Football passing sequences and networks for goal prediction

Diogo Lobo do Souto Ferreira
Master in Computer Science
Department of Computer Science
Faculty of Sciences of the University of Porto
2024

Football passing sequences and networks for goal prediction

Diogo Lobo do Souto Ferreira

Dissertation carried out as part of the Master in Computer Science

Department of Computer Science

2024

Supervisor
Pedro Manuel Pinto Ribeiro, Assistant Professor, Faculty of Sciences, University of Porto

Co-supervisor
Miguel Eduardo Pinto da Silva, Invited Assistant Professor, Faculty of Sciences, University of Porto

Abstract

This thesis explores the use of machine learning models to analyze passing sequences in football and predict whether a sequence will result in a goal. This work applies both traditional Machine Learning models and deep learning methods, such as Recurrent Neural Networks and Graph Netural Networks, to capture the dynamics of team play derived from passing data and predict the binary outcome (goal or no goal) of a sequence of passes. The dataset used in this study consists of passing events from the 2015-2016 season of the top five European football leagues: the Premier League, La Liga, Serie A, Bundesliga and Ligue 1.

An exploratory analysis was conducted to reveal important trends across various features, including pass lengths, angles, player positioning, and contextual factors like game pressure and play patterns. The analysis uncovered distinct characteristics and patterns present in goal-scoring sequences. These features help in identifying the key factors that contribute to successful plays.

Based on the exploratory analysis, we developed a range of machine learning models, including traditional techniques and advanced deep learning approaches, to predict the probability of a given passing sequence resulting in a goal. We hypothesized that this task is especially challenging because it focuses solely on passing attributes, excluding one of the most important actions in football: the shot. Without this key feature, the models must rely solely on passing data, which limits their ability to fully capture the critical moments leading to a goal.

Despite this limitation, our models demonstrated the ability to learn some meaningful relationships between the passing sequence and the likelihood of scoring. In particular, we found that traditional machine learning models, namely Random Forests, were more effective at this task than methodologies based on deep learning (although their performance was close to that of GNNs). This was possibly due to traditional models' reliance on feature engineering, which converts passing sequences into tabular data, making it easier for these models to process and analyze the data compared to deep learning models.

Keywords: Machine Learning, Sports Analytics, Football, Goal Prediction, Passing Sequences

Resumo

Esta tese explora a utilização de modelos de machine learning para analisar sequências de passes no futebol e prever oportunidades de marcar golos. Ao centrar-se nos dados de passes, este trabalho aplica modelos tradicionais e métodos avançados, como as Recurrent Neural Networks (RNNs) e Graph Neural Networks (GNNs), para captar a dinâmica do jogo de equipa. O conjunto de dados utilizado neste estudo consiste em eventos de passe da época de 2015-2016 das cinco principais ligas europeias de futebol: Premier League, La Liga, Serie A, Bundesliga e Ligue 1.

Foi efetuada uma análise exploratória para revelar tendências importantes em várias caraterísticas, incluindo comprimentos de passe, ângulos, posicionamento dos jogadores e fatores contextuais como a pressão e padrões de jogo. A análise revelou caraterísticas e padrões distintos presentes nas sequências de passe que resultam em golos. Estas caraterísticas ajudam a identificar os fatores-chave que contribuem para o êxito das jogadas.

Com base na análise exploratória, desenvolvemos uma série de modelos de machine learning, incluindo técnicas tradicionais e abordagens avançadas, para prever a probabilidade de uma determinada sequência de passes resultar num golo. Partimos da hipótese de que esta tarefa é especialmente difícil porque se concentra apenas nos atributos de passe, excluindo uma das ações mais importantes no futebol: o remate. Sem esta caraterística chave, os modelos devem basear-se apenas nos dados de passe, o que limita a sua capacidade de captar totalmente os momentos críticos que conduzem a um golo.

Apesar desta limitação, os nossos modelos demonstraram a capacidade de captar relações significativas entre as sequências de passes e a probabilidade de resultarem em golo. Especificamente, os modelos tradicionais de machine learning, como as Random Forests, superaram as abordagens baseadas em deep learning (embora o seu desempenho tenha sido semelhante ao das GNNs). Esta diferença deve-se possivelmente ao facto de os modelos tradicionais dependerem de feature engineering, que converte as sequências de passes em dados tabulares, facilitando o processamento e a análise por parte destes modelos em comparação com os modelos de deep learning.

Acknowledgements

I would like to express my deepest gratitude to my advisors, Prof. Pedro Ribeiro and Miguel Silva, for their guidance, support, and encouragement during the development of this thesis.

I would also like to extend my heartfelt thanks to my family and friends for their constant support and understanding. Your belief in me and your encouragement during challenging times has been instrumental in completing this journey. Thank you for always being there.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Football, often called "the beautiful game", captivates millions with its blend of skill, strategy, and unpredictability. At its core are passing sequences: moves that maintain possession, break down defenses, and can lead to a goal. Studying these sequences is not just about appreciating the game's beauty; it is about finding patterns that can help predict key outcomes, especially goals.

In recent years, football has become more than just a sport; it has evolved into a valuable field for data analysis, helping to better understand team strategies and player performance. Due to its global popularity and high stakes, football has become an important focus for analysis. Like other sports, such as baseball with the "Moneyball" approach (use of data analytics to identify undervalued players and strategies), football is now embracing analytics. This shift has helped teams, analysts, and fans gain a clearer understanding of different aspects of the game using data and metrics.

In particular, machine learning and deep learning have emerged as powerful tools within sports analytics, offering new ways to model and predict game events. The increasing influence of companies like OpenAI and Nvidia in advancing these technologies highlights the potential for applying cutting-edge AI techniques to sports. From a financial perspective, the application of AI in sports has opened up new markets and investment opportunities, transforming the industry.

In this thesis, we focus specifically on analyzing passing sequences in football using data from the top five European football leagues—Premier League, La Liga, Serie A, Bundesliga, and Ligue 1—across the 2015-2016 season. A team's possession and passing movements reveal important aspects of their tactics, often shown through passing maps or network analyses. By studying these patterns, we aim to identify key factors that lead to successful goal-scoring opportunities. Following this, we employ machine learning techniques to compare traditional models like Random Forest and Logistic Regression with more advanced architectures like Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs).

While previous studies have explored various aspects of passing behavior and its relationship to match outcomes [14, 22], to the best of our knowledge, no prior work has exclusively focused on using sequences of passes to directly predict whether a sequence leads to a goal. One reason for this could be that passing sequences alone may not provide sufficient information, as critical moments such as the shot are excluded. However, despite this limitation, our hypothesis is that passing sequences can still offer valuable predictive signals, potentially benefiting models like Expected Goals (xG).

1.2 Thesis Structure

This thesis is structured into the following major chapters:

Chapter 02 - Background: This chapter covers the basic concepts and methods needed to analyze football passing sequences and predict goals.

Chapter 03 - Related Work: In this chapter, we look at the current research on football analytics. Specifically, we focus on studies that have examined passing patterns, player movements, and goal prediction.

Chapter 04 - Exploratory Analysis: Here, we carefully explore the dataset to study important patterns and trends in passing sequences. This analysis is the foundation for the predictive modeling work that comes after.

Chapter 05 - Predicting Goals from Passing Events: This chapter describes how the predictive models were developed and evaluated. It covers the methodology, data preprocessing, feature selection, and model performance, with a focus on predicting goals based on passing events.

Chapter 06 - Conclusion: In the final chapter, we will sum up the findings of this thesis, consider the impact on football analytics and propose directions for future research.

Chapter 2

Background

2.1 Machine Learning

Machine Learning is a branch of artificial intelligence dedicated to creating algorithms that allow computers to learn from data and make predictions or decisions accordingly. It has become a fundamental tool in various domains, including finance, healthcare, and technology, by allowing systems to automatically improve and adapt through experience [25].

The core objective of machine learning is to construct models that can generalize from observed data to unseen instances. These models can be broadly categorized into supervised, unsupervised, and reinforcement learning, each addressing different types of learning tasks. Among these, supervised learning is particularly important for classification and regression tasks, where the goal is to predict an output based on input features.

In this section, we will explore some of the most commonly used machine learning algorithms, focusing on their theoretical foundations and practical applications. The algorithms discussed include Logistic Regression, Random Forest, and Support Vector Machine, each of which plays a important role in modern data-driven decision-making processes.

2.2 Classification Algorithms

Classification algorithms are a key component of machine learning and data science, providing methods to categorize data into predefined classes. These algorithms form the foundation for numerous applications that range from image recognition to medical diagnosis.

2.2.1 Logistic Regression

Logistic Regression is a statistical model commonly applied to binary classification tasks, where it estimates the likelihood of a binary outcome using one or more predictor variables. Although

the formulation presented here focuses on the binary case, logistic regression can be extended to handle multiclass classification problems. The logistic function, also known as the sigmoid function, is used to map predicted values to probabilities [18].

The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

The logistic regression model estimates the probability that a given input belongs to a particular class. It is expressed as:

$$P(y = 1|x) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n) \tag{2.2}$$

In this context, y is the binary outcome, x_i represents the predictor variables, and β_i are the coefficients to be estimated from the data. Logistic Regression is widely used due to its simplicity and effectiveness in linear decision boundaries. However, it is limited to linear relationships between the independent variables and the log-odds of the dependent variable [18].

2.2.2 Random Forest

Random Forest is a method of ensemble learning that merges multiple decision trees to enhance classification performance. Each decision tree is trained on a portion of the dataset, and the overall prediction is determined by combining the individual predictions from all the trees, typically through majority voting.

2.2.2.1 Decision Trees

A Decision Tree is a structure resembling a tree, where each internal node signifies a decision made using a feature, each branch indicates the result of that decision, and each leaf node corresponds to a class label or final output. The model recursively partitions the data based on feature thresholds, aiming to minimize classification error or maximize some impurity reduction criteria like *Gini impurity* or *information gain*. In classification tasks, the *Gini impurity* is often used to measure the quality of a split. It calculates the likelihood of incorrectly classifying a randomly chosen instance from the dataset[3].The formula for *Gini impurity* is:

$$Gini(T) = 1 - \sum_{i=1}^{k} p_i^2$$

Where:

- p_i represents the proportion of instances belonging to class i in the dataset T,
- k is the total number of classes.

The decision function for a single decision tree is:

$$h(x) = \sum_{i=1}^{N} w_i I(x \in R_i) \tag{2.3}$$

where w_i represents the weight associated with region R_i, and $I(\cdot)$ is the indicator function.

Decision Trees are known for their simplicity. However, they are prone to overfitting, especially with deep trees. This is where Random Forests come into play, as they mitigate this problem by averaging multiple decision trees, improving generalization [2].

2.2.2.2 Random Forest

The Random Forest model aggregates the predictions of multiple decision trees as:

$$H(x) = \text{majority\_vote}\left(h_1(x), h_2(x), \ldots, h_B(x)\right) \tag{2.4}$$

where $h_b(x)$ is the prediction of the b-th tree, and B is the total number of trees in the forest.

Random Forests offer several advantages:

- They are robust to overfitting, especially when dealing with high-dimensional data.
- They can handle large datasets with higher dimensionality.
- They provide estimates of feature importance, helping in feature selection [2].

The main drawback of Random Forests is their complexity and computational cost, especially with a large number of trees and features.

2.2.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is an effective classification algorithm that identifies the optimal hyperplane to separate data into distinct classes. Its objective is to maximize the margin between the two classes, which is the distance between the hyperplane and the nearest data point of each class.

The decision function for SVM is:

$$f(x) = \text{sign}(w \cdot x + b) \tag{2.5}$$

where w is the weight vector and b is the bias term.

The objective of SVM is to find the hyperplane that maximizes the margin γ, defined as:

$$\gamma = \frac{2}{\|w\|} \tag{2.6}$$

subject to the constraint:

$$y_i(w \cdot x_i + b) \geq 1 \quad \text{for all } i \tag{2.7}$$

SVM can be extended to non-linear boundaries using kernel functions, which map the input features to a higher-dimensional space, making linear separation achievable. Common kernel functions include:

- Linear kernel: $K(x, x') = x \cdot x'$
- Polynomial kernel: $K(x, x') = (x \cdot x' + c)^d$
- Radial Basis Function (RBF) kernel: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$ [26]

SVMs are effective in high-dimensional spaces and are particularly useful for cases where the number of dimensions exceeds the number of samples. However, they can be less efficient on large datasets due to their computational complexity [7].

2.3 Neural Networks, Deep Learning & Graph Learning

2.3.1 Foundations of Neural Networks

The origin of neural network theory dates back to 1943, when Warren McCulloch and Walter Pitts published their groundbreaking paper [23]. This work proposed a simplified model of neural networks based on the neuron, the fundamental unit of the brain. Their model, often referred to as the McCulloch-Pitts neuron, was a binary threshold device that fired (outputted a "1") if the weighted sum of its inputs exceeded a certain threshold, otherwise, it did not fire (outputted a "0"). They demonstrated that neural networks could, in principle, compute any arithmetic or logical function, effectively making them Turing complete. This concept showed that complex brain functions could originate from simple binary operations, sparking initial interest in artificial neural networks.

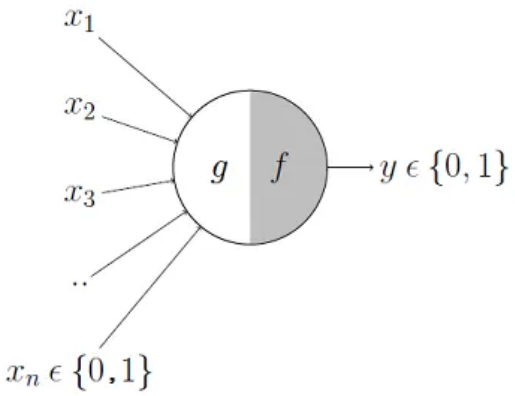


Figure 2.1: McCulloch-Pitts Neuron, source: [32].

In 1958, Frank Rosenblatt, introduced the perceptron [29], a significant advancement in the field of neural networks. The perceptron was inspired by the neurobiological processes of human vision and aimed to create a machine that could learn to recognize patterns and categorize them into different classes. Rosenblatt's model consisted of a single-layer network of neurons, where each neuron computed a weighted sum of its inputs, compared the result to a threshold, and then produced an output based on this comparison. The perceptron algorithm involved an iterative learning process where the weights of the inputs were adjusted to minimize the error in the output. If the perceptron made a mistake in its prediction, the weights were updated in the direction that would reduce the error. This iterative adjustment allowed the perceptron to learn from examples and improve its performance over time. Despite its success, the perceptron had limitations, particularly in solving non-linear problems.

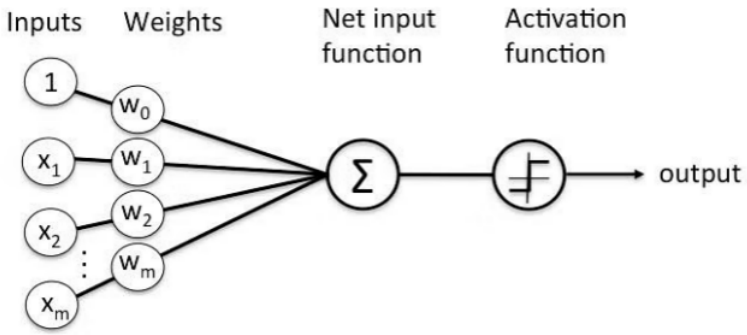


Figure 2.2: Perceptron, source: [24].

2.3.2 Evolution into Deep Learning (Neural Networks)

Early neural networks like the perceptron faced limitations in solving non-linear problems. These issues were later addressed through the development of multi-layer networks and advanced learning algorithms, leading to deep learning. Deep learning, a branch of machine learning, concentrates on utilizing extensive, multi-layered neural networks known as deep neural networks (DNNs) to model complex patterns in data. The backpropagation algorithm [30], which efficiently updates weights by computing the gradient of the loss function, played a crucial role in training these networks. The Backpropagation works in two main phases:

- Forward Pass: Input data is passed through the network layer by layer to generate an output.
- Backward Pass: The output is compared to the desired result using a loss function, and the error is propagated back through the network. During this phase, the algorithm calculates the gradient of the loss function concerning each weight and updates the weights to reduce the error.

