ENGLISH PREMIER LEAGUE MATCH PREDICTION

A MINI PROJECT REPORT

Submitted by

Jayavelan S (Reg. No. 9517202309041)

Nandhagopal V (Reg. No. 9517202309074)

Sanjay K N (Reg. No. 9517202309101)

in partial fulfillment for the requirement of the course

23AD453 – MINI PROJECT II: DATA ANALYTICS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE (AUTONOMOUS)

SIVAKASI

MAY 2025

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work done by **Jayavelan S (9517202309041), Nandhagopal V (9517202309074), Sanjay K N (9517202309101)** for the course **23AD453 - Mini Project – II: DATA ANALYTICS** work entitled **"English Premier League Match Prediction"** at Department of Artificial Intelligence & Data Science, Mepco Schlenk Engineering College, Sivakasi during the year 2024 – 2025.

…………………………	……………………………
Faculty In-Charge	***Head of the Department***

Submitted for the Practical Examination held at Mepco Schlenk Engineering College, Sivakasi on ……………………….

…………………………	……………………………
Internal Examiner - I	***Internal Examiner - II***

ACKNOWLEDGEMENT

First and foremost, we express our deepest gratitude to the **LORD ALMIGHTY** for His abundant blessings, which have guided us through our past, present, and future endeavors, making this project a successful reality.

We extend our sincere thanks to our management, esteemed Principal, **Dr. S. Arivazhagan, M.E., Ph.D.,** for providing us with a conducive environment, including advanced systems and well-equipped library facilities, which have been instrumental in the completion of this project.

We are profoundly grateful to our Head of the Artificial Intelligence and Data Science Department, **Dr. J. Angela Jennifa Sujana, M.Tech., Ph.D.,** for granting us this golden opportunity to work on a project of this significance and for her invaluable guidance and encouragement throughout.

Our heartfelt appreciation goes to our project guides, **Dr. R. Nirmalan, M.Tech., Ph.D.,** Assistant Professor, and **Mrs.M. Revatheeswari, M.Tech.,** Assistant Professor, Department of Artificial Intelligence and Data Science for their unwavering support, insightful suggestions, and expert guidance. Their experience and encouragement have been vital in shaping our project and helping us bring forth our best.

We also extend our sincere gratitude to all our faculty members, lab technicians, and our beloved family and friends, whose timely assistance and moral support played a pivotal role in making this mini-project a success.

ABSTRACT

In the modern era of data-driven decision making, sports analytics has emerged as a crucial tool for enhancing performance evaluation and strategy development. This project applies various data mining and machine learning techniques to the English Premier League (EPL) dataset to extract meaningful insights, discover hidden patterns, and predict match outcomes. The study primarily focuses on three core areas: association rule mining, classification, and clustering.

In the first phase, Association Rule Mining techniques, specifically the Apriori Algorithm and the FP-Growth Algorithm, are used to identify significant relationships between different match factors such as team performance, goals scored, assists, and match results. These algorithms help in uncovering frequent patterns and generating rules based on support, confidence, and lift values. Such insights allow for a better understanding of the factors that often contribute to winning or losing a match.

The second phase involves building Classification Models to predict match outcomes. Three prominent classifiers are explored: Decision Tree Classifier (using the ID3 algorithm), Naïve Bayes, and K-Nearest Neighbor (KNN). Among these, the ID3 decision tree algorithm demonstrates superior performance by effectively splitting the data based on information gain.

The classifiers are evaluated using various performance metrics such as accuracy, precision, recall, specificity, F1-score, ROC-AUC, and through visualization tools like the ROC Curve and Confusion Matrix. The goal of this phase is to build predictive models that can accurately forecast the result of a football match based on historical data. In the third phase, Clustering Algorithms such as k-Means, Agglomerative Clustering, and DBSCAN are employed to segment teams and players based on their performance metrics. Clustering provides an unsupervised approach to discovering natural groupings within the dataset, helping to identify similarities and differences among teams or players without relying on predefined labels. Evaluation metrics like Silhouette Score and Dunn Index are utilized to assess the quality of the clusters formed.

In conclusion, this project demonstrates the practical application of machine learning and data mining techniques in sports analytics, specifically within the domain of football. By combining association rule mining, predictive modeling, and clustering, valuable patterns and predictive insights are uncovered, offering potential benefits for coaches, analysts, and teams seeking data-driven strategies to enhance performance and gain a competitive advantage.

List of Tables

List of Figures

List of Symbols & Abbreviations

Symbol/Abbreviation	Description
ID3	Iterative Dichotomiser 3
Cart	Classification And Regression Tree
C4.5	CART 4.5
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ROC Curve	Receiver Operating Characteristic
KNN	K – Nearest Neighbours

SYNOPSIS

Module 1: Problem Statement Identification

This module is dedicated to understanding the problem at hand and structuring data effectively for analysis. It emphasizes data warehouse schemas, which form the backbone of organizing and managing large datasets. By leveraging the Star, Snowflake, and Constellation schemas, students can structure their data efficiently. Additionally, the module covers the analytical operations of Online Analytical Processing (OLAP), which facilitates multidimensional data analysis, aiding in problem scoping and decision-making.

Module 2: Data Acquisition

This module focuses on gathering data from various sources and preparing it for analysis. Students will explore different dataset types, including structured (tabular data), semi-structured (JSON, XML), and unstructured (text, images, audio). The module ensures proper categorization of attributes into numerical, categorical, ordinal, and nominal data. Further, students will learn to identify independent (features) and dependent (target) attributes, derive new attributes when necessary, and perform feature selection to retain only the most relevant attributes for analysis. The ability to source and organize high-quality data is fundamental to ensuring meaningful insights in later stages.

Module 3: Preprocessing of Datasets

Before applying machine learning algorithms, raw data must undergo preprocessing to enhance its quality and usability. This module teaches students to clean data by handling missing values, detecting and removing outliers, and eliminating duplicate records. Using Weka, students will perform data normalization to scale features within a standard range, standardization to achieve a consistent mean and variance, and transformation techniques such as one-hot encoding and discretization. Additionally, dimensionality reduction techniques, like Principal Component Analysis (PCA), will be explored to optimize model performance while reducing computational complexity.

Module 4: Model Generation

Building predictive models is the core objective of this module. It covers different machine learning techniques, including association rule mining, regression, classification, and clustering models. Students will work with algorithms like Apriori and FP Growth for association rule mining, various regression models (linear, multiple linear, polynomial, logistic), and classification models such as Decision Trees (ID3) and Naïve Bayes. For clustering, methods like k-means, Agglomerative Clustering, and DBSCAN will be applied. This module provides a hands-on approach to selecting and implementing suitable models for diverse data-driven problems.

Module 5: Testing of Model

Once a model is built, evaluating its performance is crucial to ensure its reliability and accuracy. This module introduces various evaluation metrics tailored for different models. For classification models, key metrics include accuracy, precision, recall, F1-score, and ROC-AUC. Regression models will be evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-Squared. Clustering models will be assessed using the Silhouette Score, Dunn Index, and Davies-Bouldin Index, while association rule mining will involve Support, Confidence, and Lift. Additionally, visualization techniques such as confusion matrices, ROC curves, scatter plots, and heatmaps will help interpret model results effectively.

Module 6: Deployment

The final step in the data science workflow is deploying the developed models for practical use. This module focuses on finalizing the trained models, integrating them into a user-friendly interface, and ensuring their seamless deployment. Students will learn how to prepare the final model, develop a functional UI for interaction, and host the model on appropriate platforms. This stage bridges the gap between theoretical learning and real-world implementation, allowing users to interact with the model in a meaningful way.

Table of Contents

CHAPTER 1
INTRODUCTION

Predicting the winner of a football tournament is a tiresome task that involves analyzing large and complex datasets comprising historical game statistics, team performance metrics, and contextual factors. Our project aims to address this challenge by designing a predictive system capable of forecasting game outcomes and determining the overall tournament winner. To structure and manage the data effectively, the project leverages data warehouse schemas and Online Analytical Processing (OLAP) techniques for multidimensional analysis and decision-making.

1.1 Overview of the Project

The project focuses on predicting the winning team of English Premier League by creating a virtual tournament of 8 teams using historical game data and machine learning techniques. By analyzing team performance metrics, game contexts, and other relevant features, the project aims to provide accurate predictions for individual games and simulate the progression of an entire tournament to identify the ultimate winner.

1.2 Objectives

1.2.1 Game-Level Prediction:

Develop a machine learning model to predict the outcome (win/loss) of individual football games using historical data.

1.2.2 Tournament Simulation:

Simulate the progression of a football round-of-8 tournament by iteratively predicting game outcomes and advancing winners through the bracket.

1.2.3 Insights and Interpretations:

Uncover the key factors influencing game outcomes and tournament success, providing actionable insights for teams, analysts, and fans.

1.3 Scope of the Project

The scope of this project is to develop a predictive system capable of forecasting the winner of a basketball tournament. The system will analyze historical game data, team performance metrics, and contextual factors using machine learning and data analytics techniques. The project aims to address both game-level outcome predictions and tournament-level winner forecasting, providing insights and tools for stakeholders such as sports analysts, teams, and fans.

1.4 Tools & Programming Languages Used

This project utilizes a combination of tools and programming languages to ensure efficient data analysis, visualization, model development, and deployment. Below is an overview of the tools and programming languages used:

1.4.1 Python

Python serves as the core programming language for this project. Its extensive ecosystem of libraries makes it ideal for tasks such as data preprocessing, feature engineering, model development, and deployment. Key Python libraries used include pandas for data manipulation, numpy for numerical computations, scikit-learn for machine learning, and matplotlib and seaborn for data visualization.

1.4.2 Jupyter Notebook

Jupyter Notebook provides an interactive environment for performing data analysis and visualization. It is extensively used in this project for exploratory data analysis, iterative development, and documentation.

1.4.3 Orange

Orange is an open-source data visualization and analysis tool that supports visual programming. It is used for creating workflows, visualizing data, and experimenting with machine learning models in a user-friendly, drag-and-drop interface.

1.4.4 Weka

Weka, a collection of machine learning algorithms for data mining tasks, is employed in this project for advanced analytics and experimentation with various machine learning models.

1.4.5 Flask

Flask is a Python-based framework used to create work flow between different webpages.

1.4.6 Html & CSS

Html is used to create a webpage with its skeleton and CSS is used to style the webpage for better UI experience

CHAPTER 2
PROBLEM STATEMENT IDENTIFICATION

2.1 Understanding the Problem

The primary challenge in this project is predicting the winning team of a football tournament. This involves analyzing historical basketball game data and using it to forecast future outcomes, both at the individual game level and the tournament level. To address this problem comprehensively, we must break it into several components:

2.1.1 Data Understanding

The project relies on historical football game data, which includes team performance metrics (shots and betting ratings), contextual factors (home/away games), and game outcomes (scores, win/loss). Understanding the structure, quality, and completeness of this data is critical for building an effective prediction model.

2.1.2 Defining the Objective

The objective is twofold:

- Predict the outcome of individual football games based on historical data.
- Use these predictions to simulate the progression of a tournament and identify the ultimate winner.

2.1.3 Key Challenges

Data Quality and Availability: Ensuring the dataset is clean, complete, and representative of the factors affecting game outcomes.

