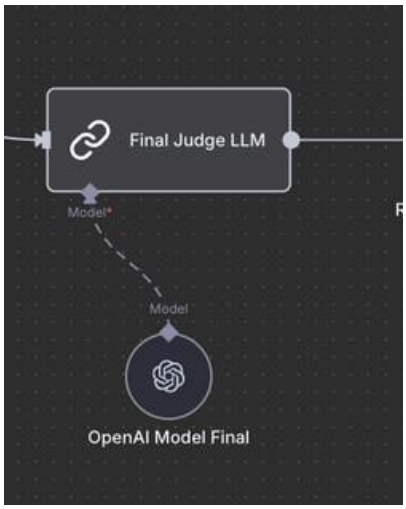
7.7 Merge All Predictions

This component collects the outputs of the four analysis paths.

Each path contributes one result, and all of them are merged into a single combined input.

This step is necessary so the final decision model can compare all analyses together.

7.8 Final Decision Component (Final Judge LLM)



Purpose

This component acts as the final decision-maker.

How it works

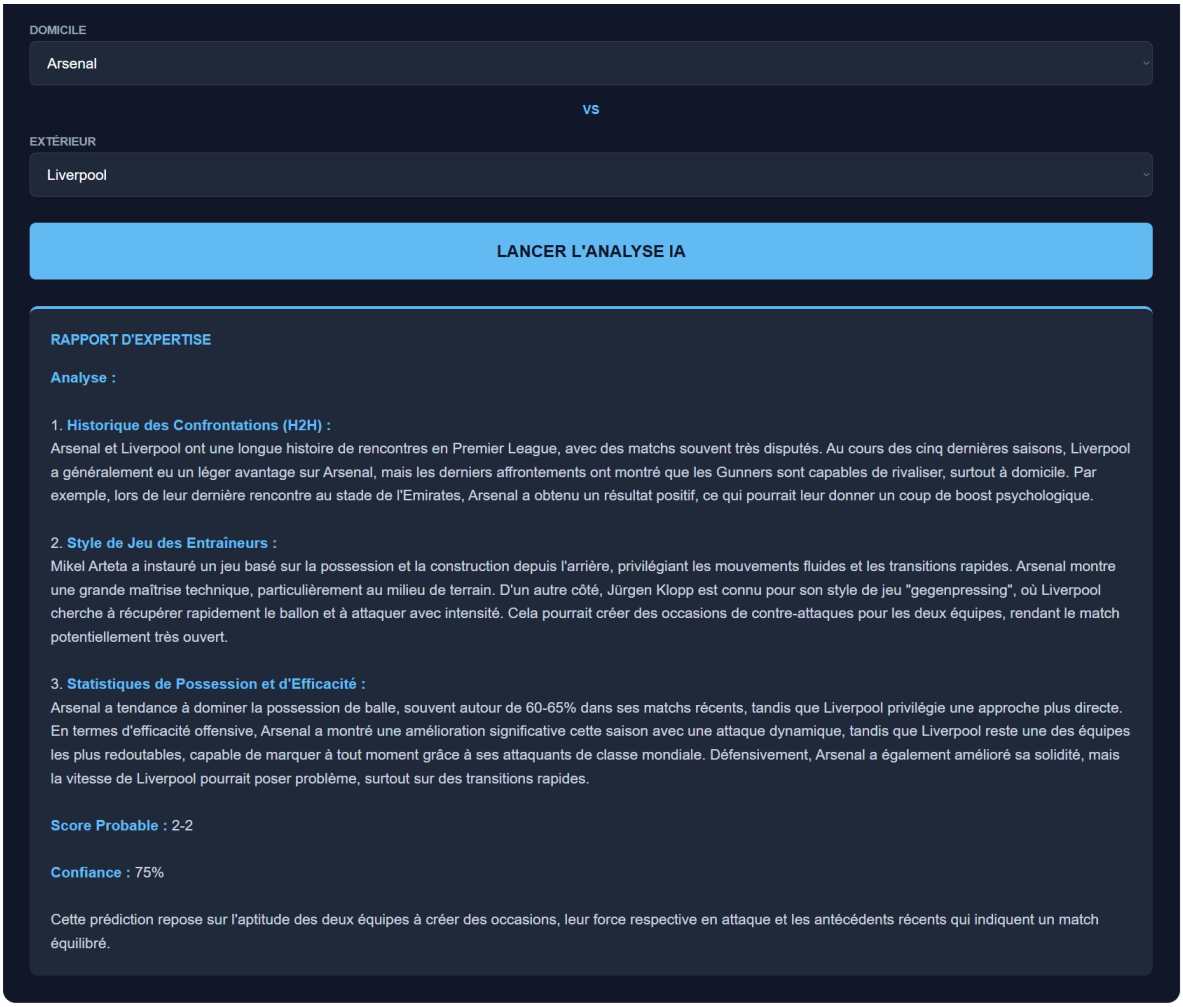
- It reads the merged results from all available paths.
- It identifies which data sources are present or missing.
- It gives higher importance to data-driven sources (Neo4j and CSV).
- MongoDB insights are used as supporting information.
- The direct LLM prediction is used only when data is weak or conflicting.