This process of adjusting weights based on the gradient ensures that the network learns from its errors and improves its accuracy over time.

The deep learning's rise in the 2010s [21] was driven by increased computational power from GPUs, the availability of vast amounts of digital data, and significant algorithmic innovations. Deep learning has made significant contributions in several areas such as computer vision with CNNs, natural language processing, and speech recognition.

2.3.3 Recurrent Neural Networks (RNN) & Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) are a type of neural network specifically designed to handle sequential data. Unlike feedforward neural networks, RNNs have connections that create directed cycles, allowing information to persist across time steps. This makes RNNs particularly effective for tasks such as time series prediction, natural language processing, and speech recognition [15].

The standard RNN suffers from the vanishing gradient problem, which limits its ability to learn long-range dependencies in sequences. To overcome this issue, the Long Short-Term Memory (LSTM) architecture was developed [17]. LSTMs incorporate a memory cell that can maintain information across time steps, and gates that regulate the flow of information into and out of the cell.

The LSTM model is defined by the following set of equations [40]:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.10}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.11}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.12}$$
$$h_t = o_t * \tanh(C_t) \tag{2.13}$$

Here, f_t, i_t, and o_t are the forget, input, and output gates, respectively; C_t is the cell state, and h_t is the hidden state at time t. The LSTM architecture addresses the limitations of standard RNNs by controlling the information that is passed through the network, allowing for the retention of long-term dependencies.

2.3.4 Graph Neural Networks (GNN)

Graph Neural Networks (GNN) are designed to operate on data structured as graphs, where the relationships between nodes (entities) are as important as the features of the nodes themselves. GNNs are widely used in domains such as social network analysis, molecular chemistry, and recommendation systems [20, 31].

A graph $G = (V, E)$ is defined by a set of nodes V and edges E that connect these nodes. In a GNN, the learning process involves the propagation of information along the edges of the graph. Each node updates its representation by aggregating information from its neighbors, a

process which can be described as:

$$h_v^{(k+1)} = \text{AGGREGATE}\left(\left\{h_u^{(k)} \mid u \in \mathcal{N}(v)\right\}\right) \tag{2.14}$$

where $h_v^{(k)}$ is the representation of node v at the k-th layer, and $\mathcal{N}(v)$ represents the set of neighbors of node v. The `AGGREGATE` function can take different forms, such as summation, mean, or max-pooling.

2.3.4.1 Neural Network Convolution (NNConv)

Neural Network Convolution (NNConv) is a type of convolutional layer used in Graph Neural Networks (GNNs) and was applied in our GNN model to incorporate both node and edge features. In standard Graph Convolutional Networks (GCNs), the message-passing process primarily relies on node features and the graph's structure, represented by the adjacency matrix, without explicitly considering edge features. NNConv, however, introduces a mechanism that transforms edge features using a neural network. These transformed edge features are then incorporated into the message-passing process, enabling the model to capture richer representations by taking into account both node features and the relationships between nodes through edges.

The message-passing rule in NNConv is formulated as:

$$x_i' = \sum_{j \in \mathcal{N}(i)} f_\theta(e_{ij}) \cdot x_j$$

where x_i' is the updated feature vector of node i, $\mathcal{N}(i)$ represents the neighbors of node i, x_j is the feature vector of node j, and e_{ij} are the edge features between nodes i and j. The neural network $f_\theta(e_{ij})$ transforms the edge features into weights for the message-passing operation. This allows NNConv to fully exploit both node and edge attributes during training.

NNConv has shown to be effective in tasks where relationships between entities, represented by edge features, are essential, such as molecular graphs [12, 13, 33].

Chapter 3

Related Work

The modern era of football has been revolutionized by the use of data analytics and advanced machine learning techniques. This chapter explores the latest developments and trends in sports analytics, specifically focusing on football. By examining the intersections of network science and deep learning, we aim to highlight the innovative approaches that have transformed our understanding of the game. This chapter will also look at the earlier research that has set the base for the current study, giving a full background to the analytical methods used in this thesis.

3.1 Sports Analytics

Sports analytics has significantly evolved over the past few decades, becoming an integral part of football strategies and decision-making processes. The primary focus of sports analytics has been on enhancing team performance, optimizing player selection, and predicting match outcomes. One of the notable studies in this field is by Eggels et al. [10], where the authors present a method for explaining football match outcomes through predictive analytics centered on goal-scoring opportunities.

In their study, Eggels et al. [10] proposed a model that accurately aligns the probabilities of goal-scoring opportunities with the actual outcomes observed in elite football matches. Their predictive model effectively demonstrates the relationship between goal-scoring chances and match outcomes, indicating a high degree of reliability in its predictions. This study highlighted the importance of estimating the probability of scoring for individual goal-scoring opportunities, which allows analysts to move beyond simple win-loss predictions and gain a deeper understanding of the factors influencing match outcomes.

Further research by Eryarsoy et al. [11] explored the potential of predicting football game outcomes using a combination of single and ensemble analytics methods. This study analyzed data from the Turkish Super League, covering a decade (2007-2017), to identify the most effective approach to predict match outcomes and the factors influencing those results. The results suggested that ensemble methods were more successful compared to single algorithms like Neural

Networks, k-Nearest Neighbors (kNN), and Support Vector Machines (SVMs). The superiority of ensemble methods, as demonstrated in this research, underscores the complexity of football data and the need for sophisticated analytical techniques to derive meaningful conclusions.

Another significant contribution to sports analytics is the work of Decroos et al. [9], who developed the Valuing Actions by Estimating Probabilities (VAEP) framework to evaluate on-the-ball actions in football. This framework assesses the impact of each action on the probability of both scoring and conceding goals, providing a more comprehensive evaluation of player contributions. Unlike traditional metrics like goals and assists, VAEP values every action (such as passes, dribbles, and tackles) based on its influence on match outcomes.

3.2 Network Science in Football

Network science has emerged as a powerful tool for analyzing the complex interactions in football, particularly in understanding passing patterns and team dynamics. Traditional statistics in football often prioritize metrics such as goals and shots; however, network science provides a new lens through which the intricate passing sequences and positional play that define a team's style can be analyzed.

An interesting study in this field is by Peña et al. [27], which highlights the significance of passing sequences at the player level in discerning unique playing styles in football. Peña et al. [27] argue that understanding how teams circulate the ball through passing is essential in characterizing their distinctive style of play. To achieve this, they introduce the concept of passing motifs—sequences of three consecutive passes—as a means to analyze players' passing behaviors. By treating passing frequencies as digital fingerprints, the study quantifies the similarity between different players and explores the potential identification of replacements for influential players like Xavi at FC Barcelona. The analysis reveals significant information on player similarities, clustering patterns, and distance scores based on their respective playing styles.

Continuing the exploration of network science in football, Gonçalves et al. [14] investigate the impact of passing networks and player movement dynamics on team performance in youth association football. The authors examine the correlation between passing networks, positioning variables, and match outcomes to better understand the elements contributing to success on the field. Their research reveals that team passing dependency and intra-team passing relations play a significant role in determining a team's performance. This comprehensive analysis underscores the importance of cohesive passing networks and effective player positioning, particularly in youth elite football.

Another notable study is by Cintia et al. [6], which applied network theory to analyze the structural properties of football passing networks. The authors focused on the Italian Serie A league, where they investigated how the network centrality of players and the overall team structure correlate with match outcomes. The findings suggest that teams with higher centralization in their passing networks tend to have better performance, emphasizing the

strategic importance of key playmakers who act as hubs within the team's network.

3.3 Deep Learning and Graph Learning in Football Analytics

The advent of deep learning has revolutionized many fields, including sports analytics. In football, deep learning models have shown significant promise in capturing the complex, nonlinear relationships present in the game. Using large datasets, these models can identify patterns and make predictions that were previously unattainable with traditional statistical methods.

3.3.1 Deep Learning in Football Analytics

Deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have been widely applied in football analytics to analyze video data, player movements, and match outcomes. For example, Tiwari et al. [39] demonstrated the use of neural networks and deep learning techniques for predicting football match results. Their study employed advanced neural network architectures to analyze match-related data and highlighted the effectiveness of deep learning in outperforming traditional machine learning techniques for result prediction.

Similarly, Herold et al. [16] and Cao [5] employed machine learning techniques to predict goal-scoring opportunities in football. Herold et al. focused on utilizing spatiotemporal tracking data to analyze player movements and team strategies, while Cao concentrated on predicting shooting outcomes by examining the dynamics of passing paths. These studies demonstrate the value of deep learning and data-driven techniques in improving tactical analysis and offering a real-time understanding of game dynamics and player behavior.

3.3.2 Graph Learning and Graph Neural Networks in Football

Graph learning, and specifically Graph Neural Networks (GNNs), have emerged as powerful tools for modeling relational data in various domains. In football, passing networks naturally lend themselves to graph-based representations, where players are nodes, and passes between them are edges. GNNs have been applied to capture the structural properties of these networks and predict outcomes based on the connectivity patterns within a team.

Battaglia et al. [1] proposed the concept of Graph Neural Networks (GNNs) as a framework for relational reasoning, introducing their applications across various domains. While their work was not focused on sports analytics, it established foundational principles for modeling complex systems with relational structures, such as player interactions in football.

Although also Tacchetti et al. [38] did not specifically address football, they extended GNN principles to multi-agent systems, demonstrating their utility in modeling dynamic

interactions. Their work showed how GNNs could facilitate understanding of agents' cooperative and competitive behaviors in environments requiring structured reasoning, highlighting their potential for applications like analyzing team strategies in sports contexts.

In the context of football analytics, Rana [28] explored the use of Graph Convolutional Networks (GCNs) for event detection, demonstrating how graph-based models could effectively capture spatial and temporal relationships in match data. By modeling football events as a graph, this work further illustrated the capabilities of GNNs in sports analysis, particularly for understanding player interactions and their outcomes during matches.

Chapter 4

Exploratory Analysis

First, we start by explaining the dataset that we use and then present an exploratory analysis. The description of the dataset shows how the data are organized and the exploratory analysis involves using different visualizations and statistical methods to understand how features are related and how they affect passing sequence outcomes. This initial analysis sets the foundation for future modeling and prediction tasks.

4.1 Data Collection and Preprocessing

Our analysis begins by collecting event data from StatsBomb using the *statsbombpy* Python package [35]. This data includes various match events, such as passes, shots, tackles, or interceptions, which we filter to focus only on passing events.

We extract all passes from matches within the 2015-2016 seasons across Europe's top 5 leagues (Premier League, La Liga, Serie A, Bundesliga and Ligue 1). Each pass contains relevant attributes, and passes within the same possession are grouped into sequences, with unique sequence IDs generated for each possession.

We identify and label the outcomes of passing sequences by determining whether a sequence led to a goal. Sequences resulting in goals are labeled as positive outcomes, while others are labeled as negative.

The final dataset contains detailed information on each passing sequence and their corresponding outcomes. The dataset was preprocessed to ensure consistency and accuracy, including the removal of duplicate columns and the standardization of boolean values.

4.2 Description of the Dataset

The *passes_sequences* structure contains information about each passing sequence derived from event data. A more detailed description of the dataset can be seen in appendix A. It contains **1,250,601** rows and **31** columns, depicting complete information on passing sequences and their attributes. Key features include:

- Each passing sequence and individual pass are assigned unique identifiers:
  - *sequence_id*: Unique identifier for each passing sequence.
  - *pass_id*: Unique identifier for each pass within a sequence.
- *aerial_won*, *goal_assist*, *cross*, *switch*, *under_pressure*: Boolean indicators of specific pass attributes.

  The *goal_assist* attribute requires further explanation. While it indicates whether a pass directly leads to a goal, this attribute was not used as a feature in the machine learning models. Instead, it was used to create the passing sequences. Sequences were defined based on possession changing, and if the final pass in the possession was a *goal_assist*, the outcome of all passes in that sequence was marked as goal/true. Otherwise, the sequence outcome was marked as no goal/false.
- *angle*, *duration*, *length*: Numeric values representing pass characteristics.
- *body_part*, *height*, *play_pattern*, *technique*, *type*: Categorical descriptions of the pass.
- *location*, *end_location*: Coordinates marking the start and end points of each pass. These coordinates are represented as (x, y) pairs. The x-axis runs from 0 to 120, representing the length of the pitch, and the y-axis runs from 0 to 80, representing the width of the pitch. The starting location (*location*) refers to where the pass originated and the ending location (*end_location*) indicates where the pass was received.
- *player*, *player_id*, *recipient*: Information about the player executing and receiving the pass.
- *possession*, *possession_team_id*, *match_id*, *minute*, *second*, *period*: Contextual details of the pass event.

The *sequences_outcome* structure includes:

- *sequence_id*: Unique identifier for each passing sequence.
- *outcome*: Boolean indicating whether the sequence resulted in a goal.

This dataset provides the foundation for the exploratory analysis, with minor adjustments applied during the modeling and prediction phases, which will be detailed in later chapters.

4.3 Exploratory Analysis

In this chapter, we take a look and analyze the passing sequences outlined in the former section. The explanation of patterns of the passing procedures is important for understanding match outcomes and, in a specific case, as in our work, for understanding the impact on the result of a play, e.g., goal creation. We perform statistical methods and use visualization tools to investigate aspects such as the frequency, distribution, and character of passing sequences in different contexts. These methods reveal underlying patterns in passing behavior.

4.3.1 General Analysis of Passing Sequences and Outcome

We start by analyzing the distribution of passing sequences per match a histogram 4.1.

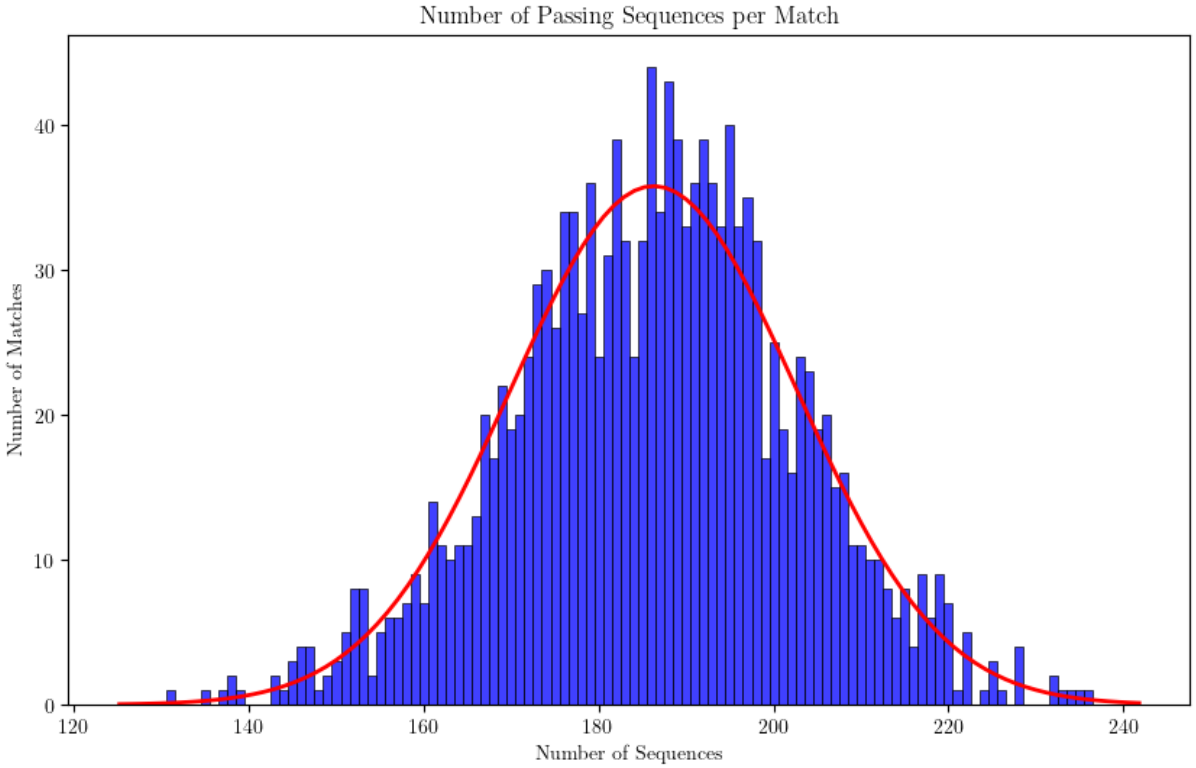


Figure 4.1: Distribution of passing sequences per match

The red curve on the histogram represents a fitted normal distribution, providing a visual benchmark to evaluate how closely the distribution of passing sequences aligns with a normal distribution. The central peak of the histogram, located around 190 passing sequences per match, indicates that the majority of matches have a passing sequence count close to this value. In contrast, the histogram shows fewer matches at the lower end of the spectrum, around 130 passing sequences, though these matches still form a notable segment of the dataset. The upper

boundary of passing activity is marked by matches with roughly 230 passing sequences, which represent the highest levels of passing observed in the dataset.