- Feature Selection: Identifying the most relevant features (e.g., offensive/betting ratings, field foul counts, home-stadium advantage) that influence game results.
- Tournament Dependencies: Accounting for the fact that the outcome of one game influences subsequent matches in the tournament bracket.
- Model Accuracy: Balancing the precision and generalization of the predictive model to ensure robust predictions across different teams and tournaments.

2.1.4 Approach to the Problem

Data Structuring: Organize the data using appropriate schemas (e.g., Star Schema, Snowflake Schema) for efficient storage and analysis.

- Data Analysis: Use tools like Jupyter Notebook for exploratory data analysis and visualization to understand trends and patterns.
- Machine Learning: Train a predictive model using tools like Python, Orange, Weka, and scikit-learn to forecast individual game outcomes.
- Tournament Simulation: Develop a framework to simulate tournament brackets using the game-level predictions.
- Deployment and Visualization: Use Flask for deploying the model as an interactive application and Plotly for creating dashboards to display insights.

2.1.5 Outcome

The project aims to provide a robust system for predicting football tournament winners. It also seeks to deliver valuable insights into the factors driving team success, which can be used by analysts, teams, and fans to enhance their understanding of the game.

2.2 Data Warehouse Design

A well-designed data warehouse is essential for storing and managing historical basketball data efficiently. This project incorporates three popular schema designs—Star, Snowflake, and Constellation—to organize the data for analytical purposes. These schemas facilitate Online Analytical Processing (OLAP) operations and make the data accessible for multidimensional analysis, enabling better decision-making and accurate predictions.

2.2.1 Star Schema

Star Schema is a type of multidimensional model used for data warehouses. In a star schema, the fact tables and dimension tables are included. This schema uses fewer foreign-key joins. It forms a star structure with a central fact table connected to the surrounding dimension tables.

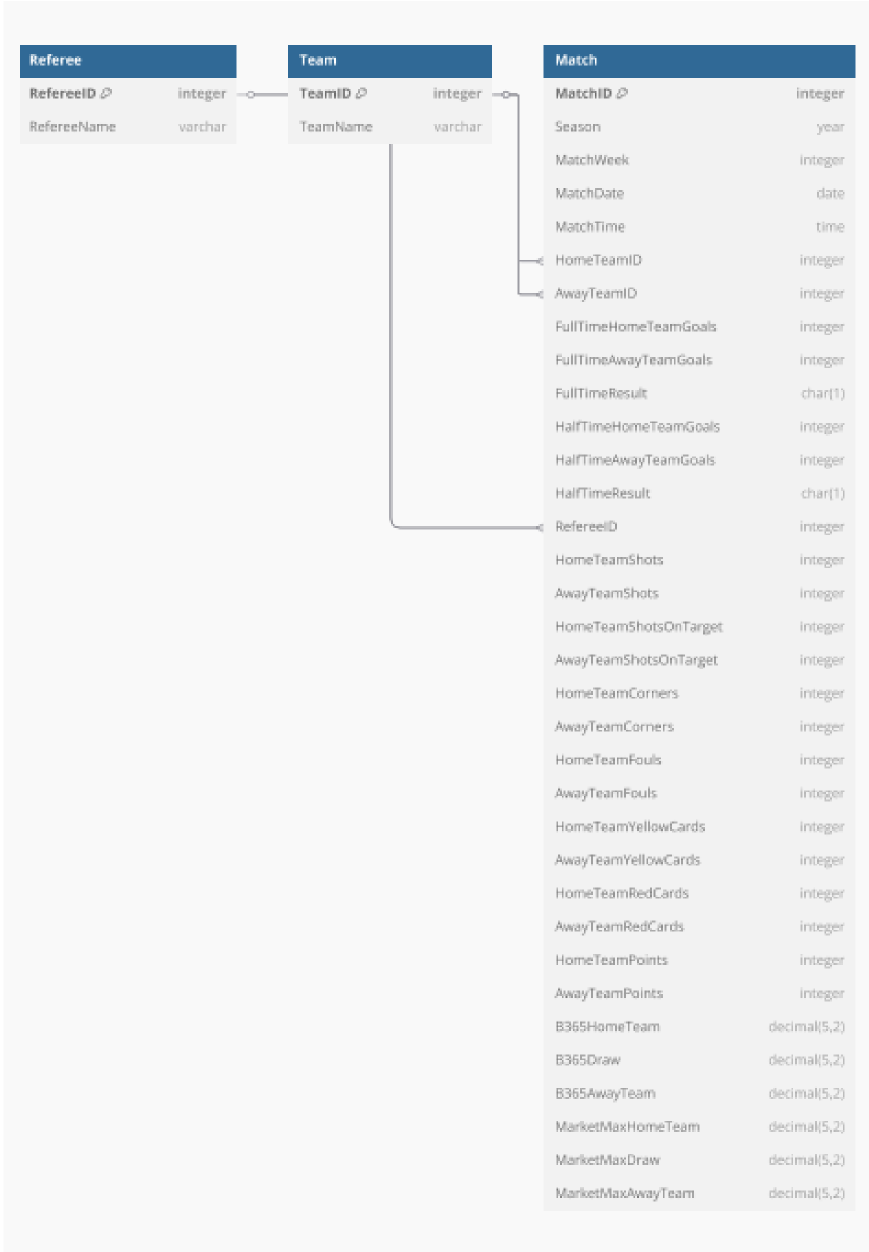

Fig 2.1 Star schema

2.2.2 Snowflake Schema

Snowflake Schema is also a type of multidimensional model used for data warehouses. In the snowflake schema, the fact tables, dimension tables and sub-dimension tables are included. This schema forms a snowflake structure with fact tables, dimension tables and sub-dimension tables.

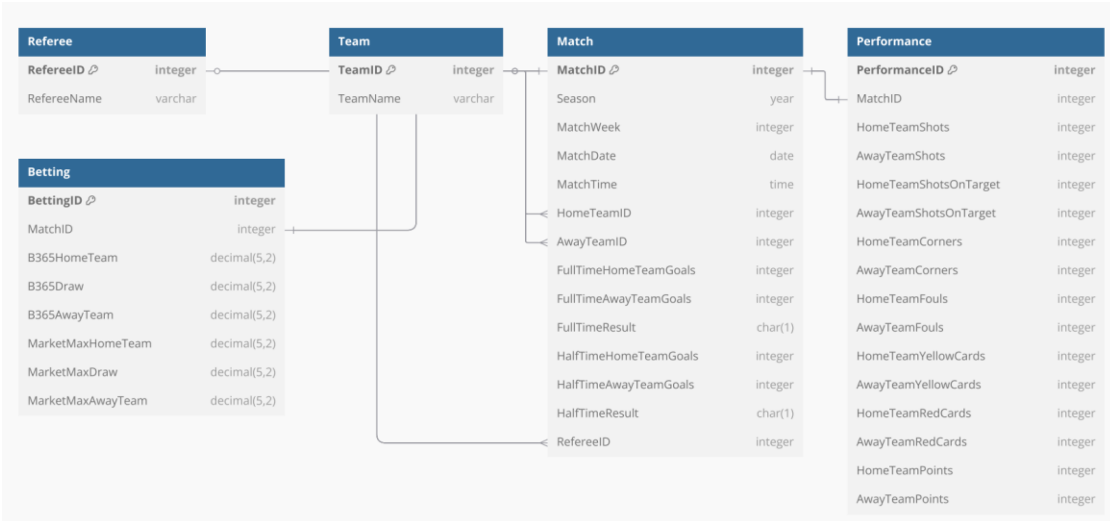

Fig 2.2 Snowflake schema

2.2.3 Fact Constellation

Fact Constellation Schema, also known as the Galaxy Schema, is an advanced data modeling technique used in designing data warehouses. Unlike simpler models like the Star Schema and Snowflake Schema, the Fact Constellation Schema consists of multiple fact tables that share common dimensional tables.

This model is ideal for handling complex systems and large-scale analytical queries, offering flexibility for business intelligence and data mining.

The core components of the Fact Constellation Schema include Fact Tables and Dimension Tables. Fact tables store measurable, quantitative data, such as sales or revenue, while dimension tables store descriptive attributes like time, location, or product. These tables are interconnected, with multiple fact tables sharing the same dimension tables.

2.3 OLAP and Analytical Operations

Online Analytical Processing (OLAP) is a powerful data analysis technique that enables multidimensional exploration of data and supports decision-making processes. In this project, OLAP operations are used to analyze basketball game data effectively, aiding in identifying patterns, trends, and insights crucial for predicting tournament outcomes. Below is an overview of the OLAP operations and their applications in this project:

2.3.1 Slice and Dice

Slice and dice operations are used to focus on specific subsets of data.

Slice: Extracts a single layer of data along a specific dimension. Slicing is done on total matches played by a team with each team to analyse their experience.

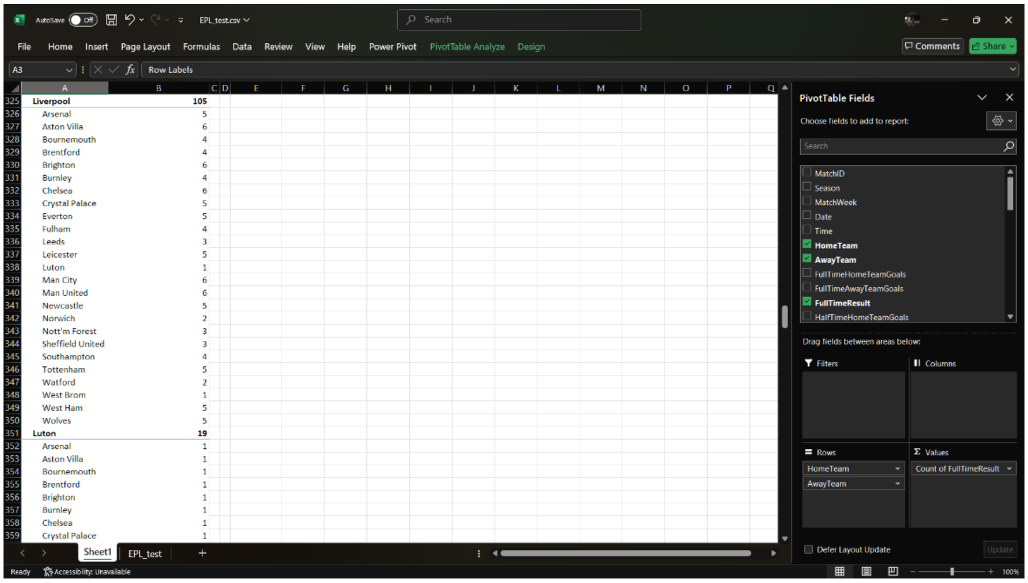

Figure 2.3 : Slice on total matches

Dice: Extracts a sub-cube of data by applying multiple filters. Dicing is done by referee names to determine the foul rate.