Output

The model produces:

- The predicted match outcome (home win, away win, or draw)
- A probable score
- A confidence level
- A concise explanation of the reasoning

7.9 Response to Website



Respond to Webhook

This final component sends the prediction back to the website in JSON format.

The website can then display the result directly to the user.

7.10 Summary

This workflow uses a multi-path architecture where each path independently analyzes the same match using different data sources or reasoning strategies.

By merging these paths and using a final judging model, the system produces a single, well-reasoned prediction that is more robust than relying on one source alone.

8. WordPress Integration and System Activation

This phase of the project focuses on establishing a seamless, real-time communication channel between the **WordPress front-end** and the **n8n logic engine**. The goal is to allow users to request predictions and receive dual-layered analysis (General AI + CSV Data) without leaving the page.

8.1. Architectural Overview (Webhook Communication)

The integration is built on a **Webhook-driven architecture**.

- **Trigger:** When a user enters two team names and clicks "Predict," a JavaScript event is triggered.
- **Data Transfer:** The names of the teams are sent via an **HTTP POST request** to the n8n Production URL.
- **Payload Structure:** The data is transmitted in a standard JSON format:

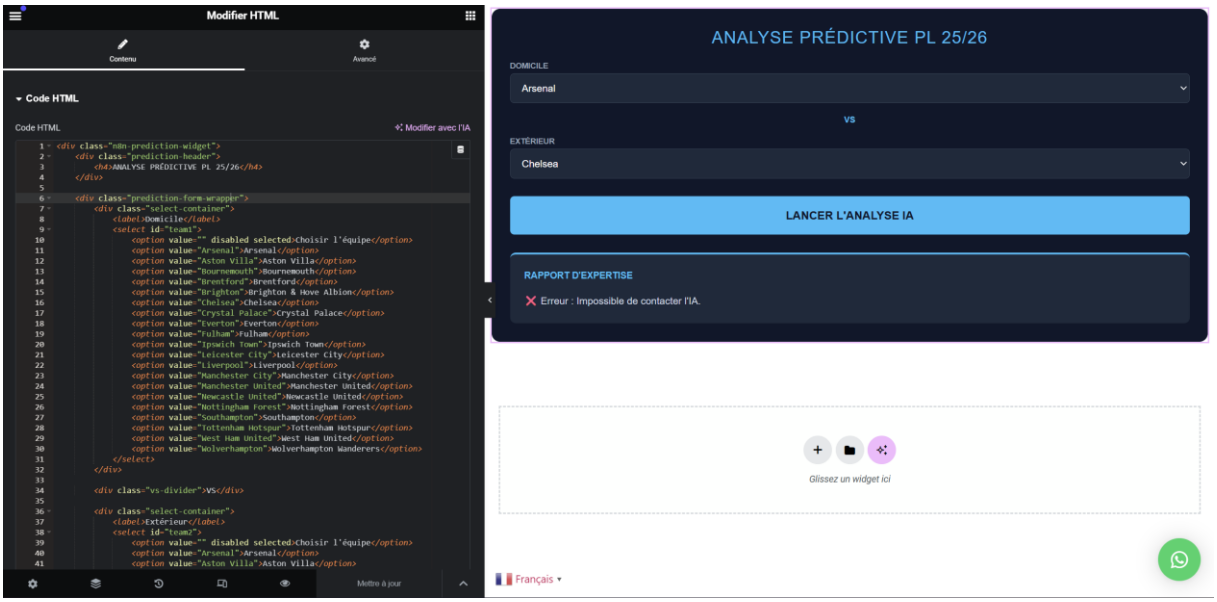
JSON

```
{
  "team1": "Arsenal",
  "team2": "Man City"
}
```

- **Processing & Response:** n8n orchestrates the parallel analysis and returns a consolidated string containing both the AI prediction and the historical data analysis.

8.2 Implementation Code (PHP & JavaScript)

To activate the feature, a custom **Shortcode** was developed. This code is placed in the functions.php file of the WordPress theme. It handles the UI generation and the asynchronous API call.



8.3 Activation Procedure

The deployment followed a strict three-step verification process:

1. **Production URL Mapping:** The n8n workflow was switched from "Test" to **"Production"** mode. This ensures the Webhook URL remains permanent and capable of handling multiple requests.
2. **Shortcode Deployment:** The shortcode [match_predictor] was embedded into a WordPress page via the Gutenberg editor.
3. **CORS & Security:** The server environment was configured to allow Cross-Origin Resource Sharing (CORS) between the WordPress domain and the n8n instance to prevent browser-side blocking.