Regarding the size of the sequences, the overall average sequence size (number of passes in a sequence) across all matches, is approximately 5.25. Figure 4.2 illustrates the distribution of sequence sizes, measured as the number of passes in each sequence, categorized by whether the sequence resulted in a goal or not. The plot uses a logarithmic scale for the y-axis to better represent the wide range of sequence sizes.

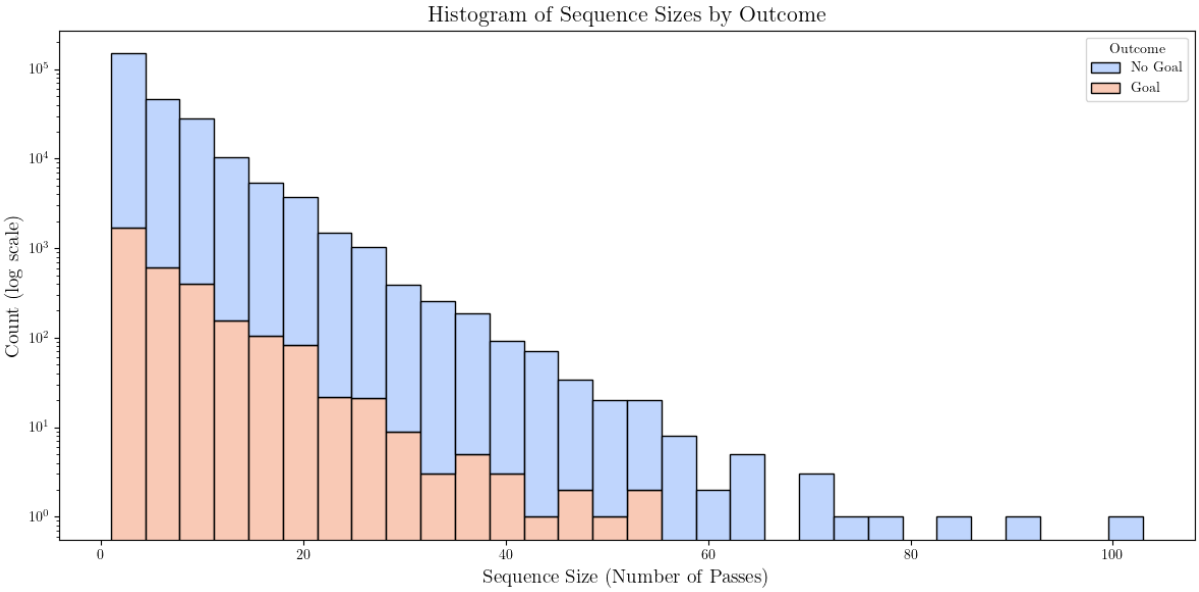


Figure 4.2: Distribution of Sequence Sizes by Outcome (Log Scale)

The plot reveals that shorter sequences, generally with fewer than 20 passes, are the most common regardless of the outcome. However, goal-scoring sequences are relatively more common within the range of approximately 5 to 30 passes compared to longer sequences. Notably, as the number of passes in a sequence increases beyond 20, the proportion of goal-scoring sequences decreases significantly.

In sequences with more than 30 passes, the number of sequences decreases sharply for both outcomes, with very few sequences extending beyond 60 passes. The distribution suggests that while short sequences are more frequent, medium-length sequences (around 10 to 20 passes) tend to be slightly more effective in leading to goals compared to much longer sequences.

Table 4.1 shows the number of sequences and the number of sequences with goals, separated by league.

Competition	Total Sequences	Goal-Sequences	Goal-Sequences (%)
England - Premier League	253,189	3,865	1.53
France - Ligue 1	260,722	3,327	1.28
Germany - 1. Bundesliga	212,370	3,285	1.55
Italy - Serie A	269,355	3,630	1.35
Spain - La Liga	254,965	4,108	1.61

Table 4.1: Number and Percentage of Goal-Scoring Passing Sequences Across the Top Five European Leagues

- **Spain - La Liga** recorded the highest percentage of goal-scoring sequences at 1.61%. This suggests that, on average, passing sequences in La Liga are more likely to result in goals compared to other leagues. This could be attributed to the style of play in La Liga, which is often characterized by quick, short passing sequences and high technical ability among players.
- **Germany - 1. Bundesliga** follows closely with 1.55% of passing sequences resulting in goals. The Bundesliga is known for its fast-paced, direct style of play, which may contribute to the relatively high conversion rate of sequences into goals.
- **England - Premier League**, despite being one of the most competitive leagues, has a slightly lower percentage of goal-scoring sequences at 1.53%. This could be due to the physical and defensive nature of the league, where teams often focus on maintaining defensive solidity.
- **Italy - Serie A** and **France - Ligue 1** have the lowest percentages of goal-scoring sequences, at 1.35% and 1.28% respectively. Serie A is traditionally known for its strong defensive tactics, which may explain the lower conversion rate of passing sequences into goals. Similarly, Ligue 1, while producing top-quality attacking talent, may see fewer goals per sequence due to the tactical approaches employed by many teams.

Repeating this analysis by team, in Table 4.2 we show the top 3 and bottom 3 teams in success rate of their passing sequences:

Team	**Total Sequences**	**Goal Scoring Sequences**	**Success Rate (%)**
Real Madrid	3,985	90	2.26
Paris Saint-Germain	3,309	72	2.18
Barcelona	3,687	79	2.14
Watford	3,245	22	0.68
Troyes	3,099	21	0.68
Atalanta	3,558	24	0.67

Table 4.2: Top and Bottom Teams by Goal Scoring Success Rate

- **Real Madrid** leads with a 2.26% success rate, showcasing their strong offense by turning passes into goals.
- **Paris Saint-Germain (PSG)** and **Barcelona** have success rates of 2.18% and 2.14%, respectively. These teams are known for their creative playmakers and clinical finishers. PSG's slight advantage may be due to their more direct style, leading to quicker goal-scoring opportunities.
- On the lower end, **Watford**, **Troyes**, and **Atalanta** exhibit much lower success rates, all around 0.68%. These figures suggest that these teams struggle to convert their passing sequences into goals. Factors such as less efficient finishing, a more defensive or cautious approach, or a lack of quality in the final third could contribute to these lower success rates.
- The significant disparity between the top and bottom teams underscores the importance of attacking efficiency in football. While teams like Real Madrid, PSG, and Barcelona capitalize on their sequences with high success, teams like Watford, Troyes, and Atalanta face challenges in translating their buildup play into tangible outcomes on the scoreboard.

4.3.1.1 Pass Duration, Length and Outcome

Examining the duration and length statistics, as presented in Table 4.3, provides valuable information on the characteristics of the pass sequences in the dataset.

Statistic	**Pass Duration (seconds)**	**Pass Length (yards)**
Mean	1.582	21.642
Min	0.0001	0.0000
Max	19.445	119.534
Standard Deviation	0.895	14.789

Table 4.3: Statistics for Pass Duration and Pass Length

The mean pass duration is approximately 1.58 seconds, with a minimum of nearly zero and a maximum of over 19.445 seconds. The relatively low standard deviation of 0.895 seconds indicates that most passes occur within a narrow time range, suggesting a consistent pace in passing sequences.

Regarding pass length, the average pass covers around 21.64 yards, with a notable range from very short (0 yards), possibly indicating a pass to a player very close or directly adjacent. The maximum pass length reaches 119.534 yards, almost the full length of a football pitch, indicating the occurrence of long passes, likely clearances, or direct play strategies aimed at quickly advancing the ball upfield. The standard deviation of 14.79 yards indicates a wider variability in pass lengths, possibly reflecting different tactical approaches across teams. In figure 4.3 we can see that in goal-scoring sequences, there appears to be a slightly higher proportion of shorter to medium-length passes compared to non-scoring sequences, which tend to feature longer passes more frequently. This could suggest that shorter, more controlled passes are more effective in building up towards a goal, while longer passes are more associated with defensive clearances or unsuccessful attempts to advance the play.

Overall, the data shows that while long passes are common in both goal-scoring and non-scoring sequences, the likelihood of scoring increases with shorter passes, which may indicate a more possession-based strategy leading to a successful goal-scoring opportunity.

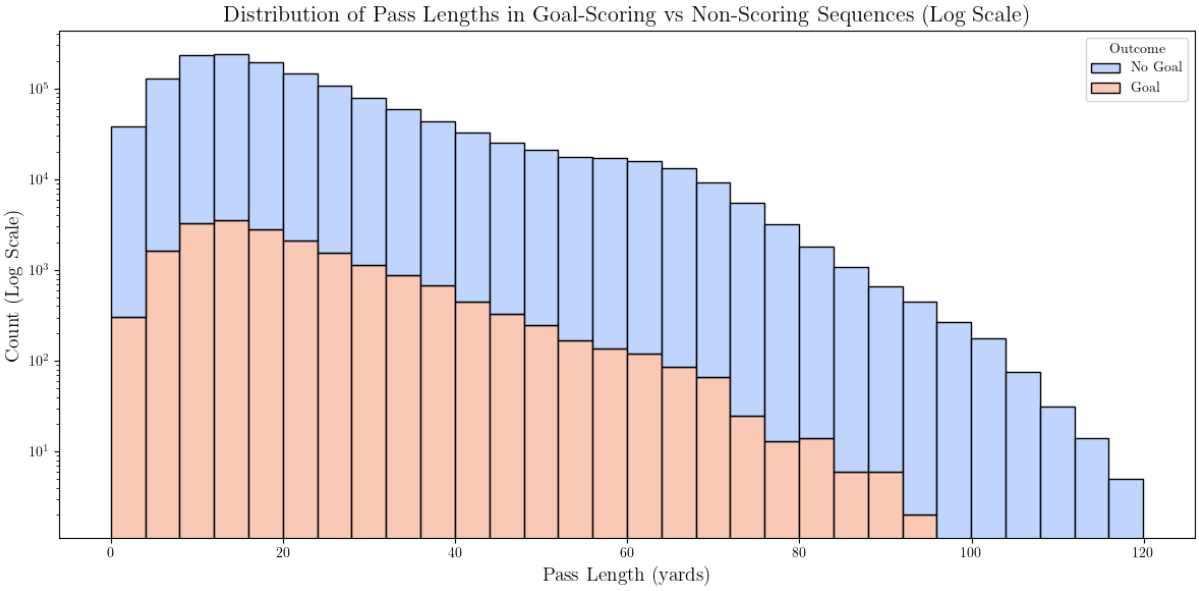


Figure 4.3: Distribution of Pass Lengths in Goal-Scoring vs Non-Scoring Sequences (Log Scale)

4.3.1.2 Pass Height and Goal Assists

As depicted in Figure 4.4, ground passes constitute the majority of goal assists, making up 47.73% of the total. This prevalence indicates that maintaining the ball on the ground increases the likelihood of converting passes into goals, likely due to the increased control and accuracy

that ground passes afford.

High passes, while less frequent than ground passes, make up 39.95% of goal-scoring assists. These passes are often used to bypass defensive lines or quickly transition play, indicating their importance in specific game scenarios.

Finally, low passes, representing only 12.32% of goal-scoring assists, are used less frequently compared to ground and high passes. While they can be valuable in certain situations, their lower percentage suggests they are not as effective as other pass types in directly leading to goals.

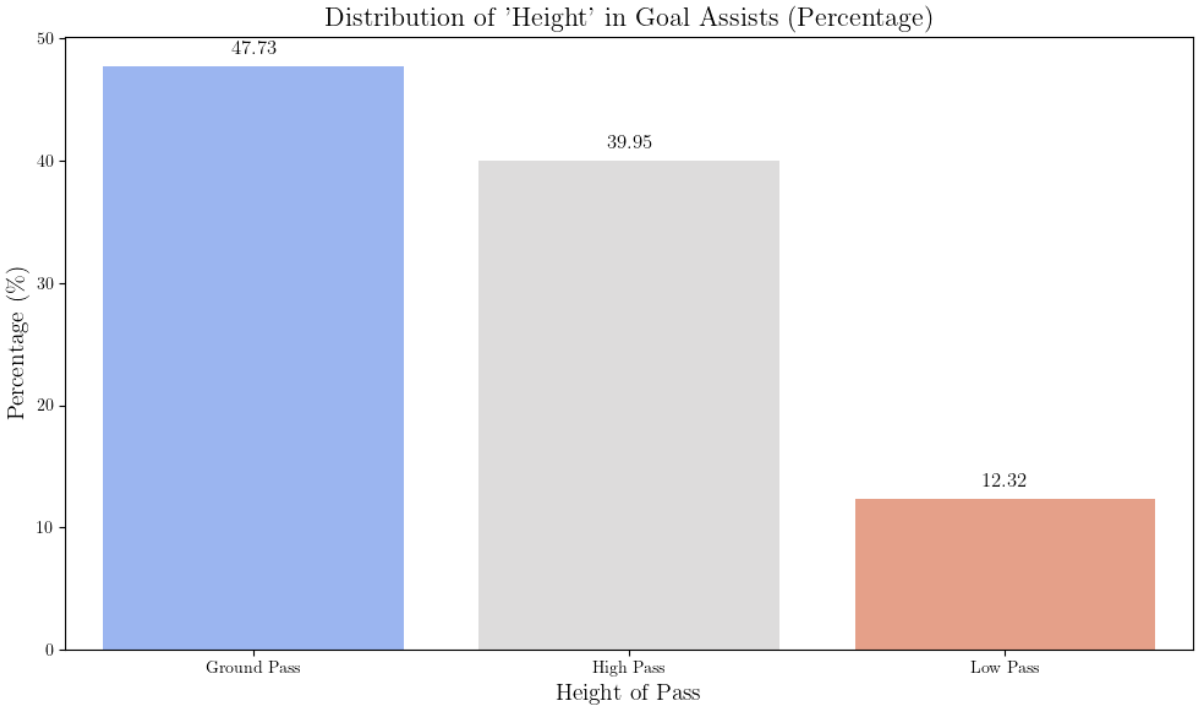


Figure 4.4: Distribution of Pass Height in Goal Assists (Percentage)

4.3.1.3 Body Part used for Goal Assists

The distribution of body parts used in goal assists, as detailed in Table 4.4, highlights the predominance, as expected, of the feet in executing passes that lead to goals. Specifically, the **right foot** is the most commonly used body part, accounting for nearly 63% of all goal assists. This is followed by the **left foot**, which contributes approximately 33% of such passes. These results emphasize the critical role of both feet, particularly the right foot, in delivering precise passes that result in goal-scoring opportunities.

Body Part	Percentage (%)
Right Foot	62.73
Left Foot	32.99
Head	2.68
Other	0.96
No Touch	0.51
Drop Kick	0.10
Keeper Arm	0.03

Table 4.4: Distribution of Body Parts in Goal Assists

Interestingly, **headers** make up about 2.68% of goal assists, reflecting the importance of aerial play, especially in set-piece situations or when delivering crosses into the penalty area. While the contribution of headers is significantly lower compared to foot passes, their impact in specific scenarios should not be underestimated.

Other body parts such as the **keeper arm**, **drop kick** (or keeper volley), and **no touch** play a minimal role, each contributing less than 1% to the total of goal assists.