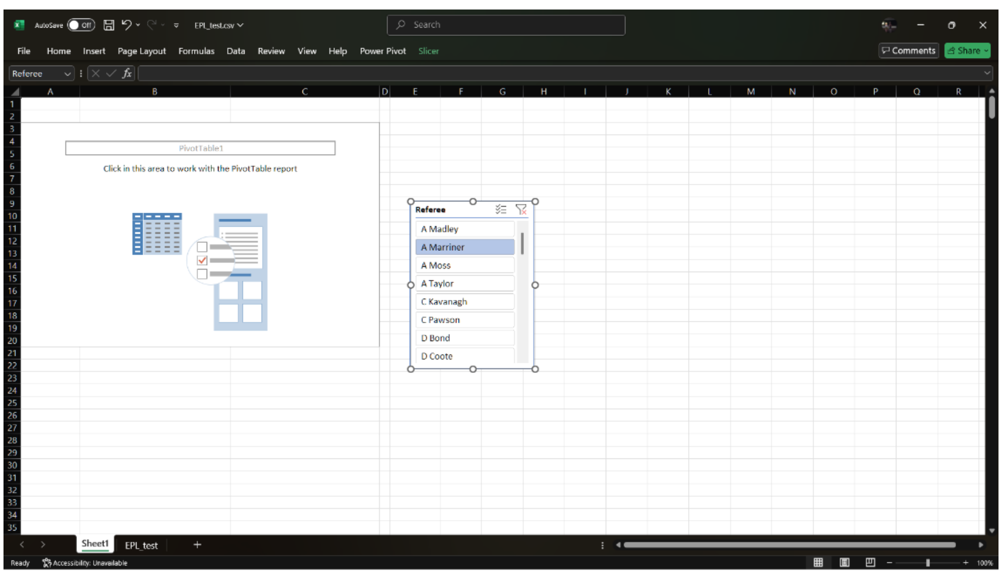

Figure 2.4 : Dice on referee

2.3.2 Roll-Up and Drill-Down

These operations allow aggregation and detailed exploration of data.

Roll-Up: Aggregates data to a higher level of abstraction. For example, rolling up game-level data to calculate team performance metrics at the season level.

Drill-Down: Allows detailed analysis by breaking data down to a finer level. Drilling down from season-level metrics to individual week statistics.

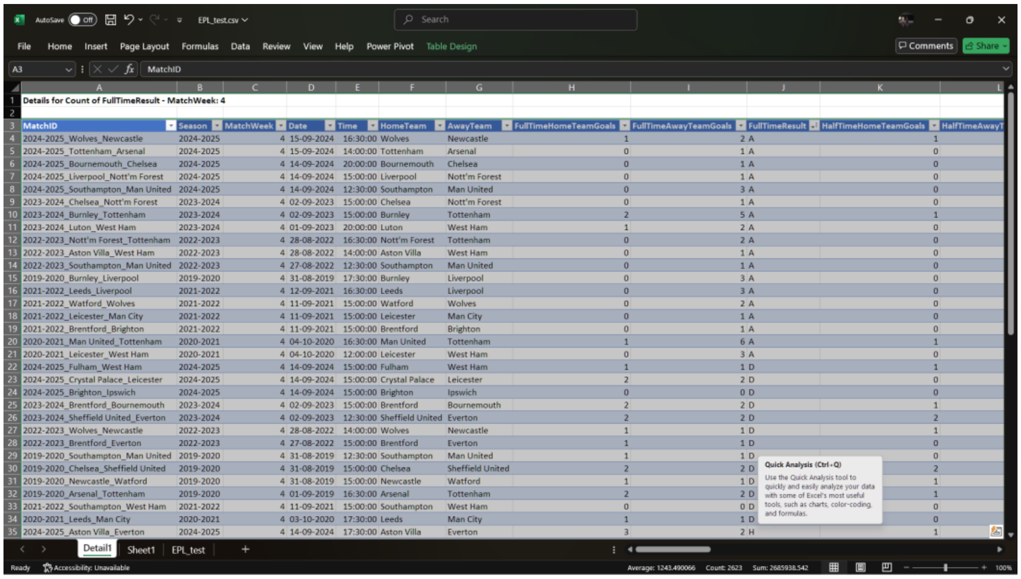

Figure 2.5 : Drill-Down to one Matchweek

2.3.3 Pivot (Cross-tabulation)

Pivoting reorients the data to view it from different perspectives.

Comparing offensive and defensive ratings across teams over multiple seasons by pivoting the data to focus on these metrics.

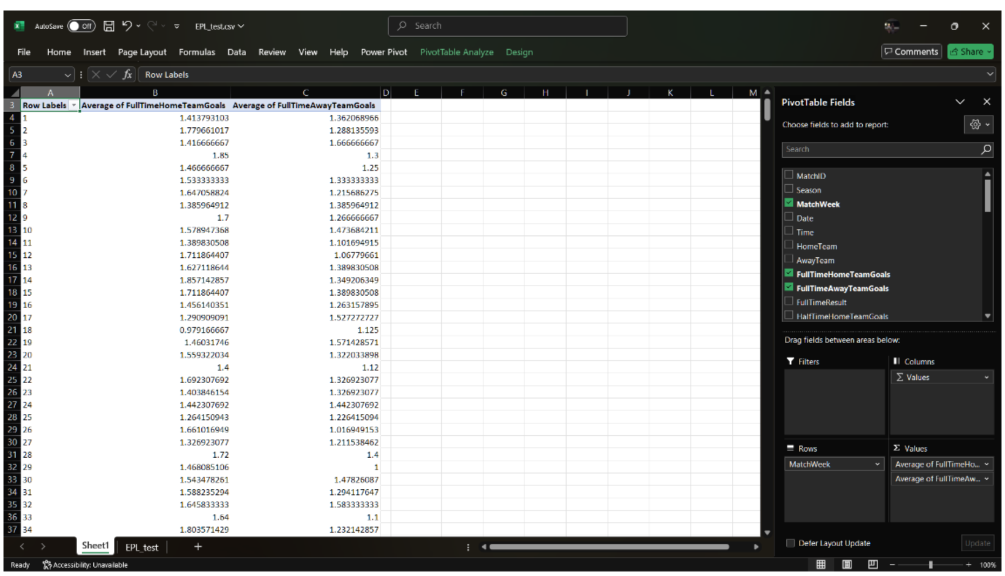

Figure 2.6 : Pivot Table

CHAPTER 3
DATA ACQUISITION

3.1 Data Collection Methods

The success of this project relies heavily on the quality and accuracy of the data collected. To ensure a robust dataset for predicting football match outcomes in the English Premier League (EPL), the following data collection methods have been employed:

3.1.1 Historical Game Data

Historical match data forms the backbone of this project. This includes detailed information about team performance, player statistics, match context, and final outcomes. The data was collected from publicly available sources such as official EPL websites, Kaggle football datasets, and sports analytics platforms.

3.1.2 Data Updation

As the EPL season progresses throughout the year, it is essential to keep the dataset up to date. The dataset is regularly updated after each matchweek to incorporate the latest match results, player performances, and standings.

3.1.3 Public APIs

Football-related APIs such as Football-Data.org, SportRadar, and API-Football are used for real-time and historical data acquisition. These APIs provide structured and reliable data, including match scores, player stats, and in-depth team analytics.

3.1.4 Manual Data Entry

In instances where automated retrieval is not feasible, data points like player injuries, managerial changes, or disciplinary actions are manually entered to ensure completeness and context.

3.1.5 Data Integration

Data from multiple sources is integrated using ETL (Extract, Transform, Load) techniques to create a unified, consistent, and accurate dataset. This process includes cleaning, merging, and aligning data from matches, players, and teams.

3.2 Types of Data

The dataset used in this project comprises various types of data essential for building predictive models and conducting comprehensive analyses. These include:

3.2.1 Match-Level Data

Information capturing the details of individual EPL matches. Examples:

- Date and time of the match
- Home team and away team
- Goals scored by each team
- Referee details
- Fouls committed
- Cards issued (yellow/red)
- Match result (win/loss/draw)

3.2.2 Team Performance Data

Metrics that describe how teams perform both offensively and defensively.

- Total shots and shots on target
- Possession percentage
- Passing accuracy
- Number of corners won
- Number of fouls committed

3.2.3 Contextual Data

Data that provides additional context to matches and player/team performance.

- Home advantage (home vs. away matches)
- Rivalry matches or historical head-to-head records
- Weather conditions (if available)
- Referee behaviour (e.g., card frequency)

3.2.4 Season and Tournament Data

Data related to the overall tournament structure.

- EPL standings
- Points accumulated
- Goal differences
- Qualification for European competitions (Champions League, Europa League)
- Relegation battles

3.2.5 Derived Metrics

Metrics computed from raw data to enhance the analysis and model building.

- Recent form (e.g., last 5 match results)
- Average goals per match
- Win streaks or losing streaks
- Expected goals (xG) estimates

3.3 Attribute Categorization and Classification

Organizing and categorizing attributes in the dataset is crucial for building a robust predictive model. This section outlines the classification of attributes and their roles in the project.

3.3.1 Independent vs. Dependent Attributes

Organizing and classifying attributes is crucial to building effective predictive models. Attributes are classified as follows:

3.3.1.1 Independent Attributes (Features):

These are the inputs used to predict outcomes.

- Home/Away status
- Number of shots on target
- Number of fouls and cards
- Past performance metrics
- Referee assigned

3.3.1.2 Dependent Attributes (Targets):

These represent the outcomes the model aims to predict

- Match result (win/draw/loss)
- Final goal difference
- Points earned (3, 1, or 0)

3.3.2 Feature Selection and Extraction

Selecting the right features and deriving new ones significantly improves model performance:

3.3.2.1 Feature Selection:

Identifying the most influential variables through:

- Correlation analysis with match outcomes
- Statistical methods like Chi-Square tests for categorical data

- Feature importance rankings from models like Random Forest

3.3.2.2 Feature Extraction:

Creating new features that better capture match dynamics:

- Form Index (points earned in last 5 games)
- Attack and Defence strength ratios
- Momentum metrics (e.g., unbeaten runs)
- Head-to-head performance metrics

CHAPTER 4
DATA PREPROCESSING

4.1 Data Cleaning

Data cleaning is a crucial step in ensuring the quality, reliability, and consistency of the dataset. For this project, where the data is sourced primarily from Kaggle football datasets, the following steps have been performed:

4.1.1 Handling Missing Values

Missing values can negatively impact model training and prediction accuracy. The following techniques were used:

- **Deletion**: Attributes with more than 50% missing data were considered for removal if they were non-essential to match outcomes.
- **Imputation**: For **numerical attributes** (like goals, shots, fouls), missing values were replaced using mean or median imputation and For **categorical attributes** (like match venue, referee name), missing entries were filled using the mode (most frequent value).
- **Domain-Specific Logic**: In cases like missing foul counts or cards, domain knowledge was applied (e.g., assuming 0 if no fouls were recorded).

4.1.2 Removing Outliers and Duplicates

- **Duplicate Removal**:
  - Checked for duplicates using unique match identifiers (e.g., Match ID, Date + Home Team + Away Team combinations).
  - Full row comparisons were performed to eliminate exact duplicates.
- **Outlier Detection**:
  - **Visualizations** like boxplots were used to spot extreme values (e.g., unusually high goals or cards).
  - **Thresholding**: Matches with improbable statistics (like more than 10 goals from a team) were flagged and reviewed.
  - **Normalization Techniques**: Z-score standardization was applied where necessary to reduce the influence of extreme outliers.

4.2 Data Normalization and Standardization

Normalization was applied to rescale features between [0, 1] so that no attribute dominates due to its scale.

Purpose: To make all attributes contribute equally to distance-based models like KNN or SVM.

Formula:

$$X_{normalized} = (X - X_{min}) / (X_{max} - X_{min})$$

Normalized Attributes:

- Shots on Target
- Ball Possession %
- Pass Accuracy %
- Attendance (where available)

Standardization

Standardization was applied to center features around mean 0 and standard deviation 1.