9. Prompt Engineering (Detailed)

In this project (n8n + AI + Neo4j), **prompt engineering** is used to guide the AI model so it produces outputs that are **accurate, consistent, and directly usable by automation**. Since n8n workflows are sequential (API → Neo4j → AI → storage → notification → evaluation), the AI response must be **structured (JSON)**, follow strict rules, and avoid irrelevant text.

9.1 Goals of Prompt Engineering in This Project

1. **Standardize the output format**

- n8n needs a stable response structure to parse results and store them in Neo4j.
- We define fixed fields like: pick, probabilities, reasons, confidence, modelVersion.

2. **Improve reasoning quality**

- Football prediction is multi-factor (form, home/away performance, goals, head-to-head, rest days).
- Prompts are designed to ensure the model considers multiple features before deciding.

3. **Reduce hallucinations and errors**

- The prompt forces the model to rely only on provided statistics (no invented injuries, transfers, etc.).

4. **Ensure automation compatibility**

- The response must be short, deterministic, and machine-readable (valid JSON only).

5. **Make the system extensible**

- The same prompts can be reused even if we add new features later (ELO, xG, injuries, odds, etc.).

9.2 Prompt Engineering Techniques Used

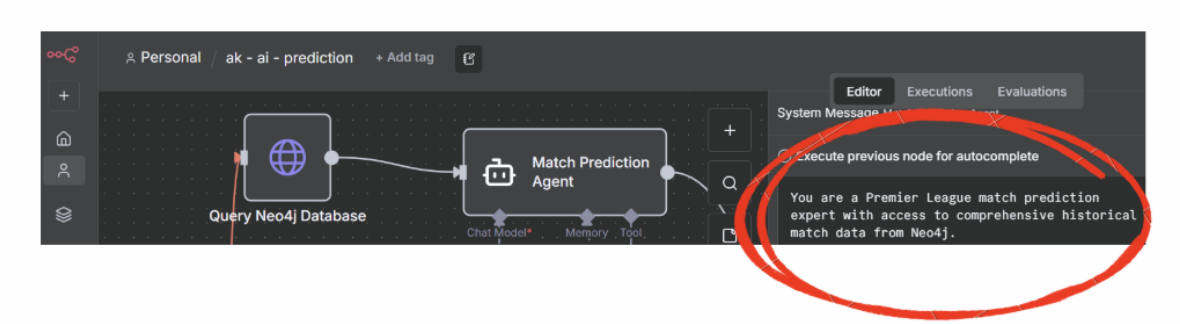
A) Role Playing

Concept: Assign a clear expert role to the AI model and define its mission.

Why it helps in this project:

- The model behaves like a **Premier League analyst / data scientist**, which improves relevance and consistency.
- It reduces generic answers and keeps the response focused.

Example (simplified):



Impact:

- More professional, focused outputs
- Less unnecessary text
- More stable predictions across matches

B) Chain-of-Thought (CoT) Prompting (Controlled Reasoning)

Concept: Encourage step-by-step reasoning to handle complex decisions.

Why it helps here:
A match prediction requires analyzing multiple factors such as:

- recent form (last 5 games)
- home vs away performance
- goals scored/conceded
- head-to-head history
- rest days

Important for automation:
We don't want the model to output long reasoning text.
So we use a *controlled* approach:

- ✓ model reasons internally
- ✓ but returns only: probabilities + final pick + short reasons.

Example instruction:

```
Your task is to:
1. **Analyze Head-to-Head Record**:
   - Who dominates this fixture historically?
   - Recent trends in their meetings
   - Typical score patterns when they play

2. **Evaluate Current Form**:
   - Which team is in better form recently?
   - Scoring and defensive trends
   - Momentum and consistency

3. **Consider Home/Away Factors**:
   - Home advantage impact
   - Away team performance on the road

4. **Assess Goal Patterns**:
   - Average goals scored/conceded
   - High-scoring or defensive matches?
   - Clean sheet likelihood

5. **Make Your Prediction**:
   - Winner (or draw prediction) with confidence level
   - Probable score based on data patterns
   - Detailed reasoning citing specific statistics
```

Impact:

- Better logical consistency
- More realistic probability distribution
- Short explanations that fit notifications