4.3.1.4 Aerial Duels and Outcome

The bar plot in Figure 4.5 illustrates the percentage distribution of the *Aerial Won* attribute in goal-scoring sequences. The data shows some interesting aspects:

- **Prevalence of Ground Play**: A significant majority of goal-scoring sequences (86.51%) occur without winning an aerial duel, similar to the 84.38% in non-goal-scoring sequences. This indicates that most successful attacks are built on the ground, suggesting that ground passes and dribbling are more effective in leading to goals than aerial play.
- **Limited Role of Aerial Play**: Only 13.49% of goal-scoring sequences involve winning an aerial duel, compared to 15.62% in non-goal-scoring sequences. While aerial passes can be important in specific attacking strategies (e.g., crosses into the box), this relatively low percentage suggests that they are not the dominant factor in generating goal-scoring sequences during general play.

These findings align with modern football tactics, where maintaining control of the ball and executing precise passes on the ground are often prioritized to break down opposing defenses. However, the importance of aerial duels should not be entirely dismissed, as they can still be relevant in specific scenarios, such as set-pieces or late-game situations where direct play is required.

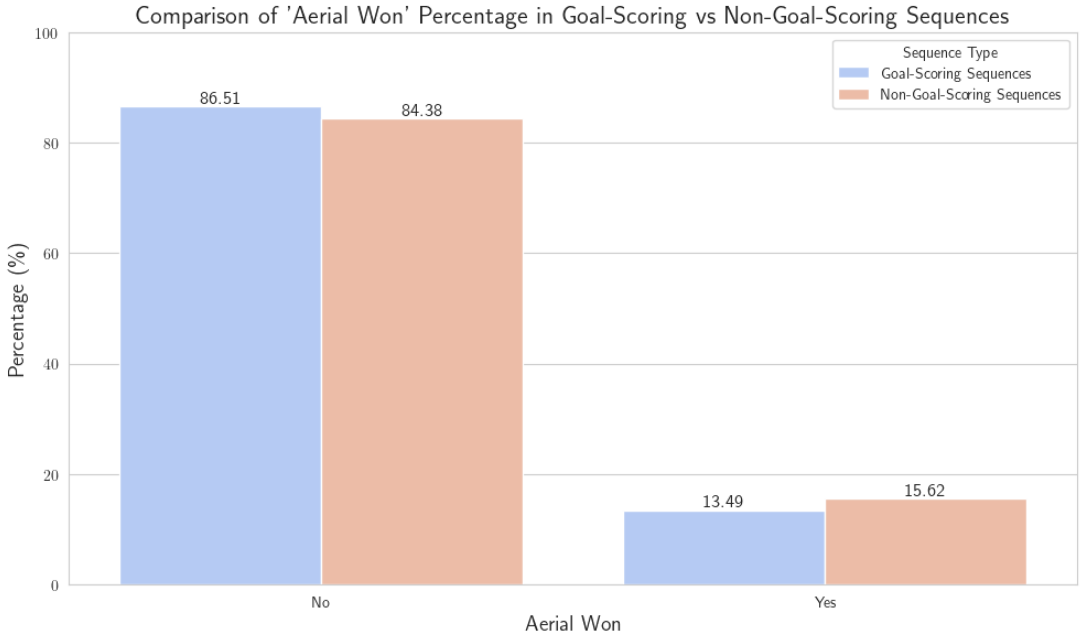


Figure 4.5: Comparison of *Aerial Won* Percentage in Goal-Scoring vs Non-Goal-Scoring Sequences

4.3.1.5 Passes Under Pressure and Outcome

The analysis of goal-scoring sequences in the top five European leagues shows interesting findings about the impact of passes made under pressure. As shown in Table 4.5, Germany's 1. Bundesliga exhibits the highest percentage of under-pressure passes at 15.92%. This could indicate a more aggressive and pressing style of play, where goal-scoring opportunities often arise in high-pressure situations.

France's Ligue 1 follows closely with 14.64%, which also suggests a relatively high incidence of successful goal-scoring passes made under defensive pressure.The similarity in pressure percentages between goal-scoring and non-goal-scoring sequences in both Ligue 1 and the Bundesliga suggests that pressure is a constant factor in both successful and unsuccessful sequences.

Italy's Serie A and England's Premier League show comparable percentages for under-pressure passes in goal-scoring sequences, at 13.99% and 13.20% respectively. Interestingly, both leagues show slightly higher percentages of under-pressure passes in non-goal-scoring sequences, which may indicate that while pressure is present, it does not necessarily increase the likelihood of scoring.

Spain's La Liga has the lowest percentage of under-pressure passes in goal-scoring sequences at 11.90%.This is closely matched by the percentage in non-goal-scoring sequences (12.04%), suggesting that La Liga teams might excel at creating goal-scoring opportunities with less defensive pressure, possibly due to a focus on technical play, movement, and positional awareness over physical pressing.

These differences highlight the tactical variations between the leagues and how these impact the creation of goal-scoring opportunities.

League	Goal-Scoring Sequences (%)	Non-Goal-Scoring Sequences (%)
England - Premier League	13.20	13.12
France - Ligue 1	14.64	14.55
Germany - 1. Bundesliga	15.92	15.55
Italy - Serie A	13.99	14.50
Spain - La Liga	11.90	12.04

Table 4.5: Percentage of 'Under Pressure' Passes in Goal-Scoring and Non-Goal-Scoring Sequences by League

As shown in Table 4.6, the significant variation in under-pressure passes in goal-scoring sequences among different teams illustrates the diversity in tactical approaches across European football. Teams such as Augsburg and Hertha Berlin, which feature the highest percentages of under-pressure passes in goal-scoring sequences, indicate a propensity to score goals even under challenging defensive conditions. This might suggest a more resilient or direct attacking style, where passes are often made under defensive pressure but still result in goals.

On the other hand, teams like Celta Vigo and Las Palmas, which have the lowest percentages of under-pressure passes in goal-scoring sequences, may prefer a more controlled and less risky approach to their build-up play. This approach ensures that passes leading to goals are made in less pressured situations. Such a style could reflect a more cautious or possession-oriented playing strategy, prioritizing control and reducing defensive pressure before executing the decisive pass.

Teams with Highest Percentage	Under Pressure (%)	Teams with Lowest Percentage	Under Pressure (%)
Augsburg	27.05	Celta Vigo	8.65
Hertha Berlin	25.30	Las Palmas	9.09
Werder Bremen	22.95	Aston Villa	9.38
Hamburger SV	22.73	Bordeaux	9.79
FSV Mainz 05	22.58	Rayo Vallecano	10.59

Table 4.6: Teams with the Highest and Lowest Percentage of Under Pressure Goal-Scoring Passes

4.3.1.6 Goal Assists Locations

The heatmap visualization in Figure 4.6 provides a visualization into the distribution of goal assists locations across the top five European football leagues: England's Premier League, France's

Ligue 1, Germany's Bundesliga, Italy's Serie A, and Spain's La Liga. The varying intensities on the heatmaps illustrate the key areas of the pitch where goal assists are most frequently initiated.

Across most leagues, including the Premier League, La Liga, and Ligue 1, there is a clear concentration of goal assists originating from central areas of the pitch, particularly within and just outside the penalty box. This suggests a common tactical approach where teams focus their attacks through the middle.

Penalty Box Hotspot: In all leagues, the most intense areas on the heatmaps are centered around the penalty box, confirming that goal assists are most frequently delivered from positions close to the goal. This emphasizes the importance of the penalty area in creating and converting goal-scoring opportunities.

League-Specific Patterns:

- **Premier League, La Liga, and Ligue 1:** These leagues show similar patterns, with a balanced distribution of goal assists coming from both central areas and wide positions. This reflects a diverse tactical approach, where teams effectively utilize both central penetration and wing play, especially in zones near the penalty box.
- **Bundesliga (Germany):** The heatmap for the Bundesliga indicates a noticeable shift towards the right side of the pitch. This could suggest a tactical preference in the Bundesliga for attacking down the right flank, with teams potentially favoring crosses or cut-back passes from this side to create goal-scoring opportunities.
- **Serie A (Italy):** In contrast to the Bundesliga, Serie A shows a greater intensity of goal assists originating from the left side of the pitch. This could indicate a preference for left-sided attacks, with teams potentially exploiting the left wing to create scoring chances.

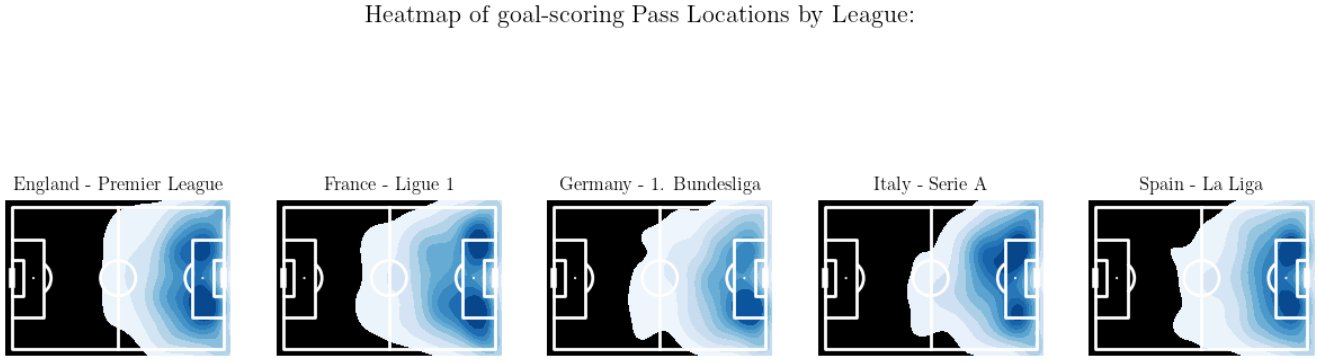


Figure 4.6: Heatmap of goal-scoring pass locations by league

4.3.1.7 Pass Angle and Goal-scoring Sequences

The polar histogram shown in Figure 4.7 illustrates the distribution of pass angles in goal-scoring sequences. The histogram reveals several interesting findings:

- **Directional Bias:** The distribution of pass angles is not uniform. There is a noticeable concentration of passes around the 0° and 90°/270° directions, corresponding to forward and sideways passes, respectively. This indicates that successful passes leading to goals are more likely to be direct or lateral, emphasizing the importance of advancing the play towards the goal or moving the ball across the field to exploit spaces.
- **Forward Passes:** The highest frequency of passes is observed in the forward direction, around 0°. This suggests that direct forward passes are a common strategy in goal-scoring sequences, highlighting the importance of progressive play in creating goal opportunities.
- **Sideways Passes:** A significant number of passes occur around the 90° and 270° marks, corresponding to sideways passes. This indicates that lateral passes are frequently utilized in sequences that ultimately result in goals. These passes help in stretching the defense, creating gaps, and opening up spaces that can be exploited for goal-scoring opportunities.
- **Backward Passes:** The distribution of passes around 180°, corresponding to backward passes, is less prominent. While backward passes can be useful in retaining possession or resetting the play, they are less frequently associated with sequences that lead directly to goals, compared to forward or sideways passes.

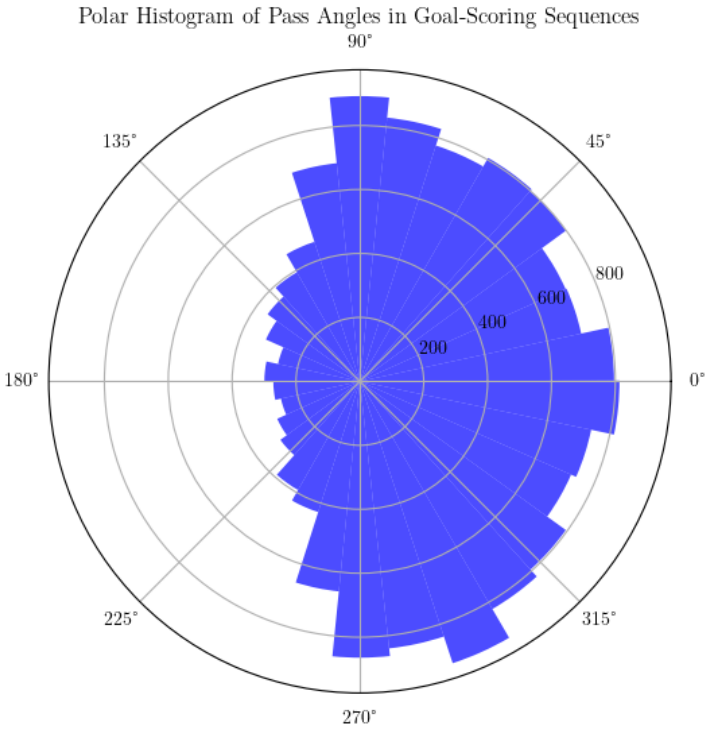


Figure 4.7: Polar Histogram of Pass Angles in Goal-Scoring Sequences

4.3.1.8 Crosses and Outcome

Figure 4.8 reveals that 4.5% of crosses lead to goals, which highlights their importance in generating goal-scoring opportunities. Despite the seemingly modest percentage, this figure indicates that crosses are particularly effective in directly contributing to goals. Crosses are typically aimed at delivering the ball into high-risk areas, which can catch defenses off-guard or provide an advantage to attacking players.

In contrast, only 1.4% of non-cross passes result in goals. This lower percentage is understandable given the much higher frequency of non-cross passes during typical gameplay. Non-cross passes include a wide variety of passing types, many of which are more focused on maintaining possession or gradually advancing the ball upfield, rather than immediately seeking to score.

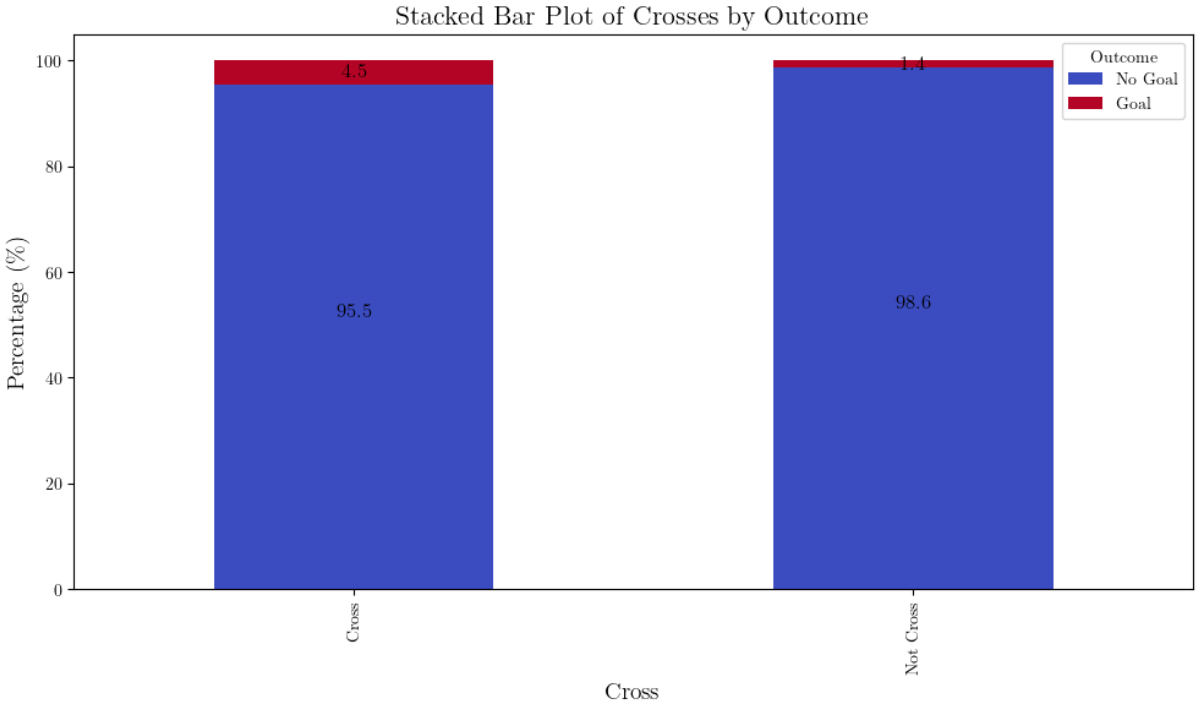


Figure 4.8: Stacked Bar Plot of Crosses by Outcome

4.3.1.9 Play Patterns and Goal Assists

In Figure 4.9, we show the distribution of play patterns in goal assists. The data reveals that 37.52% of all goal assists arise from regular play, indicating the importance of fluid, in-play action in creating scoring opportunities.