Purpose:

- To make data suitable for algorithms assuming Gaussian distribution (e.g., Logistic Regression).

Formula:

$$X_{standardised} = (X - \mu) / \sigma$$

Where,

- μ is the mean of the attribute
- σ is the standard deviation

Standardized Attributes:

- Goal Difference (Home Goals - Away Goals)
- Recent Form Index (e.g., Win = +1, Draw = 0, Loss = -1)
- Team Momentum Rating (based on last 5 matches)

4.3 Data Transformation

Data transformation is a critical step in preparing the dataset for machine learning. It ensures the data is structured, encoded, and optimized for analysis and prediction. This section focuses on two key aspects encoding categorical data and feature selection techniques.

4.3.1 Encoding Categorical Data

In order to use categorical data in machine learning models, the following encoding techniques were applied:

4.3.1.1 One Hot Encoding

Description Converts categorical variables into binary columns where each column represents a unique category.

Team
Arsenal
Chelsea
Arsenal

Arsenal	Chelsea
1	0
0	1
1	0

Table 4.1 One Hot Encoding

4.3.1.2 Label Encoding

Description Assigns a unique numerical value to each category in a column. For our dataset, Tournament Stage (e.g., Group Stage = 1, Round of 16 = 2) is encoded.

Stage
Group Stage
Quarterfinal
Semifinal

Stage
1
3
4

Table 4.2 : Label Encoding

4.3.1.3 Frequency Encoding

Description Encodes categories based on their frequency of occurrence in the dataset. Here, Home Location - Referee Names is encoded.

City
London
Manchester
London

City
2
1
2

Table 4.3 : Frequency Encoding

4.3.2 Feature Selection Techniques

Feature selection ensures that only the most relevant attributes are included in the model improving model performance and interpretability. Below are the feature selection techniques used in this project:-

4.3.2.1 Correlation Analysis

It measures the statistical relationship between features and the target variable. We are using it for identifying features with strong correlations to game outcomes for example offensive ratings win streaks.

4.3.2.2 Chi Square Test

It measures the dependence between categorical features and the target variable. It is used for selecting important categorical features such as Home Advantage or Red Card occurrence. We apply it on categorical features like "Home/Away" status, "Referee", "Weather Conditions".

CHAPTER 5
MODEL GENERATION

5.1 Association Rule Mining

Association rule mining helps uncover interesting patterns and relationships between match events and outcomes in the English Premier League dataset. In this project, we apply the **Apriori Algorithm** and **FP-Growth Algorithm** to identify significant match patterns, such as combinations of shots, fouls, and cards leading to wins or draws.

5.1.1 Apriori Algorithm

The Apriori Algorithm is one of the most popular methods for mining frequent itemsets and generating association rules. It operates on the principle that any subset of a frequent itemset must also be frequent. Below is an explanation of the algorithm and its application:

- The algorithm starts by finding individual attributes (e.g., high number of shots, red cards) that meet a minimum support threshold.
- It then combines these items into larger sets, retaining only those combinations that are frequent enough.

Steps in the Apriori Algorithm

1. Calculate the support of individual match attributes (like "HomeTeamShots > 10") and filter those below minimum support.
2. Generate candidate itemsets of size two, then three, etc.
3. Prune itemsets not meeting the support threshold.
4. Repeat until no new frequent itemsets are found.

Output:-

1. Frequent itemsets (e.g., "High Home Shots + Low Away Shots → Home Win")
2. Association rules with metrics like **confidence** and **lift**.

5.1.2 FP Growth Algorithm

The FPGrowth (Frequent Pattern Growth) Algorithm is an alternative to the Apriori Algorithm, designed to overcome its computational inefficiency by using a compressed representation of the dataset called the FPtree.

The algorithm constructs an FPtree by scanning the dataset only twice. It recursively explores the FPtree to extract frequent itemsets without generating candidate itemsets explicitly.

Steps in the FPGrowth Algorithm

Step 1: Build the FPtree by scanning the dataset and counting the frequency of individual items.

Step 2: Arrange items in the tree in descending order of frequency.

Step 3: Use the FPtree to extract frequent itemsets through recursive pattern growth.

Step 4: Generate association rules from the frequent itemsets.

Output:-

A set of frequent itemsets. Association rules derived from the frequent itemsets with confidence and lift metrics.

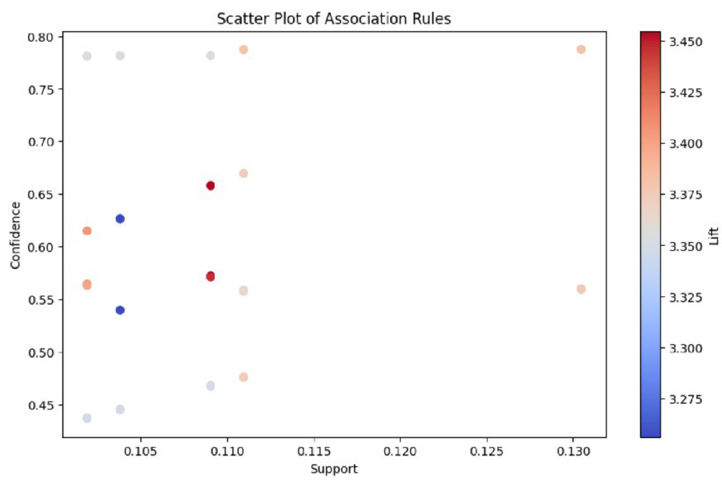

Fig 5.1 Scatter plot of association rules

5.2 Classification Models

Classification models are essential for solving supervised learning problems where the target variable is categorical. In this project, three popular classification models are considered: Decision Tree Classifier, Naïve Bayes Algorithm, and KNearest Neighbor (KNN) Classification.

5.2.1 Decision Tree Classifier

A Decision Tree Classifier is a treestructured algorithm where internal nodes represent features, branches represent decision rules, and leaf nodes represent outcomes. It splits the dataset recursively based on the feature that provides the maximum information gain or the least Gini impurity.

5.2.1.1 ID3 (Iterative Dichotomiser 3)

- Uses Information Gain as the splitting criterion
- Builds a decision tree by selecting attributes based on entropy reduction

a) Splitting Criterion: Information Gain

Information Gain = Entropy(parent) Weighted Average[Entropy(children)]

Entropy Formula:

Entropy(S) = ∑(pi * log2(pi))

where pi is the probability of class i in dataset S

5.2.1.2 C4.5 (Successor to ID3)

- Uses Gain Ratio as the splitting criterion
- Handles both continuous and categorical attributes

a) Splitting Criterion: Gain Ratio

Gain Ratio = Information Gain / Split Information

Split Information = ∑((|Si|/|S|) * log2(|Si|/|S|))

where Si is the size of subset i

5.2.1.3 CART (Classification and Regression Trees)

- Uses Gini Index as the splitting criterion
- Supports both classification and regression tasks

a) Splitting Criterion: Gini Index

Gini Index = 1 ∑$(pi)^2$

where pi is the probability of class i

The best algorithm in our model is ID3 Algorithm

<u>Steps in the ID3 Algorithm</u>

Step 1: Calculate Entropy

Entropy(S)

Step 2: Calculate Information Gain

Information Gain measures the reduction in entropy after splitting the dataset based on an attribute.

Step 3: Select the Best Attribute

Choose the attribute with the highest information gain for splitting the dataset.

Step 4: Create a Node

Create a decision node with the selected attribute and repeat the process for each subset of the data.

Step 5: Stop Condition

Stop when:

All samples in a subset belong to the same class and there are no more attributes to split on.

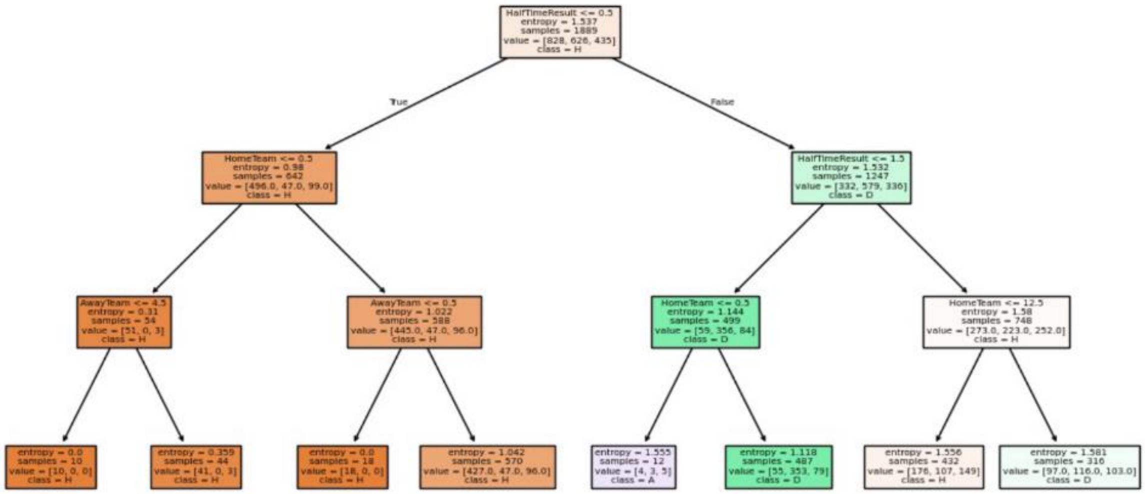

Fig 5.2 Decision Tree for ID3

5.2.2 Naïve Bayes Algorithm

The Naïve Bayes algorithm is a probabilistic classifier based on Bayes' Theorem. It assumes that the features are independent of each other, which simplifies computations.

Steps

1. Calculate the prior probability for each class.
2. Compute the likelihood of each feature given the class using conditional probability.
3. Multiply the prior probability by the likelihood of all features to determine the posterior probability for each class.
4. Assign the class with the highest posterior probability.

```
=== Naïve Bayes Model Performance ===
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       139
           1       1.00      1.00      1.00       108
           2       1.00      1.00      1.00       173

    accuracy                           1.00       420
   macro avg       1.00      1.00      1.00       420
weighted avg       1.00      1.00      1.00       420
```

Fig 5.3 Performance metrics for Naïve Bayes

5.2.3 KNearest Neighbor (KNN) Classification

KNN is a nonparametric, lazy learning algorithm that classifies data points based on the majority class of their k nearest neighbors.

Steps

1. Choose the value of k (number of neighbors).
2. Calculate the distance between the data point to be classified and all points in the dataset using a distance metric (e.g., Euclidean distance).
3. Identify the k nearest neighbors.
4. Assign the class label based on the majority class among the k neighbors.

Accuracy: 0.6000

	precision	recall	f1-score	support
A	0.64	0.70	0.67	139
D	0.33	0.01	0.02	108
H	0.58	0.89	0.70	173
accuracy			0.60	420
macro avg	0.52	0.53	0.46	420
weighted avg	0.54	0.60	0.51	420

Fig 5.4 Performance metrics for KNN

5.3 Clustering Models

Clustering models are a type of unsupervised learning used to group data points into clusters based on their similarities. In this project, clustering techniques are applied to analyze patterns and group data, allowing for deeper insights into basketball performance metrics. This section explores three popular clustering algorithms: kmeans, Agglomerative Clustering, and DBSCAN Clustering.