C) Few-Shot Prompting

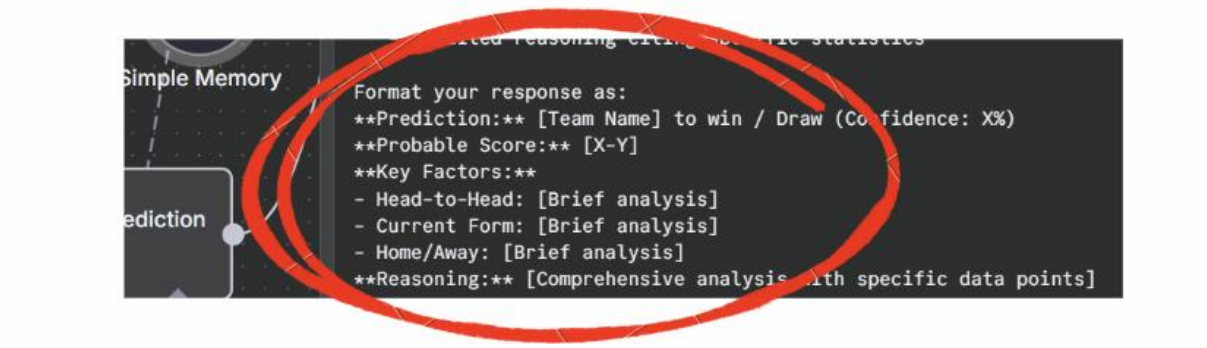
Concept: Provide a few examples (input → expected output) so the model copies the exact format and decision style.

Why it helps here:
Without examples, outputs may vary:

- missing fields
- invalid JSON

- probabilities not summing to 1
- reasons too long

Approach used:



- Provide 2–5 sample matches with stats and correct JSON output
- Ask the model to predict the new match using the same schema

Impact:

- Strong improvement in output consistency
- Reliable JSON formatting for production automation

9.3 Guardrails (Rules to Prevent Failures)

To make the AI safe for automated pipelines, we enforce strict constraints:

- **Return ONLY valid JSON** (no extra text)
- Probabilities must be between **0 and 1**
- Probability sum must be **exactly 1**
- **Maximum 3 reasons**
- **No invented data**: if something is missing → use "unknown"
- Always include **confidence** level (Low / Medium / High)

These rules are critical because n8n must parse the response and store it correctly in Neo4j.

9.4 Standard Output Format Used (AI Response)

Example of the final output schema (compatible with n8n + Neo4j):


```
{
 "matchId": "EPL_2026_ARS_MCI",
 "pick": "AWAY",
 "probabilities": {
  "HOME": 0.25,
  "DRAW": 0.27,
  "AWAY": 0.48
 },
 "reasons": [
  "Away team has better form in the last 5 matches",
  "Home team concedes frequently at home",
  "Recent head-to-head results favor the away team"
 ],
 "confidence": "Medium",
 "modelVersion": "v1.0"
}
```


9.5 How It Integrates into n8n

In the n8n workflow:

1. n8n fetches fixtures from a football API
2. n8n queries Neo4j (Cypher) to compute features
3. n8n sends features + prompt to the AI model
4. AI returns JSON prediction

5. n8n validates the JSON format
6. n8n stores predictions in Neo4j
7. n8n sends notifications (Email/Telegram/Discord)
8. After matches end: n8n fetches results and evaluates model performance

9.6 Result and Benefits

Using **Role Playing + Controlled CoT + Few-Shot** improved the system by:

- stable, machine-readable predictions
- better reasoning and consistency
- fewer formatting failures in n8n
- easier storage and evaluation in Neo4j
- easy future upgrades (more features / retraining loop)

10. Conslusion

This report presented the design and implementation of a complete football match result prediction system that combines **Artificial Intelligence**, **workflow automation with n8n**, and **graph-based data modeling using Neo4j**. Starting from the initial challenge of handling heterogeneous and noisy football datasets, the project successfully built an end-to-end pipeline capable of collecting historical match data, cleaning and transforming it, storing it in a structured graph representation, and generating predictions through multiple complementary analysis strategies.

A key contribution of the system is its **multi-path architecture**, where the same match request is analyzed in parallel using different sources and reasoning methods (Neo4j graph queries, CSV/Google Sheets analysis, MongoDB context reuse, and direct LLM reasoning). By merging these outputs and applying a final "judge" decision component, the workflow increases robustness and reduces the risk of relying on a single model or dataset. This design also improves system reliability when one data source is missing or produces conflicting results.

The project also demonstrated the importance of **Prompt Engineering** for building trustworthy AI within an automated environment. Techniques such as **Role Playing**, **Controlled Chain-of-Thought**, and **Few-Shot prompting**, combined with strict JSON guardrails, ensured consistent and machine-readable outputs that integrate smoothly with n8n workflows and database storage.

Finally, the integration with **WordPress** proved the feasibility of real-world deployment by enabling users to request predictions through a simple web interface and receive structured results in real time. Overall, this project highlights a modern approach to sports analytics by merging **automation, scalable graph data management, and AI-driven decision support**, while also offering a strong foundation for future improvements such as richer features (ELO/xG/odds), advanced ML training, continuous evaluation, and model retraining for higher predictive accuracy.