Set-pieces also play a significant role, with throw-ins (16.98%), free kicks (16.66%) and corners (13.21%) contributing substantially to goal creation. These findings highlight the value of effective set-piece strategies in football. Counter-attacks also have an interesting prevalence (8.93%).

In contrast, plays initiated from goal kicks (3.99%) or by the goalkeeper (1.82%) are less likely

to lead to goals. Kick-offs are the least likely to result in goals, contributing to just 0.89% of the goal assists. These results suggest that while all play patterns can lead to goals, the majority of successful sequences stem from regular, in-play situations and set-pieces.

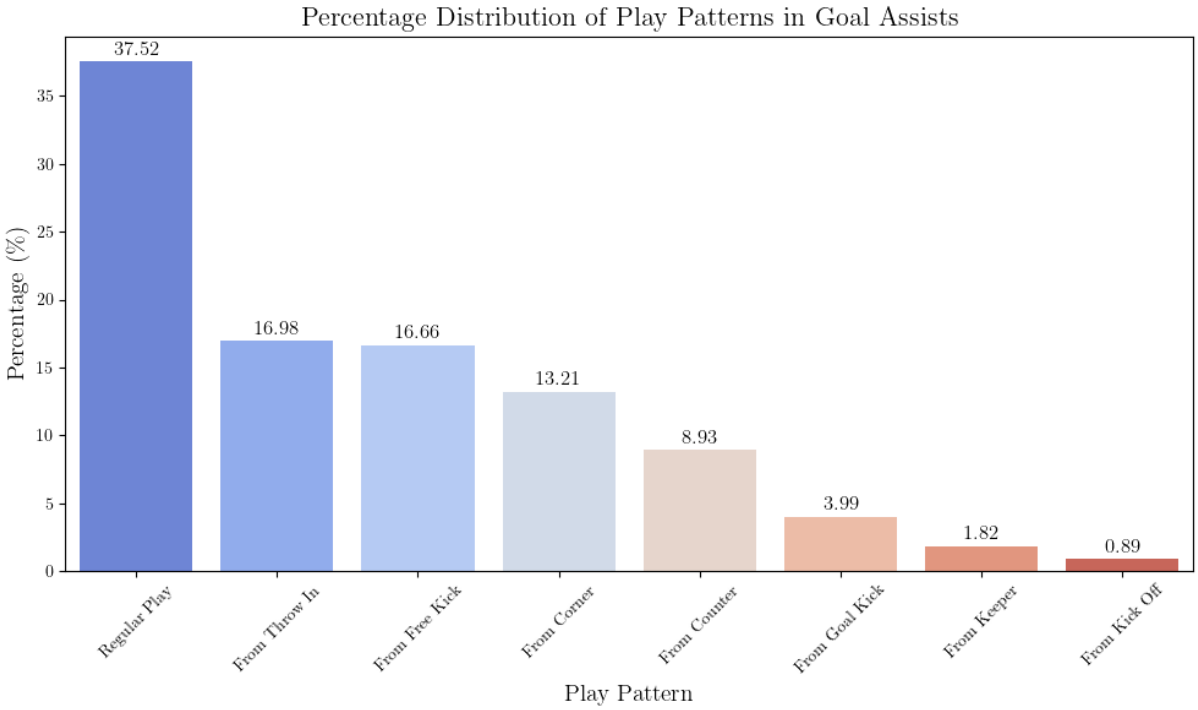


Figure 4.9: Percentage Distribution of Play Patterns in Goal Assists

4.3.1.10 Temporal Analysis of Goal Assists

Figure 4.10 shows the distribution of goal assists across different minutes of the match. The analysis reveals several key trends:

- There is an initial rise in the number of goal assists during the first 20 minutes, suggesting a gradual buildup of offensive opportunities as the match progresses.
- The middle period of the match (between the 20th and 80th minutes) exhibits substantial fluctuations, with the number of goal assists varying approximately between 20 and 50. This likely reflects the dynamic nature of the matches during this phase, where both teams attempt to exploit weaknesses.
- There is a noticeable decline in goal assists after the 90th minute. This may be because play tends to become less structured during stoppage time, resulting in fewer completed passing sequences. Additionally, teams that are ahead may shift to a more defensive approach, further reducing offensive opportunities.

Overall, the data indicates that goal assists are more frequent in the middle phase of the

match and close to the 80th minute, with a slight reduction in offensive intensity towards the end (during stoppage time).

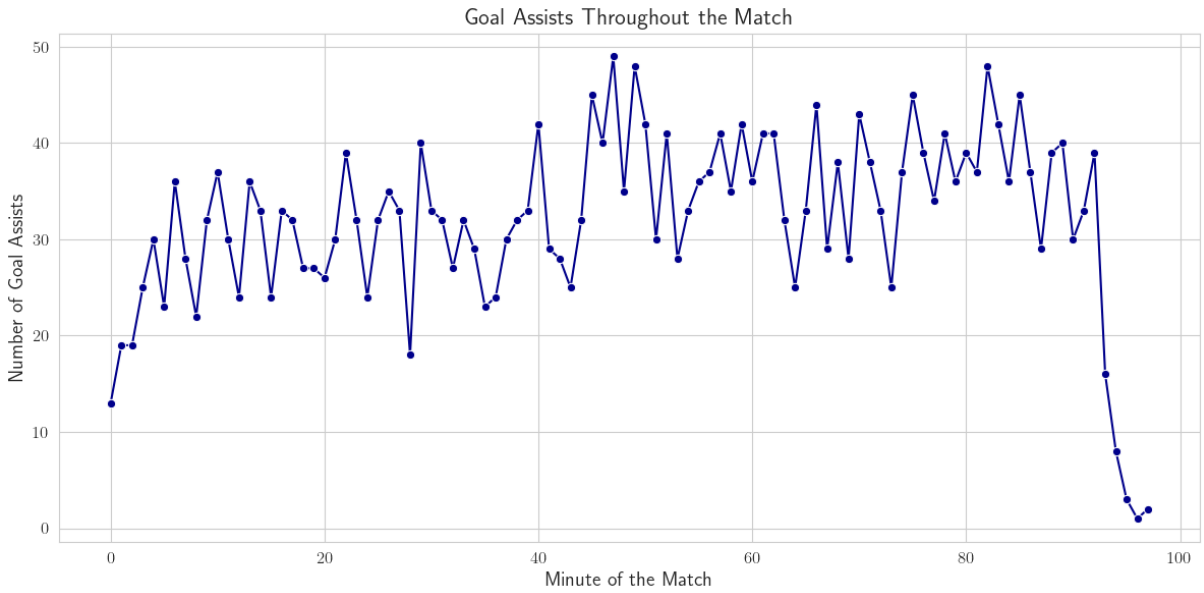


Figure 4.10: Goal Assists Throughout the Match

4.3.1.11 Player Position and Goal-scoring Sequences

Figure 4.11 shows the distribution of goal-scoring sequences by player position. Several interesting observations can be made from the data:

- **Defensive Positions Lead in Goal Involvement:** Right backs, left backs, and center backs are involved in the highest number of goal-scoring sequences, suggesting that modern defensive players play an active role in initiating or contributing to goal-scoring plays, likely because they exchange passes among themselves in defensive areas before launching an attacking move or possible providing key passes from deeper positions.
- **Midfielders Contribute Consistently:** Midfielders, particularly defensive and central midfielders, show a consistent contribution to goal-scoring sequences. This indicates their critical role in both supporting defense and creating offensive opportunities.
- **Goalkeepers and Center Forwards Least Involved:** Goalkeepers and center forwards show the least involvement in goal-scoring sequences. The low involvement of goalkeepers is expected due to their primary defensive role. However, the lower contribution of center forwards might suggest that they rely more on receiving final passes or crosses to finish scoring opportunities rather than being directly involved in the buildup.

These findings reflect the increasingly dynamic nature of football, where players from traditionally defensive positions are actively involved in offensive play, contributing to a more

fluid and interconnected team structure.

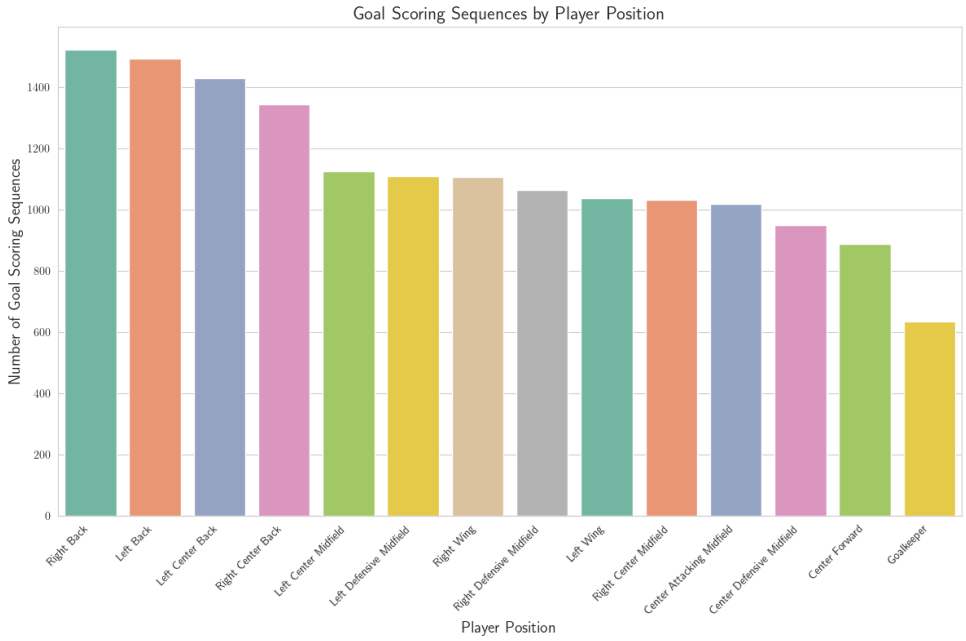


Figure 4.11: Goal Scoring Sequences by Player Position

Chapter 5

Predicting goals from passing events

In this section, we explore the encoding of features and the preparation of the dataset for the classification task of predicting goals from passing events. Additionally, we examine the characteristics of passing sequences and their significance in predicting goal outcomes.

5.1 Traditional Machine Learning

5.1.1 Dataset Modifications

Modifying the data set is essential to prepare the data for traditional machine learning models, which require structured and properly formatted inputs. Traditional models like Logistic Regression, Random Forest, and SVM are not designed to work directly with sequential data. Therefore, appropriate feature engineering techniques are often necessary to transform unstructured data, such as the passing sequences in our dataset, into structured feature vectors. The following subsections detail the preprocessing steps and their underlying reasons.

5.1.1.1 Categorical Feature Encoding

One-Hot Encoding: Categorical variables are converted into a one-hot encoded format, where each category for an individual instance is represented as a unique binary vector. This method ensures that machine learning models can process the categorical data without assuming any ordinal relationship between categories. The following columns were encoded:

Competition, *Body Part*, *Pass Height*, *Outcome Type*, *Play Pattern*, *Position*, and *Possession Team*.

5.1.1.2 Handling Missing Values

Dropping Rows with Missing Values: Rows with missing values in the `body_part` column are removed. Only individual passes without information on the body part are excluded, rather than entire sequences. Given the importance of this feature in analyzing passing sequences, removing these incomplete records ensured that the dataset remains clean and reliable without discarding full sequences unnecessarily.

5.1.1.3 Feature Engineering

Aggregation of Sequence-Level Features: For each passing sequence (`sequence_id`), several features are aggregated to provide a comprehensive representation of the whole sequence:

- **Pass Counts**: We calculate the total number of passes in each sequence, as well as the number of passes made under pressure and the number of aerial duels won.
- **Proportions**: We compute the proportion of passes made under pressure and the proportion of aerial duels won for each sequence.

Body Part, Height, and Position Counts: Counts of the different body parts, heights, and positions are calculated for each sequence, capturing the diversity and frequency of these attributes.

Aggregating Numerical Features: Numerical features like `duration` and `length` are aggregated by taking the mean and standard deviation across the sequence.

First and Last Pass Features: Features related to the first and last passes in each sequence are retained separately, offering relevant information about the sequence's initiation and conclusion.

Special Features:

- **Switch**: A switch refers to a pass or a series of passes that move the ball from one side of the pitch to the other. If any pass made the ball transition from one flank to the opposite side, the entire sequence is marked as a switch.
- **Cross Binary**: If any pass in a sequence was a cross, the sequence is marked with a binary indicator.

5.1.1.4 Training and Testing Dataset Splitting

For model evaluation, the dataset is split into training and testing sets, maintaining consistency across different datasets. This splitting is performed both on the overall dataset (including all leagues) and on individual league-specific datasets. Each dataset is divided as follows:

- **Training Set**: 80% of the data.
- **Testing Set**: 20% of the data.

This splitting strategy allows for robust model evaluation and comparison across different leagues, ensuring that models trained on one portion of the data could be reliably tested on unseen data.

The dataset splitting process ensures that the data is stratified based on the target variable (`outcome_x`), preserving the distribution of goals and non-goals across all subsets. This stratified approach helps maintain the balance between the classes, which is important when training models that are sensitive to class imbalance.

This process is consistently applied to all datasets, including those corresponding to individual leagues such as La Liga, Serie A, Premier League, Ligue 1, and Bundesliga.

5.1.1.5 Performance Metrics

The performance of each model is evaluated using a variety of metrics in a holdout test set. These metrics offer a comprehensive view of how well the models perform in classifying sequences that lead to goals versus those that do not. Below is a detailed explanation of each metric:

- **Accuracy**:
  Accuracy evaluates the model's overall correctness by determining the proportion of true predictions (both true positives and true negatives) out of the total number of predictions. It is defined as:
  $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
  where:
  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives
  - FN = False Negatives
- **Precision**:
  Precision, also referred to as Positive Predictive Value, measures how accurate the positive predictions are. It is the proportion of true positive predictions out of all positive predictions made by the model. The formula is:
  $$\text{Precision} = \frac{TP}{TP + FP}$$
  High precision indicates that the model has a low false positive rate.

- **Recall**:
  Recall, also referred to as Sensitivity or the True Positive Rate, evaluates the model's effectiveness in accurately detecting all relevant positive instances within the dataset (true positives). It is calculated as:
  $$\text{Recall} = \frac{TP}{TP + FN}$$
  High recall means that the model is effectively capturing most of the actual positive instances.

- **F1-score**:
  The F1-score represents the harmonic mean of precision and recall, offering a single metric that balances the trade-off between the two, especially useful when the class distribution is imbalanced. The formula for the F1-score is:
  $$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
  The F1-score ranges from 0 to 1, with 1 indicating perfect precision and recall.

- **Brier Score**:
  The Brier Score measures the accuracy of probabilistic predictions. It is the mean squared difference between the predicted probabilities assigned to the possible outcomes (goal or no goal) and the actual outcomes. The formula is:
  $$\text{Brier Score} = \frac{1}{N} \sum_{i=1}^{N} (f_i - o_i)^2$$
  where:
  - N = Number of predictions
  - f_i = Predicted probability for instance i
  - o_i = Actual outcome for instance i (1 if goal, 0 otherwise)

  A lower Brier Score indicates better model calibration [4].

- **Log Loss**:
  Log Loss (Logarithmic Loss) evaluates the performance of a classification model where the predicted output is a probability value between 0 and 1. It penalizes the model for making predictions that are far from the actual class labels. The formula for Log Loss is:
  $$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^{N} [o_i \log(f_i) + (1 - o_i) \log(1 - f_i)]$$
  where:
  - N = Number of predictions
  - f_i = Predicted probability for instance i
  - o_i = Actual outcome for instance i (1 if goal, 0 otherwise)

Lower Log Loss values indicate better model performance [8].