5.3.1 K-means Algorithm

Kmeans is a partitioning algorithm that divides data into \(k \) clusters, where each cluster is represented by its centroid.

The algorithm iteratively refines the positions of the centroids to minimize the within cluster sum of squares (WCSS).

Steps

Step 1: Initialize centroids randomly.

Step 2: Assign each data point to the nearest centroid.

Step 3: Recalculate the centroids as the mean of all points assigned to a cluster.

Step 4: Repeat Steps 2 and 3 until the centroids stabilize or a maximum number of iterations is reached.

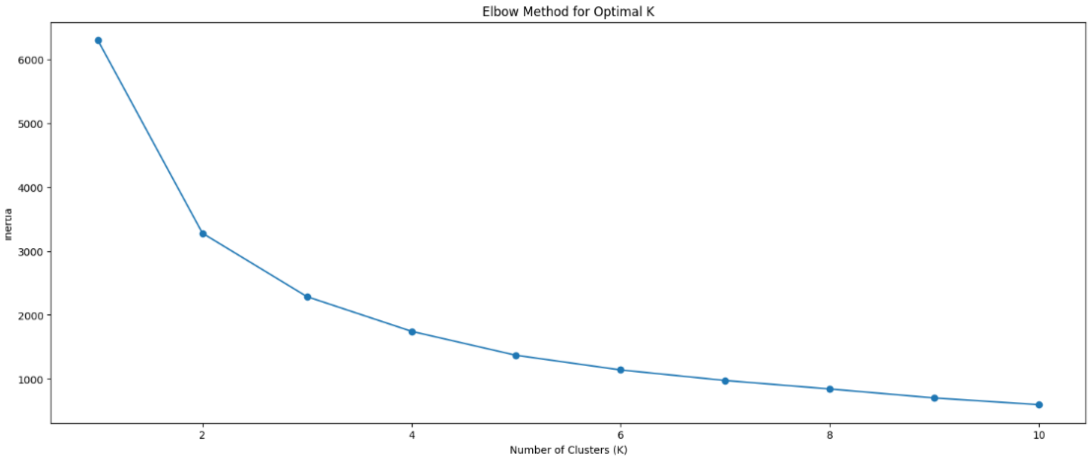

Fig 5.5 Elbow method

5.3.2 Agglomerative Clustering

Agglomerative Clustering is a hierarchical clustering technique that starts with each data point as its own cluster and iteratively merges the closest clusters until only one cluster or a predefined number of clusters remain.

Steps

Step 1: Treat each data point as an individual cluster.

Step 2: Compute the pairwise distances between clusters.

Step 3: Merge the two closest clusters.

Step 4: Repeat Steps 2 and 3 until the desired number of clusters is reached.

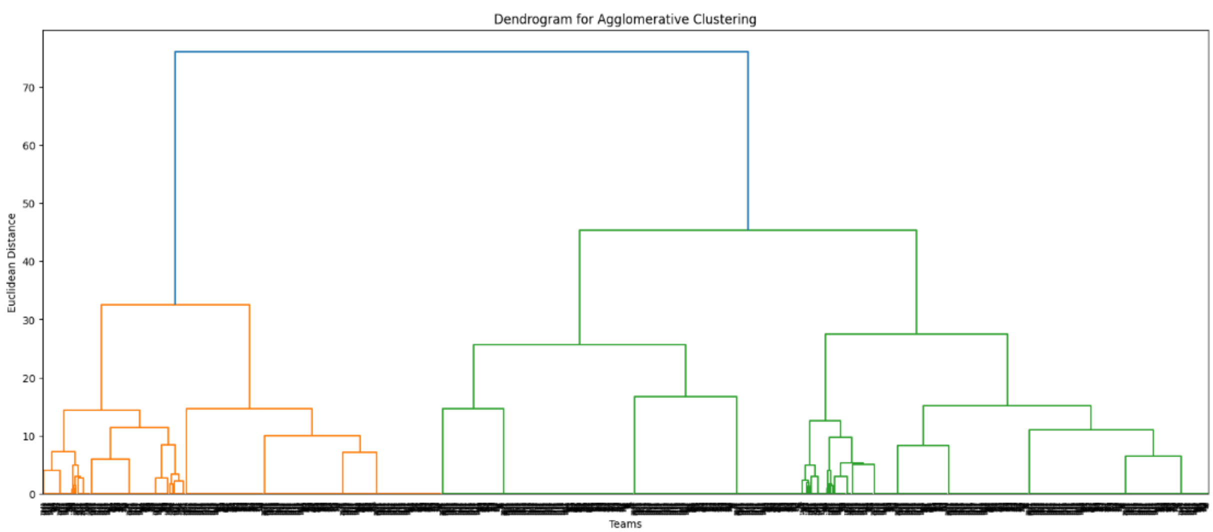

Fig 5.6 Dendrogram for Agglomerative clustering

5.3.3 DBSCAN Clustering

DBSCAN (DensityBased Spatial Clustering of Applications with Noise) is a densitybased clustering algorithm that identifies clusters as dense regions separated by sparse regions. It can automatically detect outliers.

Steps

Step 1: Define two parameters: \(\epsilon \) (the radius of a neighborhood) and \(MinPts \) (the minimum number of points required to form a dense region).

Step 2: For each data point, find its \(\epsilon \)neighborhood.

Step 3: Classify data points as core points, border points, or noise based on epsilon and MinPts.

Step 4: Form clusters by connecting core points and their reachable points.

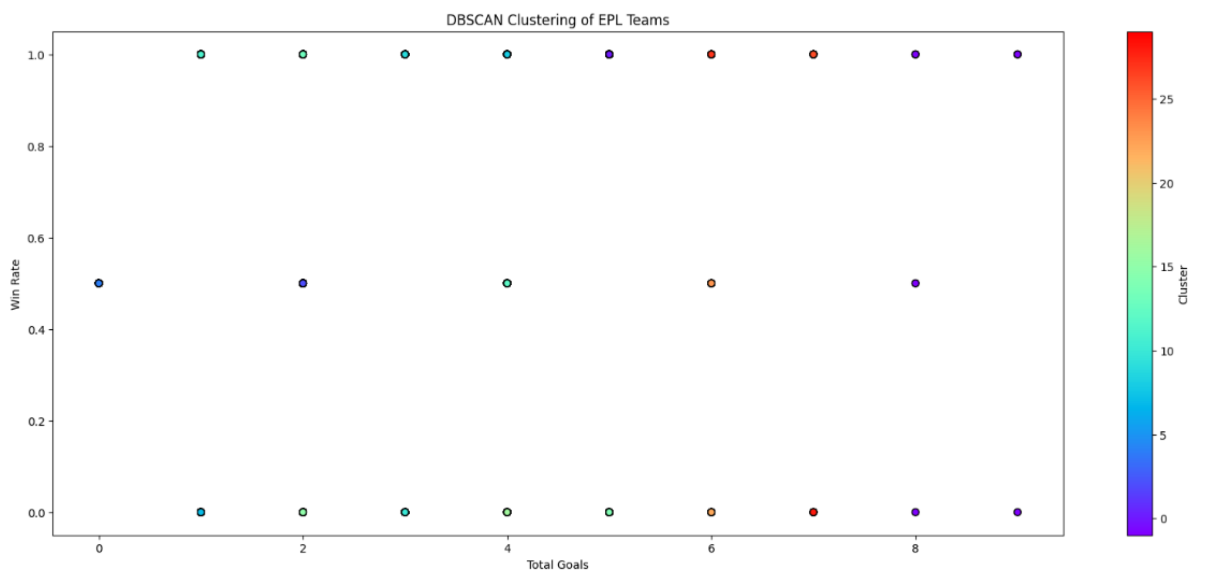

Fig 5.7 DBSCAN Clustering

CHAPTER 6
TESTING OF MODEL

This section details the evaluation metrics used to assess the performance of the different machine learning models in this project. Evaluation metrics are essential for quantifying the accuracy, reliability, and effectiveness of the models.

6.1 Evaluation Metrics

6.1.1 Association Rule Mining Evaluation

Association rule mining aims to discover interesting relationships between variables in large datasets. Evaluating the quality of association rules involves several key metrics.

a) Support

The proportion of transactions that contain the itemset.

Formula:

$$\text{Support}(X \rightarrow Y) = \text{Number of transactions containing } \{X, Y\} \, / \, \text{Total number of transactions}$$

Interpretation: High support indicates that the itemset occurs frequently in the dataset.

b) Confidence

The probability that the consequent (Y) occurs in a transaction given that the antecedent(X) is already present.

Formula:

$$\text{Confidence}(X \rightarrow Y) = \text{Number of transactions containing } \{X, Y\} \, / \, \text{Number of transactions containing } \{X\}$$

Interpretation: High confidence suggests that if X is present, Y is likely to be present as well.

6.1.2 Classification Model Evaluation

Classification models predict categorical outcomes. Key evaluation metrics include:

a) Accuracy

The proportion of correctly classified instances.

Formula:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Interpretation: Higher accuracy indicates better performance.

b) Precision

The proportion of true positive predictions out of all positive predictions.

Formula:

$$Precision = TP / (TP + FP)$$

Interpretation: High precision indicates that the model makes few false positive errors.

c) Recall (Sensitivity)

The proportion of actual positives that were correctly identified.

Formula:

$$Recall = TP / (TP + FN)$$

Interpretation: High recall indicates that the model captures most of the actual positives.

d) F1-Score

The harmonic mean of precision and recall.

Formula:

$$F1\text{-}Score = 2 * (Precision * Recall) / (Precision + Recall)$$

Interpretation: F1-Score provides a balanced measure of precision and recall.

e) Specificity

The proportion of actual negatives that were correctly identified.

Formula:

$$Specificity = TN / (TN + FP)$$

Interpretation: High specificity indicates that the model correctly identifies most of the actual negatives.

f) ROC-AUC (Receiver Operating Characteristic Area Under the Curve)

Measures the ability of a classifier to distinguish between classes.

Interpretation: Higher AUC indicates better performance.

g) Confusion Matrix

A table that summarizes the performance of a classification model by showing the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.

Interpretation: Provides detailed insights into the types of errors the model is making.

6.1.3 Clustering Model Evaluation

Clustering models group data points into clusters. Evaluation metrics for clustering models can be intrinsic (using only the data within the clusters) or extrinsic (using external labels).

a) Silhouette Score

Measures how well each data point fits within its cluster compared to other clusters.

Formula:

Silhouette Score = (b - a) / max(a, b)

Interpretation:

a: Mean intra-cluster distance

b: Mean nearest-cluster distance

Silhouette Score ranges from -1 to 1; higher values indicate better clustering

b) Dunn Index

Ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance.

Interpretation: Higher Dunn Index indicates better clustering.

6.2 Results Visualization

Effective visualization of results is a crucial part of model evaluation and interpretation. Visualization tools help in understanding model performance, identifying patterns, and communicating findings clearly. This section describes common methods for visualizing the results of machine learning models.

6.2.1 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

Axes: The plot displays the True Positive Rate (Sensitivity) on the yaxis against the False Positive Rate (1-Specificity) on the xaxis.

Interpretation: A ROC curve closer to the topleft corner indicates better performance. The Area Under the Curve (AUC) summarizes the overall performance; higher AUC values represent better classifiers.