- **ROC-AUC**:
  ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is a metric used to evaluate the performance of classification models across different threshold levels. ROC is a probability curve, and AUC represents the degree or measure of separability between the classes (goal vs. no goal). The formula for AUC is complex and generally computed numerically, but conceptually:

  $$\text{AUC} = \text{Area Under the ROC Curve}$$

  AUC values range from 0 to 1, with a value closer to 1 indicating a model that is better at distinguishing between positive and negative classes.

- **Confusion Matrix**:
  The Confusion Matrix is a summary of prediction results on a classification problem. It is a 2x2 table that compares the predicted labels with the actual labels. The Confusion Matrix offers a comprehensive overview of the correct and incorrect predictions made by the model, allowing for more detailed information about its performance. The matrix is structured as follows:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

  where:

  - TP = True Positives: Correctly predicted positive cases.
  - TN = True Negatives: Correctly predicted negative cases.
  - FP = False Positives: Incorrectly predicted positive cases.
  - FN = False Negatives: Incorrectly predicted negative cases.

  The Confusion Matrix allows us to calculate several important metrics, such as precision, recall, and accuracy, by examining the counts of true positives, true negatives, false positives, and false negatives.

5.1.2 Model Selection

In this study, three traditional machine learning models are selected to predict whether a sequence of passes leads to a goal: Logistic Regression, Random Forest, and Support Vector Machine (SVM). These models are chosen based on their specific strengths, collectively providing a robust framework for binary classification tasks. Below is an overview of each model, including the rationale for their selection and the approach used for hyperparameter tuning.

5.1.2.1 Logistic Regression

Model Overview: Logistic regression is a basic algorithm used for binary classification. It relies on a linear relationship between the input features and the log-odds of the binary outcome. By using the logistic function, it transforms the output of the linear equation into a probability value ranging from 0 to 1, which helps in classifying outcomes.

Why Chosen: The reason that Logistic Regression was chosen is because of its simplicity which makes it an ideal choice for understanding how individual features influence the predicted outcome. Its simplicity makes it a reliable baseline model for binary classification.

Model Parameters and Hyperparameter Tuning: The primary hyperparameter for Logistic Regression is the regularization strength (C), which controls the trade-off between maximizing the likelihood and minimizing the model complexity. The selection of optimal hyperparameters is important for enhancing the performance of machine learning models. In this study, a detailed hyperparameter tuning process is used, using grid search combined with 10-fold cross-validation. Grid search involves systematically searching through a predefined set of hyperparameter values to find the optimal combination, while cross-validation divides the dataset into multiple folds to ensure that the model's performance generalizes well. This approach systematically explores a range of hyperparameter values to identify the best configuration for each model, balancing accuracy, precision, recall, and F1-score.

For the Logistic Regression model, the following hyperparameters were tuned:

- **C**: This parameter controls the inverse of regularization strength. A smaller value specifies stronger regularization. The grid search explored values including 1, 10 and 100.
- **class_weight**: To address class imbalance, the class weights are set to `balanced`, ensuring that the model paid equal attention to both classes despite the imbalance in the dataset.
- **max_iter**: This parameter defines the maximum number of iterations for the solver to converge. The value is set to 1000 to guarantee complete convergence.
- **solver**: `liblinear` was selected as the solver for the optimization problem due to its efficiency in handling smaller datasets and compatibility with L1 regularization.

The final tuned model uses the following configuration after grid search:

```
LogisticRegression(C=100, class_weight='balanced',
max_iter=1000, solver='liblinear')
```

This configuration was selected based on its superior performance during cross-validation, where it consistently provided a good balance between precision, recall, and overall accuracy across different folds of the data.

5.1.2.2 Random Forest

Model Overview:
Random Forest is an ensemble learning technique that builds multiple decision trees and combines their outputs to make a final prediction. For classification tasks, the model predicts the class that is most frequently chosen by the individual trees.

Why Chosen:
Random Forest was chosen for its robustness and ability to effectively handle large feature sets. Its ensemble nature helps to reduce overfitting, making it particularly useful for capturing complex patterns in the data.

Model Parameters and Hyperparameter Tuning:

Once again, we used grid search combined with 10-fold cross-validation to systematically explore and identify the best set of hyperparameters for each model.

For the Random Forest model, the following hyperparameters were tuned:

- **n_estimators**: This parameter represents the number of trees in the forest. A higher number generally improves performance but also increases computational cost. The grid search explored values such as 50, 100, and 200.
- **max_depth**: This parameter controls the maximum depth of each tree. Limiting the depth can prevent overfitting, especially in datasets with noisy data. Values such as 10, 20, and 30 were tested.
- **class_weight**: Once again, to address the issue of class imbalance, the class weights are set to `balanced`. This automatically adjusts the weights inversely proportional to the class frequencies, ensuring that the model is not biased towards the majority class.

The final tuned model has the following configuration:

```
RandomForestClassifier(class_weight='balanced',
max_depth=30, n_estimators=200)
```

5.1.2.3 Support Vector Machine (SVM)

Model Overview: Support Vector Machines (SVM) are robust classification models that work by identifying the optimal hyperplane that separates different classes in the feature space.

Why Chosen: SVM was chosen for its strong performance in high-dimensional spaces and its flexibility in handling complex decision boundaries through the use of different kernel functions, such as linear and radial basis functions.

Model Parameters and Hyperparameters Tuning:

This time, because of computational cost, we employed grid search combined with only 5-fold cross-validation to explore and identify the optimal set of hyperparameters for the Support Vector Machine (SVM) model.

For the SVM model, the following hyperparameters were tuned:

- **C**: The regularization parameter (`C`) controls the trade-off between minimizing the training error and reducing the model complexity. The hyperparameter tuning process, conducted via cross-validation, explored values of `C` such as {0.1, 1 and 10}. A value of `C=10` was identified as optimal, effectively balancing underfitting and overfitting in the model.
- **class_weight**: Set to `balanced` to account for class imbalance in the dataset. This adjustment scales the cost function to ensure that the model does not favor the majority class excessively.
- **gamma**: This parameter determines the extent to which a single training example influences the model, with the `scale` option chosen for its ability to adjust gamma based on the number of features.
- **kernel**: The type of kernel used in the SVM algorithm. The `rbf` (Radial Basis Function) kernel was selected, as it allows the model to handle non-linear relationships within the data effectively.

The final SVM configuration determined through grid search and cross-validation are:

```
SVC(C=10, class_weight='balanced', gamma='scale', kernel='rbf')
```

5.1.3 Analysis of Results

In this section, we evaluate the performance of three traditional machine learning models—Logistic Regression, Random Forest, and Support Vector Machines (SVM). The evaluation is based on various performance metrics including accuracy, precision, recall, F1-score, Brier Score, and Log Loss. Additionally, we analyze the impact of adjusting the classification threshold based on the ROC curve to optimize the trade-off between true positive and false positive rates.

Table 5.1 shows the ROC-AUC across all leagues:

Model	ROC-AUC
Logistic Regression	0.97
Random Forest	0.97
SVM	0.96

Table 5.1: ROC-AUC Scores for Each Model Across All Leagues

The results for each model across the different leagues, including the optimal threshold analysis, are summarized in the tables 5.2, 5.3 and 5.4 below:

Dataset	Accuracy	Precision	Recall	F1-score	Brier Score	Log Loss
La Liga	0.8717	0.0905	0.931	0.1650	0.0743	0.2342
Serie A	0.8378	0.0649	0.992	0.1218	0.0691	0.2208
Premier League	0.8919	0.1054	0.932	0.1893	0.0759	0.2430
Ligue 1	0.8754	0.0822	0.941	0.1512	0.0737	0.2303
Bundesliga	0.9040	0.1179	0.937	0.2095	0.0586	0.1952
All Leagues	0.8686	0.0850	0.955	0.1561	0.0748	0.2302

Table 5.2: Performance Metrics for Logistic Regression Across Different Leagues (Optimal Threshold)

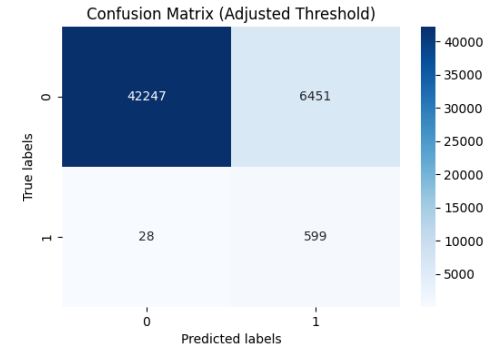
Dataset	Accuracy	Precision	Recall	F1-score	Brier Score	Log Loss
La Liga	0.8903	0.1026	0.910	0.1844	0.0132	0.0642
Serie A	0.9303	0.1269	0.876	0.2218	0.0138	0.0624
Premier League	0.9192	0.1364	0.932	0.2380	0.0137	0.0597
Ligue 1	0.9616	0.2220	0.899	0.3561	0.0125	0.0555
Bundesliga	0.9378	0.1661	0.892	0.2801	0.0120	0.0534
All Leagues	0.9450	0.1750	0.895	0.2927	0.0112	0.0500

Table 5.3: Performance Metrics for Random Forest Across Different Leagues (Optimal Threshold)

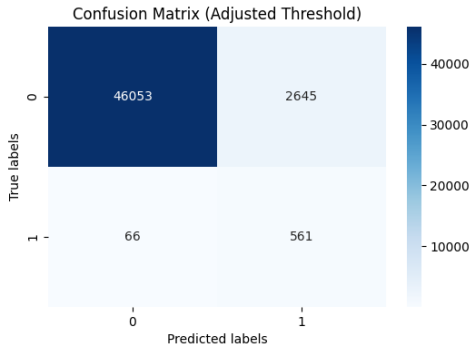
Dataset	Accuracy	Precision	Recall	F1-score	Brier Score	Log Loss
La Liga	0.979	0.325	0.504	0.395	0.011	0.042
Serie A	0.980	0.275	0.463	0.345	0.009	0.036
Premier League	0.979	0.333	0.505	0.401	0.011	0.041
Ligue 1	0.982	0.346	0.555	0.426	0.009	0.036
Bundesliga	0.980	0.333	0.505	0.401	0.011	0.041
All Leagues	0.979	0.325	0.504	0.395	0.011	0.042

Table 5.4: Performance Metrics for SVM Across Different Leagues (Optimal Threshold)

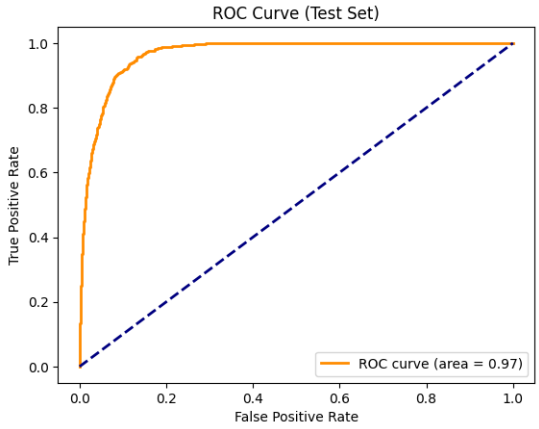
Across all leagues, in the Logistic Regression model, while the accuracy is generally high, the confusion matrices reveal a considerable number of false positives (Figure 5.1a). This indicates that the model, despite being conservative in its positive predictions, still misclassified many non-goal sequences as goals even after adjusting the threshold based on the ROC curve (Figure 5.1c). In contrast, the Random Forest model shows a better balance between precision and recall, especially after adjusting the threshold based on the ROC curve (Figure 5.1b and Figure 5.1d).



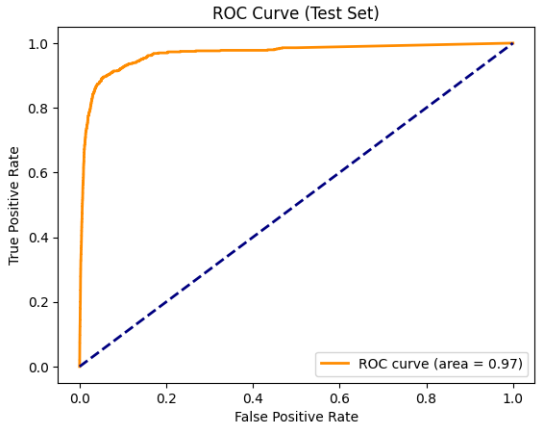
(a) Logistic Regression - Confusion Matrix



(b) Random Forest - Confusion Matrix



(c) Logistic Regression - ROC Curve



(d) Random Forest - ROC Curve

Figure 5.1: Confusion Matrices and ROC Curves for Logistic Regression and Random Forest Across All Leagues

5.1.3.1 Feature Importance

We analyze the feature importance using the coefficients from the Logistic Regression model and the feature importance scores from the Random Forest model. These analyses help to identify which features significantly contribute to predicting the outcome of goal-scoring sequences.

In the **Logistic Regression model**, the importance of features is derived from the absolute value of the model coefficients. The most influential features identified by the model are shown in the table 5.5:

Feature	Importance
shot_assist	8.629
O_Completed	6.174
O_Out	5.395

Table 5.5: Top Features in Logistic Regression Model

These results indicate that `shot_assist` is the most critical feature, suggesting that passes leading directly to shots, as expected, significantly influence the likelihood of a sequence resulting in a goal. The `O_Completed` and `O_Out` features, which relate to the completion status of a pass, also play vital roles.

In the **Random Forest model**, feature importance is determined by the mean decrease in impurity (Gini importance). The table 5.6 shows the top three features identified by the Random Forest model:

Feature	Importance
O_Completed	0.138
x_end_location	0.135
x_location	0.104

Table 5.6: Top Features in Random Forest Model

The `O_Completed` feature is again highlighted as significant, reinforcing its importance in predicting goal-scoring sequences. Additionally, spatial features such as `x_end_location` and `x_location` are also relevant, indicating that the position on the field where a pass is made or received can be predictive of a successful outcome.

5.1.4 Summary of Findings

This section evaluates the performance of the traditional machine learning models used —Logistic Regression, Random Forest, and Support Vector Machines (SVM)— on a dataset of passing sequences from top European football leagues. Key performance metrics such as accuracy, precision, recall, F1-score, Brier Score, Log Loss and ROC-AUC are used to evaluate the models. Optimal classification thresholds are also examined using ROC curves to refine the balance between true positive and false positive rates.

- **Logistic Regression**:
  - Logistic Regression demonstrates decent performance across all leagues, with accuracy ranging from 83.78% (Serie A) to 90.40% (Bundesliga). However, it struggles with precision, especially in predicting goal-scoring sequences, as evidenced by the generally low precision scores (ranging from 0.0649 to 0.1179).
  - The recall is consistently high across all leagues, indicating that the model is able to identify most goal-scoring sequences but at the expense of a high false positive rate. The F1-scores are consequently low, reflecting the trade-off between precision and recall.
  - The model's Brier Scores and Log Loss values suggest that while Logistic Regression is reasonably calibrated, it is less effective in minimizing prediction uncertainty compared to more complex models like Random Forest.

- **Random Forest**:
  - Random Forest outperforms Logistic Regression in almost every metric, particularly in precision and F1-score. The model achieves accuracy levels as high as 96.16% (Ligue 1) and demonstrates strong performance across all other leagues.
  - The model's ability to balance precision and recall, especially in leagues like Ligue 1 and the Bundesliga, indicates its robustness in predicting goal-scoring sequences.
  - Random Forest's Brier Scores and Log Loss values are significantly lower than those of Logistic Regression, reflecting better-calibrated probability estimates and lower uncertainty in predictions. This model is particularly successful in utilizing the complexity of the feature space to improve predictive performance.
- **Support Vector Machine (SVM)**:
  - SVM exhibits strong performance across all leagues, with accuracy exceeding 97% in each case. The model is able to achieve a better balance between precision and recall compared to Logistic Regression, with F1-scores ranging from 0.345 (Serie A) to 0.426 (Ligue 1).
  - Although SVM's precision and recall are slightly lower than those of Random Forest, its performance is consistent across different datasets, making it a reliable choice for this task.
  - The model's Brier Scores and Log Loss values are competitive with those of Random Forest, indicating that SVM is also effective in providing well-calibrated probability estimates.