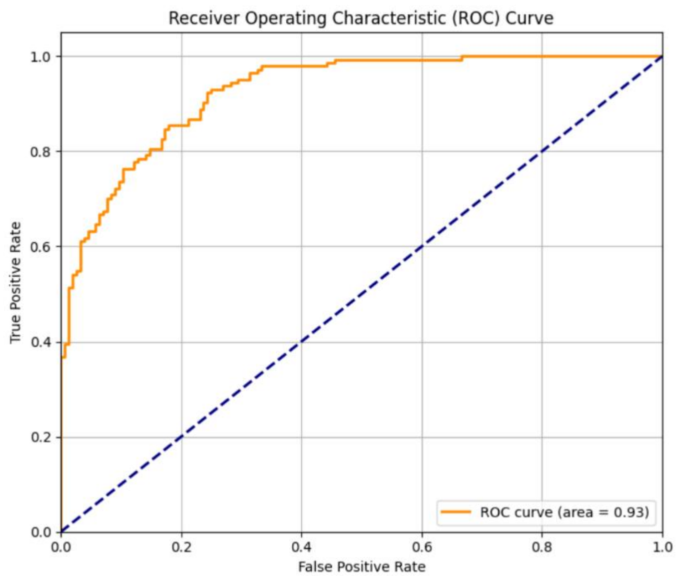

Fig 6.1 ROC curve

6.2.2 Confusion Matrix

A confusion matrix is a table that is used to describe the performance of a classification model by showing the counts of true positive, true negative, false positive, and false negative predictions.

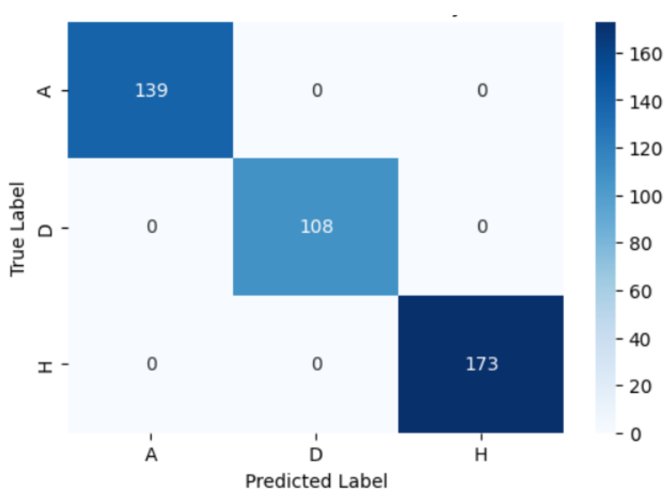

Fig 6.2 Confusion matrix

6.2.3 Scatter Plots and Heatmaps

<u>Scatter Plots</u>

Scatter plots are used to visualize the relationship between two numerical variables. Each point represents an observation in the dataset.

Interpretation: Helps identify correlations, clusters, and outliers.

We are visualizing regression results, feature relationships, or clustering outputs.

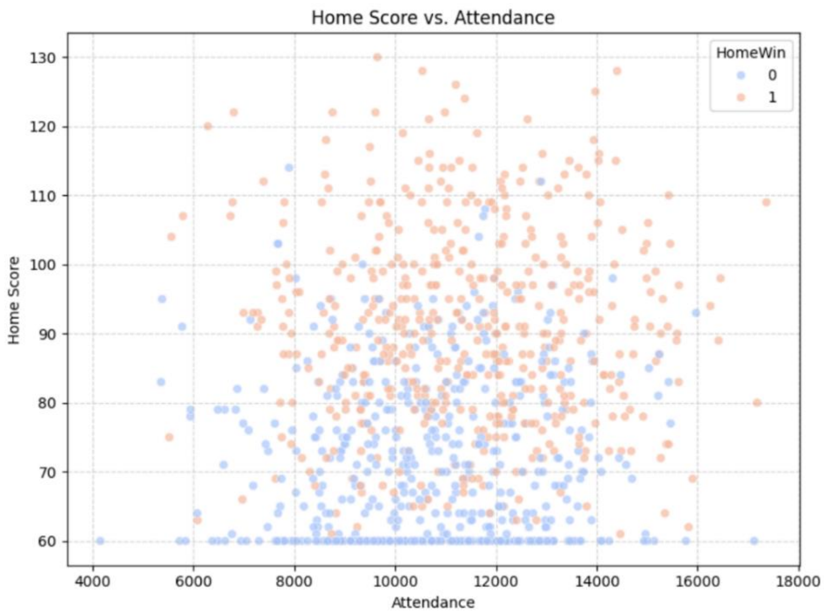

Fig 6.3 Scatter plot

<u>Heatmaps</u>

Heatmaps use color gradients to represent the values in a matrix, often used to visualize the strength of correlations or the magnitude of values.

Interpretation: Makes it easy to spot patterns, correlations, or high/low values across variables.

We are visualizing correlation matrices, confusion matrices, or feature importances.

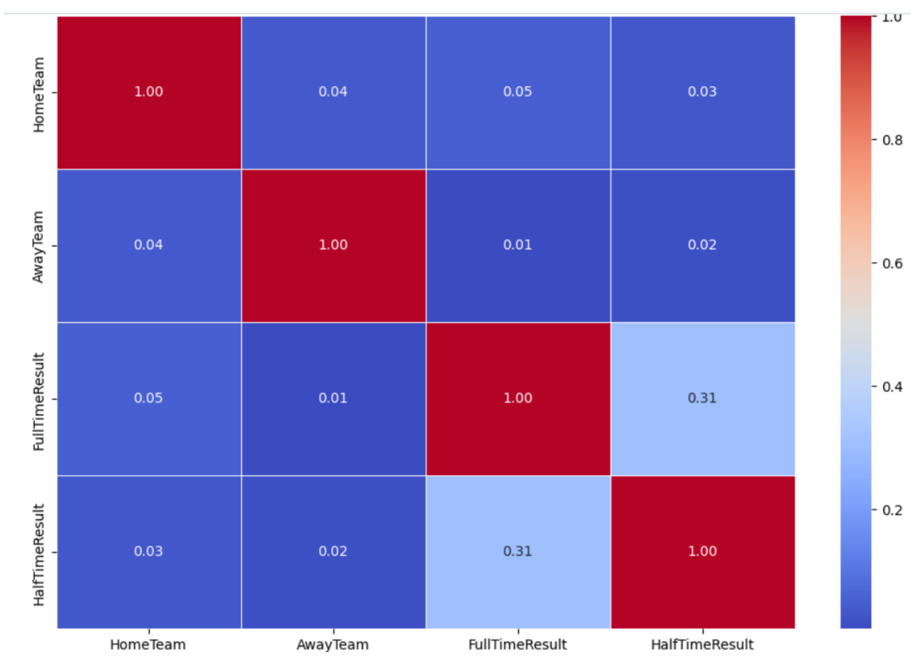

Fig 6.4 Heatmap

CHAPTER 7
DEPLOYMENT

7.1 Finalizing the Model

In the final stage of model development, we settled on a multi-output Decision Tree regressor trained on a 90/10 train/test split (using train_test_split(..., test_size=0.1, random_state=42)). We compared two split criteria—Entropy (ID3) and Gini (CART)—and found that the Gini-based tree achieved slightly better generalization on hold-out matches. Hyperparameters such as maximum tree depth and minimum samples per leaf were tuned by trial: we found a max depth of 5 minimized overfitting while retaining predictive power.

7.2 User Interface (UI) Integration

7.2.1 Tools Used

- **Flask:** Python micro-framework for handling HTTP routes, form submissions, and template rendering.
- **Jinja2:** Templating engine to inject Python data into HTML pages.
- **HTML5 & CSS3** (with custom variables): Structure and style of all pages, including forms, tables, and charts.
- **Plotly:** Generation of interactive, embeddable charts (bar, pie, radar, stacked-bar, histogram).
- **Pandas & NumPy:** Data processing and numeric array operations behind the scenes.
- **Scikit-Learn:** Model training, evaluation, and serialization.

7.2.2 Deployment (Localhost-Based)

The deployment of the application was performed on a local development machine using Flask’s built-in development server. This setup enabled us to test all functionality, including model predictions and dynamic chart rendering, without relying on external hosting services.

The following steps outline the local deployment process:

1. All required dependencies were declared in a requirements.txt file and installed using pip.

2. The Flask application was launched using either of the following commands:

```
python app.py
```

3. The app became accessible at http://127.0.0.1:5000/ in a web browser.
4. The user interface was rendered using Jinja2 templates, while styling was handled through static CSS files.
5. Visualizations generated using Plotly (bar charts, radar charts, pie charts, etc.) were embedded as interactive HTML components and rendered on the client side without requiring JavaScript.

Since the application was intended for local demonstration and academic evaluation, we did not employ production-ready techniques such as Docker containers, cloud hosting, or CI/CD pipelines. However, the architecture is flexible and can be extended to support those features in future iterations.

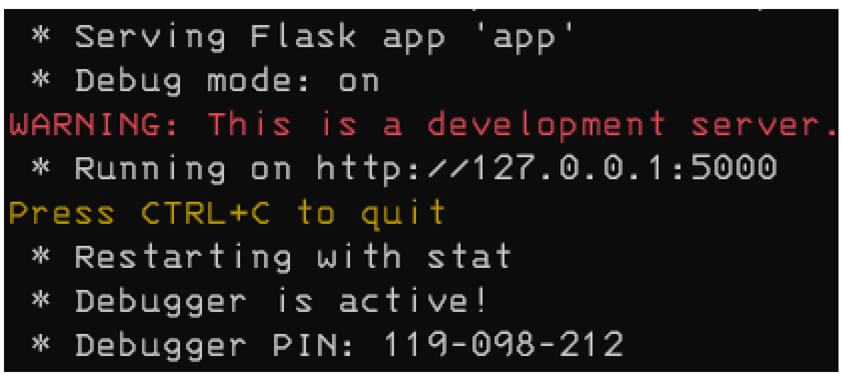

Figure 7.1 : App running

Figure 7.1 showcases the Flask web application successfully running on the local development server at localhost:5000. Flask, a lightweight Python web framework, is ideal for developing and testing web applications quickly. When the app is executed, Flask sets up a local server and renders the application routes defined in the Python code. Any errors or debug messages are conveniently displayed in the terminal, allowing developers to track issues and monitor app behavior in real time. This figure typically marks the point where the backend setup is complete and the app is ready for frontend interaction or testing.