In summary, Random Forest emerges as the best-performing model across all metrics, particularly in its ability to manage the trade-off between precision and recall. Logistic Regression, while simpler and more interpretable, is less effective, particularly in precision. SVM provides a strong balance of performance, consistently delivering high accuracy with moderate precision and recall. These findings highlight the importance of model selection based on the specific objectives and constraints of the predictive task.

5.1.5 Limitations

While traditional machine learning models proved effective in capturing the general patterns within the dataset, they exhibit certain limitations, particularly in their ability to handle more complex, non-linear relationships inherent in football data. For example:

- **Feature Interactions**: Traditional models like Logistic Regression struggle to capture interactions between features unless explicitly modeled. This limitation may result in missed opportunities to uncover deeper meanings from the data.

- **Handling Unstructured Data**: Traditional models are less equipped to handle unstructured data, such as sequences of passes or spatial information, without substantial preprocessing.
- **Imbalanced Data**: Despite using techniques like class weighting and optimal threshold adjustment, these models sometimes struggle with the imbalanced nature of the dataset, particularly when predicting rare events like goals.

5.1.6 More Advanced Models

To overcome the limitations observed with traditional machine learning models and to better capture the intricate patterns within the data, the next sections of this thesis will explore more advanced modeling techniques. Specifically, the focus will shift to Deep Learning models, such as Recurrent Neural Networks (RNNs), which are effective in handling sequential data, and Graph Neural Networks (GNNs), which are well suited for modeling relational data like passing sequences in football.

5.2 Deep Learning - Recurrent Neural Network (LSTM)

5.2.1 Dataset Modifications

To transition from traditional machine learning models to Recurrent Neural Networks (RNNs), the dataset is modified to capture the sequential nature of the data. These changes are essential for RNNs to effectively model temporal dependencies and patterns. The following sections outline the key modifications and the reasons behind them.

5.2.1.1 Sequence Creation

A critical step in preparing the dataset for RNNs involves preparing the sequences from the original dataset. Unlike the previous models, where each sequence was aggregated into a single feature vector, RNNs require the entire sequence of events to be fed into the model.

Rationale for Sequence Length: The length of each sequence is determined by the number of passes within a single possession. Given the variability in sequence lengths, shorter sequences are padded to a uniform length, while excessively long sequences are truncated to maintain computational efficiency. This padding and truncation process ensures that all input sequences have a consistent shape, which is required for batch processing in RNNs.

Padding and Masking: Sequences are padded with zeros to standardize their length. A corresponding mask is generated to differentiate between real data and padded values, allowing the model to focus on the actual sequence data during training. This masking mechanism is vital

in preventing the model from learning patterns in the padding, which would introduce noise and reduce performance. Additionally, the padded values are excluded from gradient calculations, ensuring they do not influence the model's learning process.

5.2.1.2 Feature Representation

Numerical and Categorical Features: The feature representation is adapted to suit the requirements of an RNN model. Numerical features, such as `angle`, `duration`, `length`, and spatial coordinates (`x_location`, `y_location`, etc.), are normalized using a `StandardScaler`. This normalization process ensures that all numerical inputs are on a similar scale, which aids the convergence of the model during training.

Categorical features, previously encoded using one-hot encoding, are retained in their binary vector form. These features are directly fed into the model without scaling, as one-hot encoded variables inherently have a binary range (0 or 1) and do not require further transformation.

Exclusion of Aggregated Features: In contrast to traditional models, where aggregated features like the mean and standard deviation of sequence attributes were used, the RNN model utilizes the raw, unaggregated sequences. This approach allows the RNN to learn temporal patterns within each sequence, capturing the dynamic nature of passing plays in football, by processing each pass individually within a sequence.

5.2.1.3 Data Augmentation

While data augmentation techniques were considered to enhance the diversity of the training data, the primary focus remains on ensuring that the sequences accurately reflected the in-game situations. No synthetic data is added to the sequences, as the inherent variability in football plays provides sufficient diversity for model training.

5.2.1.4 Dataset Splitting and Stratification

To evaluate the model's performance across different competitions, the dataset is split into training, validation and testing subsets, ensuring that each subset maintains the same distribution of sequences across leagues. This stratification is important to prevent the model from being biased toward any particular league. The dataset is split as follows:

- **Training Set**: This set comprises 60% of the data and is used to train the Recurrent Neural Network (RNN), enabling the model to learn and capture temporal patterns and dependencies within the sequences.
- **Validation Set**: Consisting of 20% of the data, the validation set is used to tune hyperparameters and monitor the model during training to prevent overfitting, ensuring

that the model generalizes well to unseen data.

- **Test Set**: The remaining 20% of the data is reserved for the test set, which provides a final, unbiased evaluation of the model's performance on data that is not used during training or validation.

The final dataset that we used to train the RNN consists of 750,360 sequences, with 169 features each, preserving the detailed information from the original passes without any aggregation.

5.2.2 Model Selection

In this section, we discuss the rationale behind selecting Recurrent Neural Networks (RNNs) for the analysis, detail the architecture of the model employed, and describe the training process, including the selection of appropriate hyperparameters and the optimization techniques used.

5.2.2.1 Why RNN?

Recurrent Neural Networks (RNNs) were chosen for this task due to their inherent capability to model and deal appropriately with sequential data. Football passing sequences naturally exhibit sequential characteristics where the outcome of each pass can be influenced by the previous ones. Unlike traditional feedforward neural networks, RNNs can maintain a form of memory through their hidden states, enabling them to process sequences of variable length while capturing the temporal dynamics [17].

RNNs are especially useful for tasks where the order of inputs matters, such as in time series prediction, natural language processing, and, in this case, analyzing sequences of football passes to predict the likelihood of a goal [37].

5.2.2.2 RNN Architecture

The RNN model that we use in this study is based on a LSTM (long-short-term memory) architecture. LSTMs are a type of RNN developed to overcome the issue of vanishing gradients, allowing the model to capture long-term dependencies more effectively [17].

- **Input Layer**: The input to the LSTM consists of sequences of passes, where each pass is represented by a feature vector comprising both numerical and categorical features.
- **LSTM Layers**: The core of the model is composed of multiple LSTM layers, which process the input sequences and produce hidden states that encapsulate the temporal dependencies within the sequence [36]. The number of LSTM layers is treated as a hyperparameter, referred to as `num_layers`, and the number of hidden units per layer is treated as another

hyperparameter, referred to as `hidden_size`. These hyperparameters were explored through grid search.

- **Dropout Layers**: To prevent overfitting, dropout layers are included in the model, with a dropout probability of 0.4 [34]. This technique randomly deactivates a fraction of the neurons during training, helping the model generalize better to unseen data.
- **Fully Connected Layer**: The output of the LSTM layers is passed through a fully connected linear layer, which maps the hidden states to the final output.
- **Sigmoid Activation**: A sigmoid activation function is applied to the output layer to produce a probability score between 0 and 1, suitable for binary classification tasks (goal or no goal) [15].

5.2.2.3 Training Process

The training process for the LSTM model is designed to optimize performance while preventing overfitting. The following components are key to the training process:

Loss Function: The loss function we use is Binary Cross-Entropy with Logits, denoted as `(BCEWithLogitsLoss)`, which combines sigmoid activation with binary cross-entropy loss. This choice is appropriate for binary classification tasks and is robust to imbalances in the dataset when combined with class weights. In this context, class weights are computed to adjust for the imbalance between goal-scoring and non-goal-scoring sequences. This is achieved by calculating the class distribution within the training set and using those values to assign a higher weight to the minority class (goal sequences).

Class Weights and Weighted Sampling: To address the imbalance in the dataset, class weights are calculated based on the frequency of goal and non-goal-scoring sequences. These weights are used to create a `WeightedRandomSampler` for the training process, ensuring that both classes are represented fairly during each training batch. This would allow the model to learn more effectively from the underrepresented class (goal sequences). The class weights are mapped to individual samples, and the sampler ensured that the training data is sampled in a balanced manner.

- **Class Weight Calculation**: The class weights are computed using the formula:

$$\text{Class Weight} = \frac{\text{Total Number of Samples}}{\text{Number of Samples in Class}}.$$

Optimizer: The model is trained using the Adam optimizer, a widely-used optimization algorithm that adapts the learning rate for each parameter, offering a balance between speed and convergence stability. An initial learning rate (`lr`) is set, and a maximum learning rate (`max_lr`) is defined for the learning rate schedule.

Learning Rate Schedule: A dynamic learning rate schedule is implemented, which adjusts the learning rate during training to improve convergence. This schedule helps the model escape local minima and converge to a better solution.

Early Stopping: To prevent overfitting and reduce unnecessary computations, early stopping is employed. If the validation loss did not improve for a specified number of epochs (`patience = 10`), training is stopped and the model parameters of the best epoch are restored.

Grid Search: A grid search was conducted to explore various combinations of hyperparameters, including the number of hidden units (`hidden_size`), the number of LSTM layers (`num_layers`), and the learning rate (`lr`). The search explored values such as `hidden_size` ranging from 64 to 256, `num_layers` between 1 and 3, and learning rates of 0.0001 and 0.001. The goal was to identify the hyperparameter configuration that minimized validation loss. The best configuration consists of `hidden_size=64`, `num_layers=1`, `batch_size=32`, `learning_rate=0.001`, and `max_lr=0.1`.

Training and Validation Loss Analysis:

Figure 5.2 illustrates the training and validation loss curves of the Recurrent Neural Network (RNN) over several epochs. The training loss (blue curve) decreases in the early epochs, stabilizing around a value of 0.70, suggesting that the model learns some of the patterns in the training data.

Similarly, the validation loss (orange curve) decreases sharply during the initial epochs, but it remains higher than the training loss throughout the process. This discrepancy between the two curves points to a degree of overfitting, where the model performs better on the training data compared to the validation data.

After approximately 10 epochs, both losses stabilize, with the validation loss remaining relatively consistent, although higher than the training loss. This indicates that the model may generalize reasonably well but could maybe benefit from additional regularization techniques.

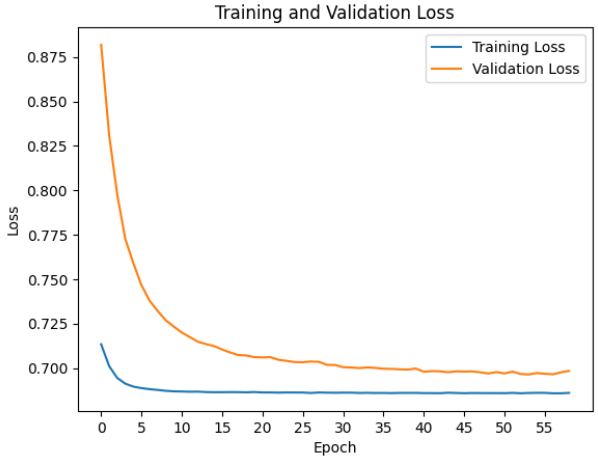


Figure 5.2: Training and Validation Loss over epochs for the RNN model

5.2.3 Analysis of Results

5.2.3.1 Performance Metrics

The performance of the Recurrent Neural Network (RNN) model is evaluated using several metrics, including accuracy, precision, recall, F1-score, Brier Score, and Log Loss. These metrics are compared against those obtained from the traditional machine learning models discussed earlier. The performance of the RNN is assessed both before and after training, as well as after adjusting the classification threshold based on the ROC curve.

The following tables 5.7 and 5.8 summarize the key metrics:

Metric	Value
Accuracy	0.012
Precision	0.012
Recall	1.0
F1-score	0.024
Brier Score	0.390
Log Loss	0.980

Table 5.7: RNN Performance Metrics Before Training

These metrics indicate that the RNN, prior to training, is heavily biased towards predicting every sequence as leading to a goal. This is reflected in the perfect recall but extremely poor precision and accuracy, signaling the model's inability to differentiate between goal-scoring and non-goal-scoring sequences effectively.

Metric	Value
Accuracy	0.0135
Precision	0.0135
Recall	1.0
F1-score	0.0266
Brier Score	0.3870
Log Loss	0.9734

Table 5.8: RNN Performance Metrics After Training

After training, the RNN shows a slight improvement in performance, but the metrics remain far from satisfactory. The model continues to exhibit a significant bias towards predicting every sequence as a goal, as shown by the high recall and very low precision and F1-score (tables 5.8 & 5.9).

Class	Precision	Recall	F1-Score	Support
0.0	0.00	0.00	0.00	133847
1.0	0.01	1.00	0.03	1826
Accuracy			0.01	135673
Macro Avg	0.01	0.50	0.01	135673
Weighted Avg	0.00	0.01	0.00	135673

Table 5.9: Classification Report for RNN Model

Optimal Threshold ROC Curve Results:

Metric	Value
Accuracy	0.0158
Precision	0.0135
Recall	0.9995
F1-score	0.0266
Brier Score	0.2526
Log Loss	0.6984

Table 5.10: RNN Performance Metrics After Adjusting the Threshold (ROC Curve)

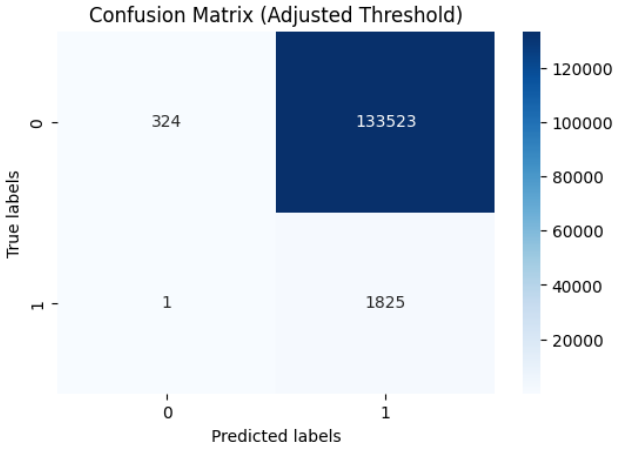


Figure 5.3: Confusion Matrix of the RNN Model with Adjusted Threshold

After adjusting the threshold based on the ROC curve (Table 5.10 and Figure 5.3), the RNN's performance metrics improves slightly in terms of accuracy and log loss. However, the model still exhibits a strong bias towards the positive class, predicting most sequences as goals. This indicates that while threshold adjustment can help, the model's architecture and the features used are not sufficient to capture the complexity of the task effectively.

These results suggest that, although the RNN learns to some extent during training, it still struggles to differentiate between goal-scoring and non-goal-scoring sequences. The persistent low precision and accuracy, along with the unchanged recall, indicate that the model fails to generalize effectively from the training data. This could be attributed to the limited scope of the features used, which only considered passing sequences without incorporating other important aspects of gameplay, such as shooting, dribbling, or defensive actions.

5.2.3.2 Learning Curves

The learning curves for the RNN, showing both the training and validation loss over epochs, reveal that the model experiences difficulty in converging, with the validation loss remaining high throughout the training process. This indicates that the model may have been struggling with overfitting or underfitting issues, or that the complexity of the data is not adequately captured by the chosen architecture.

The persistent high loss in validation suggests that the RNN is unable to generalize well from the training data to unseen data, a critical requirement for robust predictive models.

5.2.3.3 Impact of Sequence Length

The length of the sequences in the dataset also may have a significant impact on the RNN's performance. Longer sequences, while providing more contextual information, increase the complexity of the model's learning process. However, due to the nature of the dataset—focusing exclusively on passing sequences without considering other important attributes such as shooting, dribbling, or defensive actions—the RNN's ability to accurately predict goals remains limited.

This limitation highlights a key challenge: while RNNs are well suited to capturing temporal dependencies in sequence data, the quality and diversity of the input features are crucial for their success. The exclusion of features related to other phases of play (e.g., shooting, defending) likely constrained the RNN's ability to make accurate predictions, as it was unable to consider the full context of each sequence.