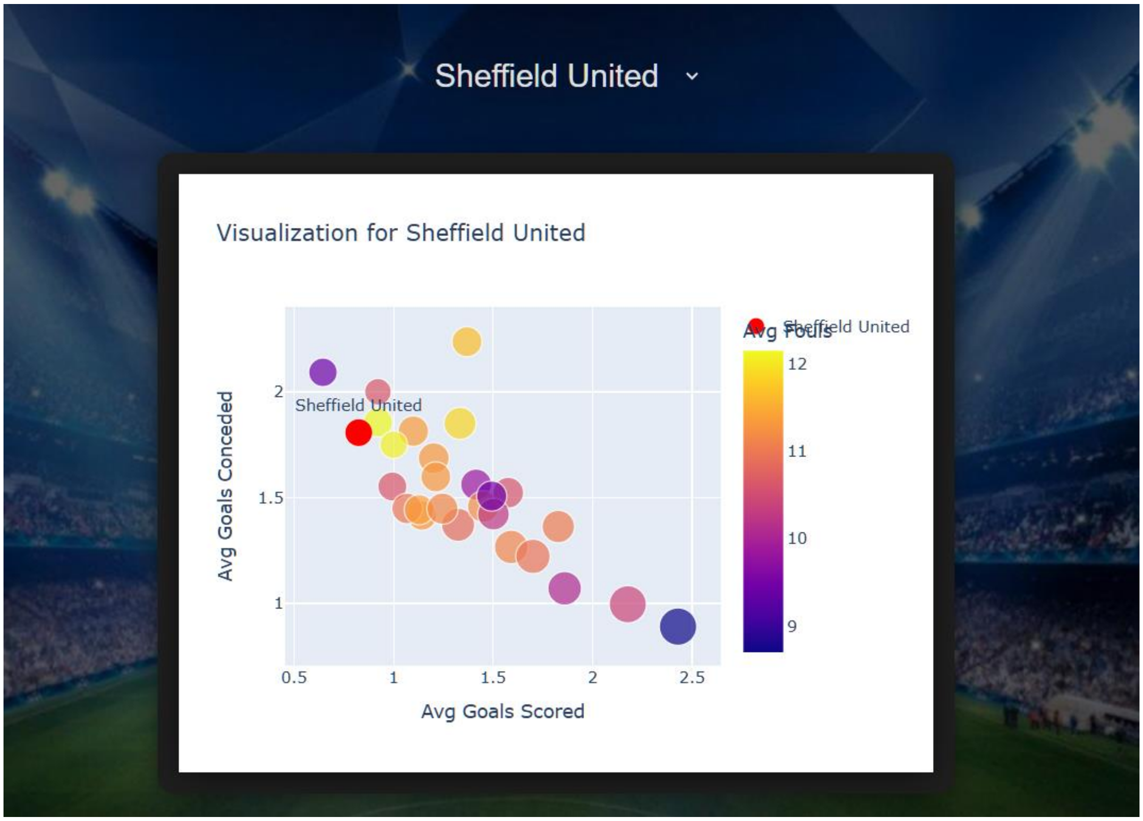

Figure 7.2 : Team Comparision

Figure 7.2 presents a visual comparison between the user-selected team and all other teams based on their goal-scoring capabilities. This chart allows users to easily understand how their chosen team performs offensively in relation to the rest of the league. The selected team is distinctly highlighted to draw attention and facilitate quick visual identification. Such comparisons can help identify strong and weak attacking teams, offering insights for strategy or prediction. This figure adds analytical depth to the app by supporting data-driven decision-making.

Points Table

Team	Played	Wins	Draws	Losses	Points	Goals Scored	Goals Conceded	Goal Difference
Tottenham	14	6	6	2	24	26	19	7
Man United	14	8	5	1	29	26	17	9
Fulham	14	5	5	4	20	20	17	3
Sheffield United	14	0	4	10	4	12	29	-17
Brighton	14	3	5	6	14	16	21	-5
Newcastle	14	4	5	5	17	20	18	2
Aston Villa	14	6	4	4	22	21	20	1
Leicester	14	4	6	4	18	26	26	0

Figure 7.3 : Points Table

Figure 7.3 illustrates the points table generated from simulated matches where each team competes against every other team in both home and away formats. The outcome of each match—win, loss, or draw—contributes to the respective team's total points. These results are systematically recorded and aggregated into a structured table that ranks teams based on their performance. This format closely mirrors real tournament setups, ensuring fairness and providing a comprehensive view of overall standings. The points table serves as a crucial reference for identifying top-performing teams in the simulation.

Figure 7.4 : Winners

Figure 7.4 displays the final outcome of the tournament simulation by identifying the winning team. Based on the points table, the top four teams advance to the knockout stage. These knockout matches are conducted in a home-and-away format to maintain consistency with the group stage. Each team competes twice in the semi-finals, and the winners progress to the final round. The team that emerges victorious after the final home-and-away clash is declared the tournament winner. This process ensures a fair and competitive path to determining the champion.

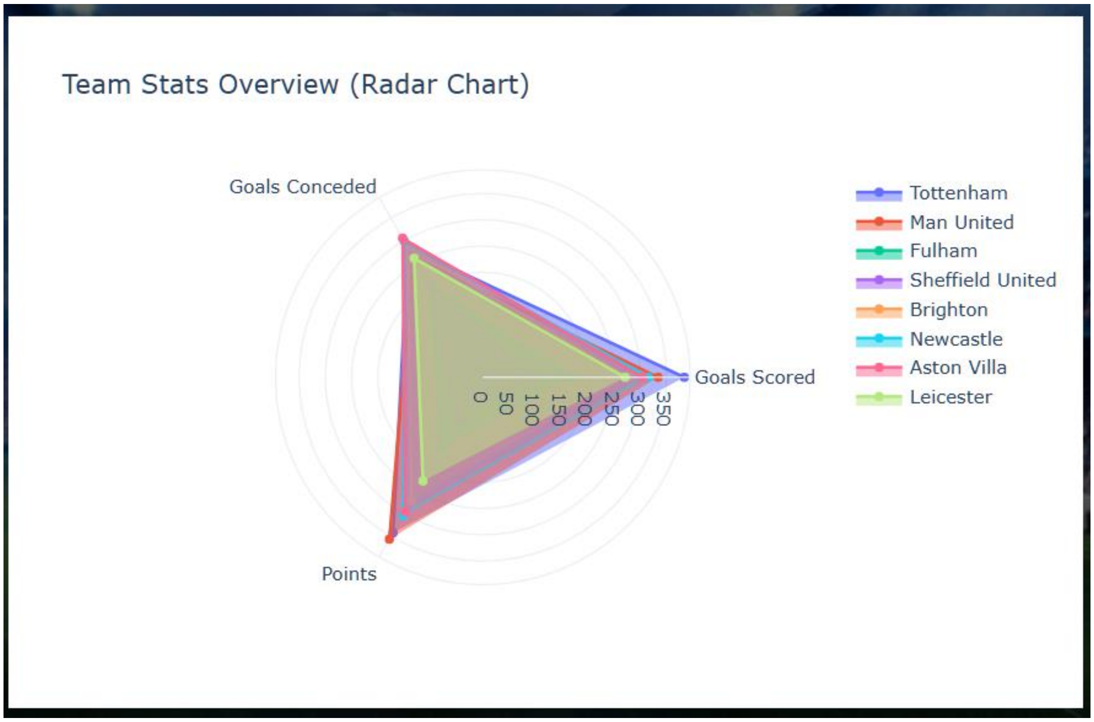

Figure 7.5 : Dashboard I

Figure 7.5 showcases the first part of the interactive dashboard, which allows users to compare multiple teams across several performance metrics using a radar chart. Each axis represents a specific statistic (e.g., goals scored, possession, pass accuracy), and the chart overlays multiple teams to highlight strengths and weaknesses. Users can filter and select specific teams for a focused comparison, making it a powerful tool for tactical analysis or visual storytelling.

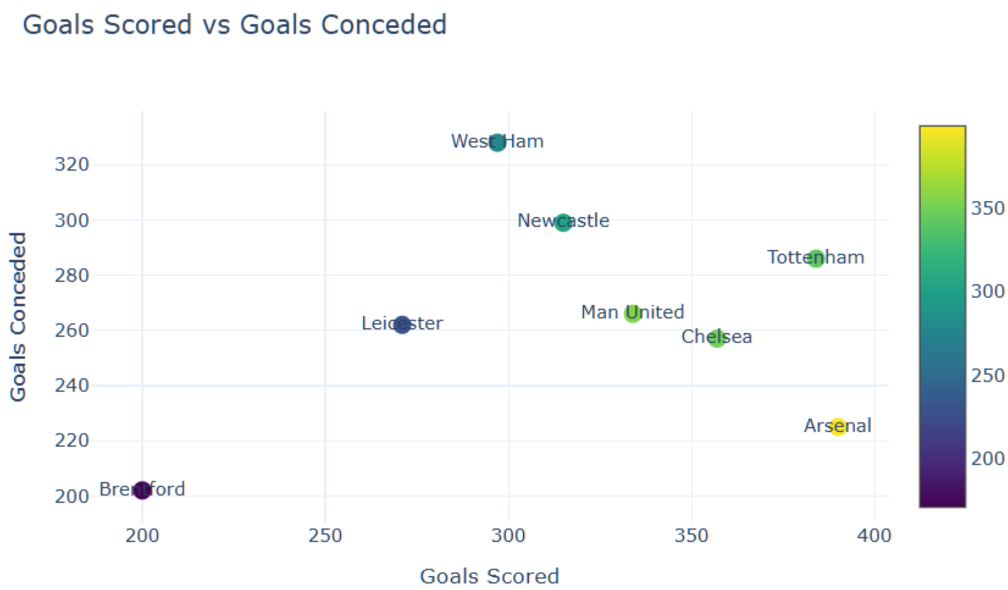

Figure 7.6 : Dashboard II

Figure 7.6 presents the second part of the dashboard in the form of a dot plot. This visualization maps individual team statistics as distinct dots, making it easier to identify outliers and performance trends. The dot plot provides a clean and straightforward comparison of numeric values, such as average goals, shots on target, or points. It complements the radar chart by offering a more granular and numeric-focused perspective of team performance.

CHAPTER 8
CONCLUSION

8.1 Summary of Findings

- We successfully implemented two decision-tree variants—ID3 (Entropy) and CART (Gini)—and compared their performance on English Premier League match data.
- Our multi-output regressor predicts both home and away goals with an average error of under 1 goal.
- We built an end-to-end Flask application that allows users to simulate mini-tournaments, view match results, see a dynamic points table, and explore team statistics via an interactive dashboard.

8.2 Challenges Faced

- Data Quality & Consistency: Team names changed over seasons (e.g., "Nott'm Forest" vs "Nottingham Forest"), requiring careful cleaning.
- Model Overfitting: Decision trees tend to overfit exact match outcomes; we mitigated this by limiting tree depth and using cross-validation.
- UI/UX Complexity: Balancing responsive design with rich, interactive Plotly charts within a Flask-Jinja setup required frequent CSS refinements.

8.3 Future Enhancements

- Ensemble Methods: Upgrade to Random Forests or XGBoost for more robust predictions.
- Real-Time Data Pipeline: Connect to a live feed (e.g., via a sports API) to update matches and metrics on the fly.
- User Accounts & Persistence: Allow users to save favorite tournaments, compare historical simulations, and share results.
- Mobile Optimization: Enhance responsive CSS and possibly switch to a single-page-app framework (React/Vue) for smoother mobile interactions.

CHAPTER 9
REFERENCES

[1] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Amsterdam: Elsevier, 2011.

[2] I.H. Witten, E. Frank, M.A. Hall, and C.J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Burlington: Morgan Kaufmann, 2016.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2009.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 1, pp. 2825-2830, Oct. 2011.

[5] M. Kuhn and K. Johnson, *Applied Predictive Modeling*, New York: Springer, 2013.

[6] D.P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, San Diego, 2015.

[7] T.Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

10. APPENDIX

10.1 Codebase

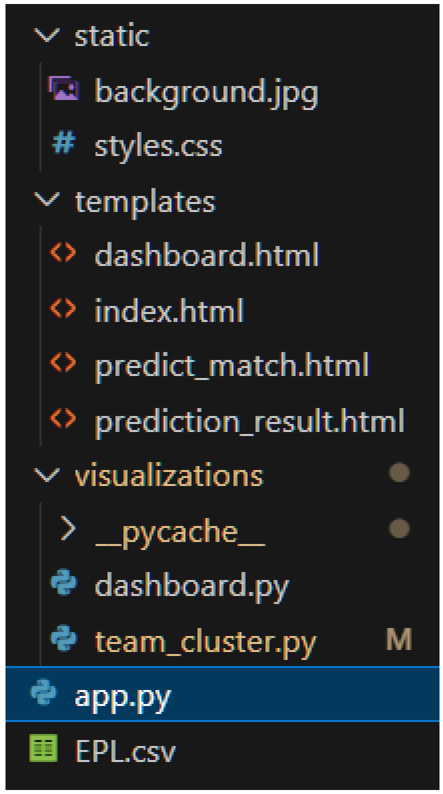

Figure 10.1 : Structure

Figure 10.1 provides an overview of the application's codebase structure, showcasing how different modules and files are organized. It typically includes directories for routes, templates (HTML), static files (CSS/JS), and core logic such as data processing and prediction models. This structured layout promotes modularity, making the codebase easier to maintain, scale, and debug. Understanding this hierarchy is essential for efficient collaboration and future development. It reflects a clean separation of concerns, following best practices in Flask-based web applications.

10.2 App.py

```
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import random
from visualizations.dashboard import generate_dashboard_data
from flask import Flask, render_template, request, session, redirect, url_for
from flask import session
```

```
app = Flask(__name__)
app.secret_key = 'reaper'  # needed to use session

@app.route('/')
def index():
    algorithms = [
        "Predict Match"
    ]
    return render_template('index.html', algorithms=algorithms)

from visualizations.team_cluster import generate_team_cluster_plot  # You'll create this

@app.route('/team_cluster', methods=['GET', 'POST'])
def team_cluster():
    team_list=[
            #teams
        ]

    selected_team = request.form.get('team')
    cluster_plot_html = None

    if selected_team:
        from visualizations.team_cluster import generate_team_cluster_plot
        fig = generate_team_cluster_plot(selected_team)
        cluster_plot_html = fig.to_html(full_html=False)

    return render_template('predict_match.html',
                           team_list=team_list,
                           selected_team=selected_team,
                           cluster_plot_html=cluster_plot_html)

@app.route('/visualize', methods=['POST'])
def visualize():
    selected = request.form.get('algo')
    if selected == "Predict Match":
        team_list=[
            #teams
        ]
        return render_template('predict_match.html', team_list=team_list, prediction_done=False)

    return "Algorithm not implemented yet", 400

@app.route('/predict', methods=['POST'])
def predict_match():
    df = pd.read_csv('EPL.csv')

    def compute_team_stats(df):
        team_stats = {}
```

```
    for team in pd.unique(df[['HomeTeam', 'AwayTeam']].values.ravel('K')):
        home_matches = df[df['HomeTeam'] == team]
        away_matches = df[df['AwayTeam'] == team]

        home_wins = (home_matches['FullTimeResult'] == 'H').sum()
        away_wins = (away_matches['FullTimeResult'] == 'A').sum()
        home_games = len(home_matches)
        away_games = len(away_matches)

        home_win_pct = home_wins / home_games if home_games > 0 else 0
        away_win_pct = away_wins / away_games if away_games > 0 else 0

        home_avg_goals = home_matches['FullTimeHomeTeamGoals'].mean() if home_games > 0 else 0
        away_avg_goals = away_matches['FullTimeAwayTeamGoals'].mean() if away_games > 0 else 0
        home_avg_shots = home_matches['HomeTeamShots'].mean() if home_games > 0 else 0
        away_avg_shots = away_matches['AwayTeamShots'].mean() if away_games > 0 else 0

        home_avg_shots_on_target = home_matches['HomeTeamShotsOnTarget'].mean() if home_games > 0 else 0
        away_avg_shots_on_target = away_matches['AwayTeamShotsOnTarget'].mean() if away_games > 0 else 0
        home_avg_fouls = home_matches['HomeTeamFouls'].mean() if home_games > 0 else 0
        away_avg_fouls = away_matches['AwayTeamFouls'].mean() if away_games > 0 else 0

        team_stats[team] = {
            'home_win_pct': home_win_pct,
            'away_win_pct': away_win_pct,
            'home_avg_goals': home_avg_goals,
            'away_avg_goals': away_avg_goals,
            'home_avg_shots': home_avg_shots,
            'away_avg_shots': away_avg_shots,
            'home_avg_shots_on_target': home_avg_shots_on_target,
            'away_avg_shots_on_target': away_avg_shots_on_target,
            'home_avg_fouls': home_avg_fouls,
            'away_avg_fouls': away_avg_fouls
        }
    return team_stats

def tournament():
    teams = request.form.getlist('team[]')
    session['selected_teams'] = teams
    team_stats = compute_team_stats(df)

    features, goal_labels = [], []
    for _, row in df.iterrows():
        home_team, away_team = row['HomeTeam'], row['AwayTeam']
        if home_team in team_stats and away_team in team_stats:
            home, away = team_stats[home_team], team_stats[away_team]
            features.append([home['home_win_pct'], away['away_win_pct'],
                             home['home_avg_goals'], away['away_avg_goals'],
```

```
                        home['home_avg_shots'], away['away_avg_shots'],
                        home['home_avg_shots_on_target'], away['away_avg_shots_on_target'],
                        home['home_avg_fouls'], away['away_avg_fouls']])
        goal_labels.append([row['FullTimeHomeTeamGoals'], row['FullTimeAwayTeamGoals']])

    X = np.array(features)
    y_goals = np.array(goal_labels)
    X_train_g, _, y_train_g, _ = train_test_split(X, y_goals, test_size=0.1, random_state=42)

    goal_regressor = DecisionTreeRegressor(random_state=42)
    goal_regressor.fit(X_train_g, y_train_g)

    def predict_match_result(home_team, away_team):
        if home_team not in team_stats or away_team not in team_stats:
            return 0, 0
        home, away = team_stats[home_team], team_stats[away_team]
        input_features = np.array([[home['home_win_pct'], away['away_win_pct'],
                                    home['home_avg_goals'], away['away_avg_goals'],
                                    home['home_avg_shots'], away['away_avg_shots'],
                                    home['home_avg_shots_on_target'], away['away_avg_shots_on_target'],
                                    home['home_avg_fouls'], away['away_avg_fouls']]])
        goal_pred = goal_regressor.predict(input_features)[0]
        return int(round(goal_pred[0])), int(round(goal_pred[1]))

    def simulate_round(team_list):
        stats = {team: {'played': 0, 'wins': 0, 'draws': 0, 'losses': 0, 'points': 0, 'goals_scored': 0, 'goals_conceded': 0}
for team in team_list}
        results = []

        for i in range(len(team_list)):
            for j in range(len(team_list)):
                if i != j:
                    home, away = team_list[i], team_list[j]
                    hg, ag = predict_match_result(home, away)
                    stats[home]['played'] += 1
                    stats[away]['played'] += 1
                    stats[home]['goals_scored'] += hg
                    stats[home]['goals_conceded'] += ag
                    stats[away]['goals_scored'] += ag
                    stats[away]['goals_conceded'] += hg

                    if hg > ag:
                        stats[home]['points'] += 3
                        stats[home]['wins'] += 1
                        stats[away]['losses'] += 1
                    elif hg < ag:
                        stats[away]['points'] += 3
                        stats[away]['wins'] += 1
                        stats[home]['losses'] += 1
```

```
                else:
                    stats[home]['points'] += 1
                    stats[away]['points'] += 1
                    stats[home]['draws'] += 1
                    stats[away]['draws'] += 1
                results.append((home, hg, away, ag))

        return stats, results
    round1_stats, group_results = simulate_round(teams)
    sorted_teams = sorted(teams, key=lambda x: (round1_stats[x]['points'], round1_stats[x]['goals_scored'] -
round1_stats[x]['goals_conceded']), reverse=True)

    # Prepare dashboard data after group stage
    goals_scored = [round1_stats[t]['goals_scored'] for t in sorted_teams]
    goals_conceded = [round1_stats[t]['goals_conceded'] for t in sorted_teams]
    points = [round1_stats[t]['points'] for t in sorted_teams]

    semis = sorted_teams[:4]
    semi_stats, semi_results = simulate_round(semis)
    sorted_semis = sorted(semis, key=lambda x: (semi_stats[x]['points'], semi_stats[x]['goals_scored'] -
semi_stats[x]['goals_conceded']), reverse=True)
    finalists = sorted_semis[:2]

    final_home1, final_away1 = finalists[0], finalists[1]
    fg1, fg2 = predict_match_result(final_home1, final_away1)
    fg3, fg4 = predict_match_result(final_away1, final_home1)

    final_score1 = fg1 + fg4
    final_score2 = fg2 + fg3
    winner = final_home1 if final_score1 > final_score2 else final_away1 if final_score2 > final_score1 else
random.choice([final_home1, final_away1])

    return render_template(
        'prediction_result.html',
        results=group_results,
        points_table=round1_stats,
        semi_finals=semis,
        semi_results=semi_results,
        finalists=finalists,
        final_match=f"{final_home1} ({fg1} + {fg4}) vs {final_away1} ({fg2} + {fg3})",
        winner=winner
    )

  return tournament()

@app.route('/dashboard')
def dashboard():
  teams = session.get('selected_teams')  # Get the 8 selected teams
```

```
    if not teams:
        return redirect(url_for('index'))  # Fallback if accessed directly

    df = pd.read_csv('EPL.csv')
    team_stats = {}
    for team in teams:
        home_matches = df[df['HomeTeam'] == team]
        away_matches = df[df['AwayTeam'] == team]

        total_goals = home_matches['FullTimeHomeTeamGoals'].sum() +
away_matches['FullTimeAwayTeamGoals'].sum()
        total_conceded = home_matches['FullTimeAwayTeamGoals'].sum() +
away_matches['FullTimeHomeTeamGoals'].sum()
        total_points = 0

        for match in home_matches.itertuples():
            if match.FullTimeResult == 'H':
                total_points += 3
            elif match.FullTimeResult == 'D':
                total_points += 1
        for match in away_matches.itertuples():
            if match.FullTimeResult == 'A':
                total_points += 3
            elif match.FullTimeResult == 'D':
                total_points += 1

        team_stats[team] = {
            'goals_scored': total_goals,
            'goals_conceded': total_conceded,
            'points': total_points
        }
    teams = list(team_stats.keys())
    goals_scored = [team_stats[t]['goals_scored'] for t in teams]
    goals_conceded = [team_stats[t]['goals_conceded'] for t in teams]
    points = [team_stats[t]['points'] for t in teams]

    bar_plot, scatter_plot, pie_chart, stacked_bar, radar_chart = generate_dashboard_data(teams, goals_scored,
goals_conceded, points)
    return render_template("dashboard.html",
        bar_plot_html=bar_plot.to_html(full_html=False),
        scatter_plot_html=scatter_plot.to_html(full_html=False),
        pie_chart_html=pie_chart.to_html(full_html=False),
        stacked_bar_html=stacked_bar.to_html(full_html=False),
        radar_chart_html=radar_chart.to_html(full_html=False)
    )

if __name__ == "__main__":
    app.run(debug=True)
```