5.2.4 Summary of Findings

The Recurrent Neural Network (RNN) model may have the potential to capture temporal dependencies within passing sequences, which is a significant advantage over traditional models. However, the overall performance of the RNN in this study was suboptimal. Despite its theoretical advantages, the model struggles to differentiate effectively between goal-scoring and non-goal-scoring sequences, as reflected in the evaluation metrics. The RNN shows a strong bias toward predicting sequences as goals, leading to high recall but very low precision and accuracy.

5.2.4.1 Comparison with Traditional Models

When comparing the RNN to traditional machine learning models such as Logistic Regression, Random Forest, and Support Vector Machines (SVM), several observations can be made:

1. **Performance Metrics**: Traditional models, particularly Random Forest, outperform the RNN in terms of overall accuracy and precision. The Random Forest model provides a

better balance between true positives and false positives, while the RNN struggles with high false positive rates. This underperformance of the RNN suggests that while temporal dependencies are important, the RNN is not able to fully exploit this information in the context of the features provided.

2. **Model Complexity and Interpretability**: Traditional models, especially Logistic Regression, offer greater interpretability and are easier to tune and deploy. The RNN, although more powerful in theory, is more complex and requires careful tuning and more extensive training data to achieve optimal results.

5.2.4.2 Next Approach

While the RNN model offers unique advantages in modeling sequential data, its performance in this study highlights the challenges associated with training deep learning models on limited or domain-specific datasets. The underwhelming results from the RNN suggest that there may be additional, more complex relationships within the data that are not being fully captured by either traditional models or RNNs.

This observation leads to the exploration of Graph Neural Networks (GNNs) in the next section. GNNs are particularly well suited for structured data, such as networks or graphs, where the relationships between data points (e.g., passes between players) are as important as the data points themselves. GNNs have the potential to model these intricate relationships more effectively, providing a more nuanced understanding of the factors that contribute to goal-scoring opportunities.

5.3 Deep Learning - Graph Neural Network

5.3.1 Dataset Modifications

This section outlines the modifications made to prepare the dataset for use with Graph Neural Networks (GNNs). Converting traditional tabular data into graph-structured data is necessary to enable GNNs' capability to model complex relationships between entities.

Graph Construction: To utilize a GNN, the dataset is transformed into a series of graphs where each graph represents a sequence of passes within a football match. The nodes on these graphs correspond to the players involved in the passing sequence, while the edges represent the passes between players. Each node is characterized by features that encode the player's identity and their position on the pitch, while edges are associated with features that describe the characteristics of each pass.

The process of constructing these graphs involves several steps:

- **Node Definition**: Each node in the graph represents a unique combination of a player and their specific location on the pitch, which is divided into 18 predefined sections (6 along the width and 3 along the height). The section number is determined based on the player's coordinates during a pass (Figure 5.4). Players are assigned numerical identifiers (1–15) for each match, ensuring consistency in representation. For each team, up to 14 players are numbered according to their roles: 11 from the starting lineup and 3 substitutes. The 15th identifier is reserved for "unknown players"—those whose identities are unspecified in the data. This approach ensures that both individual players and their spatial positions are distinctly captured for analysis.
- **Edge Definition**: Edges are defined between nodes where a pass occurred from one player to another. The features associated with each edge include various pass characteristics such as duration, length, whether the pass was under pressure and whether it was a cross, etc. Edge features play an important role in capturing the nuances of interactions between players.

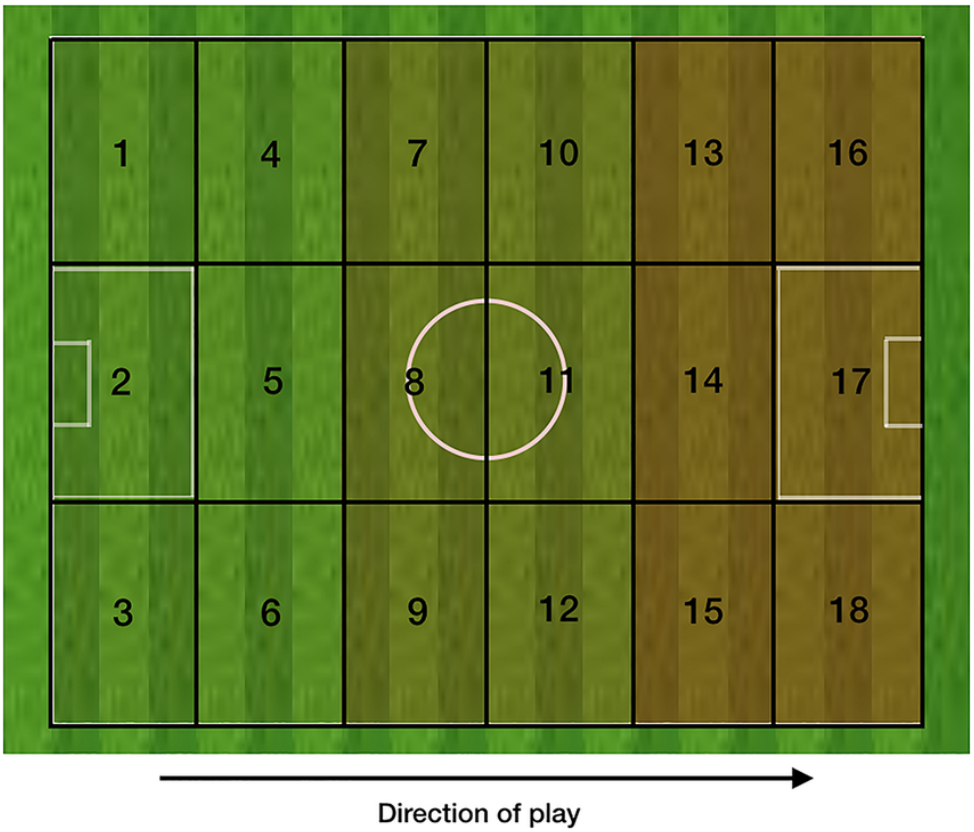


Figure 5.4: The pitch divided into 18 zones, source: [19].

Handling Imbalance: For the GNN, class weights are incorporated into the loss function, and stratified sampling is used when splitting the data to maintain a representative distribution of classes across training, validation, and testing sets.

Dataset Splitting for GNN Training: As with the RNN model, the dataset for training

the Graph Neural Network (GNN) model is divided into three separate sets: training, validation, and testing. This procedure guarantees that the model can generalize properly and avoids overfitting to the training data. The dataset is split with 60% assigned for training, 20% designated for validation, and 20% allocated for testing.

5.3.2 Model Selection

In this section, we discuss the rationale for choosing the graph neural network (GNN) as the model for this task. We also describe the specific GNN architecture used and outline the training process employed to optimize the model.

5.3.2.1 Why GNN?

Graph Neural Networks (GNNs) were chosen for their ability to model complex relationships in graph-structured data. This is especially useful in football analytics, where player interactions through passing can be modeled as a graph. GNNs learn representations that respect the graph structure, helping the model to understand each pass within the broader context of the game.

5.3.2.2 GNN Architecture

The architecture of the GNN model in this study is designed to use both node and edge features effectively. The model consists of the following components:

- **Convolutional Layers**: The GNN uses two **NNConv (Neural Network Convolution)** layers, which are designed to incorporate both node and edge features into the convolution process [12, 13, 33]. The first convolutional layer (conv1) maps input node and edge features to a higher-dimensional space, while the second layer (conv2) refines these features. Dropout layers are applied after each convolution to prevent overfitting.
- **Global Pooling**: After the convolutions, a global mean pooling operation aggregates the node features across the graph, reducing the variable-size node set into a fixed-size representation suitable for the next stage.
- **Final MLP**: The pooled graph representation is fed into a Multi-Layer Perceptron (MLP), which consists of a hidden layer with ReLU activation, a dropout layer, and a final linear layer to predict the probability of a goal.

5.3.2.3 Training Process

The training process for the GNN involves several key steps and considerations specific to graph data:

- **Loss Function**: The model is trained using the Binary Cross-Entropy with Logits Loss function (`BCEWithLogitsLoss`), which is well suited for binary classification tasks. Given the class imbalance in the dataset (more non-goal sequences than goal sequences), the loss function is weighted to penalize errors in predicting the minority class more heavily.
- **Optimizer**: The Adam optimizer is employed due to its adaptive learning rate capabilities, which are beneficial in complex models like GNNs. The learning rate and other hyperparameters were fine-tuned using a grid search.
- **Early Stopping**: To prevent overfitting, early stopping is implemented. The model's performance on the validation set is tracked, and training is stopped if there is no improvement in validation loss for a specified number of epochs (patience = 10). This ensures that the model generalizes well to unseen data.
- **Hyperparameter Tuning**: A grid search was conducted to identify the best combination of hidden layer dimensions, learning rates, and batch sizes. The search explored values such as hidden layer sizes of 16, 32 and 64, learning rates of 0.001 and 0.0001, and batch sizes of 16 and 32. This process was essential for determining an optimal model architecture that balances complexity with generalization ability. The best-performing model achieved a validation loss of 0.0915, with a hidden layer dimension of 64, a learning rate of 0.001, and a batch size of 32.

The table 5.11 below summarizes the best hyperparameters identified during the grid search:

Hyperparameter	Value
Hidden Dimension	64
Learning Rate	0.001
Batch Size	32

Table 5.11: Best Hyperparameters for GNN Model

5.3.3 Analysis of Results

Performance Metrics: The performance of the Graph Neural Network (GNN) model is evaluated using several key metrics, including accuracy, precision, recall, F1-score, Brier Score, Log Loss and ROC AUC. These metrics are compared to those obtained from traditional machine learning models and the Recurrent Neural Network (RNN). The results before and after training are summarized below.

Before Training: The initial performance of the GNN before training is largely ineffective, as expected. The model defaulted to predicting the majority class (non-goal sequences), leading to the following metrics shown in table 5.12:

Metric	Value
Accuracy	0.99
Precision	0.00
Recall	0.00
F1-score	0.00
Brier Score	0.19
Log Loss	0.57
ROC AUC	0.53

Table 5.12: GNN Performance Metrics Before Training

The table 5.13 further illustrates the model's initial bias:

Class	Precision	Recall	F1-Score	Support
0	0.99	1.00	0.99	51403
1	0.00	0.00	0.00	574
Accuracy			0.99	51977
Macro Avg	0.49	0.50	0.50	51977
Weighted Avg	0.98	0.99	0.98	51977

Table 5.13: Classification Report for GNN Model

These metrics indicate that the untrained GNN is unable to correctly classify any of the positive cases (goal sequences), leading to perfect recall for the majority class but failing entirely for the minority class.

After Training: Post-training, the GNN demonstrates significant improvement, particularly in its ability to identify goal-scoring sequences. The following table 5.14 summarizes the key metrics after training:

Metric	Value
Accuracy	0.93
Precision	0.12
Recall	0.84
F1-score	0.21
Brier Score	0.05
Log Loss	0.17
ROC AUC	0.94

Table 5.14: GNN Performance Metrics After Training

The confusion matrix (Figure 5.5), ROC curve (Figure 5.6) after training and the table 5.15 provide some more information about the model's performance:

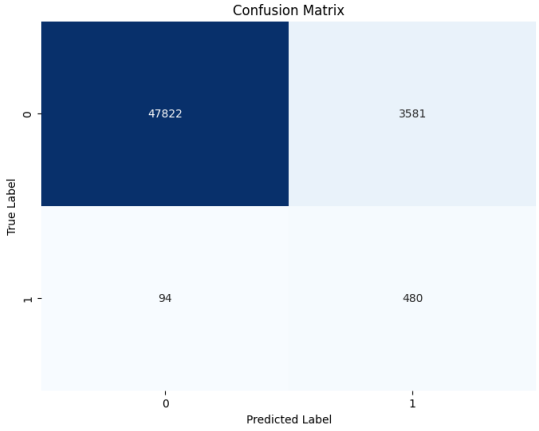


Figure 5.5: Confusion Matrix After GNN Training

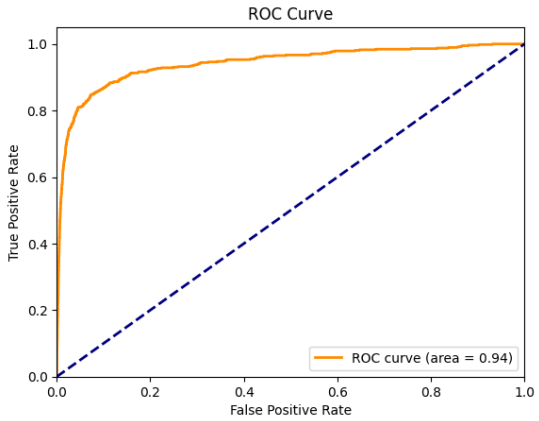


Figure 5.6: ROC Curve After GNN Training

Class	Precision	Recall	F1-Score	Support
0	1.00	0.93	0.96	51403
1	0.12	0.84	0.21	574
Accuracy			0.93	51977
Macro Avg	0.56	0.88	0.59	51977
Weighted Avg	0.99	0.93	0.95	51977

Table 5.15: Classification Report After GNN Training

These results indicate that the GNN is able to significantly reduce both the Brier Score and Log Loss, reflecting a more confident and accurate prediction model. The ROC AUC score of 0.94 highlights the model's ability to distinguish to some extent between goal-scoring and non-goal-scoring sequences, outperforming the RNN and performing similarly to the traditional models.

5.3.4 Summary of Findings

The Graph Neural Network (GNN) demonstrates interesting performance in capturing the complex relationships associated with the task of predicting goal-scoring sequences in football. Unlike traditional machine learning models and even Recurrent Neural Networks (RNNs), the GNN is able to model the interactions between players and passes, taking into account not only individual features but also the structural relationships between nodes (players) and edges (passes). This ability to model both the features and the relationships within the data allows the GNN to outperform some other models, particularly in distinguishing goal-scoring sequences from non-goal-scoring ones.

Comparison with RNN and Traditional Models: The performance of the GNN is better than that of the RNN, but slightly lower than the traditional machine learning models.

The RNN, which performs the worst among the models, struggles to capture the nuances of sequential dependencies in the data, failing to make accurate predictions, as evidenced by its poor precision and recall metrics. Traditional models, while simpler and unable to fully capture the relational structure of the data, still outperform GNNs in terms of predictive performance. Despite their limitations in modeling the complex interactions between players and passes, these models are able to make more accurate predictions for goal-scoring sequences in this context.

In contrast, the GNN utilize the graph structure of the data, where each pass and player is treated as part of a larger, interconnected network. This approach probably allows the GNN to identify key passes and influential player positions that contributed significantly to goal-scoring sequences, resulting in a high recall and overall accuracy. The GNN's ability to model these complex relationships results in an interesting predictive performance.

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
GNN	0.93	0.12	0.84	0.21	0.94
RNN	0.0135	0.0135	1.0	0.0266	0.70
Random Forest	0.945	0.175	0.895	0.2927	0.97

Table 5.16: Comparative Performance of GNN, RNN, and Traditional Models

As shown in the table 5.16, the Random Forest model outperforms both the GNN and RNN across most metrics, particularly in precision, F1-score, and ROC AUC, highlighting its ability to balance between identifying true positives (goal-scoring sequences) and minimizing false positives. However, the GNN still demonstrats strong performance in recall and ROC AUC, emphasizing its capability to correctly identify goal-scoring sequences, although with a lower precision compared to the Random Forest model.

Final Thoughts: The application of GNNs to the problem of predicting goal-scoring sequences in football has proven to be somewhat effective, particularly due to their ability to capture and model the complex relationships between players and passes. While GNNs offer significant advantages in modeling complex interactions, traditional models like Random Forests can sometimes achieve better performance, particularly when hyperparameter tuning and architecture optimization for GNNs are limited. Traditional machine learning models tend to be simpler and require less fine-tuning, which can make it easier to achieve good results quickly. However, with more extensive tuning and optimization, or with larger datasets, GNNs have the potential to outperform traditional models by capturing deeper relational patterns within the data.

Chapter 6

Conclusion

This thesis aimed to investigate passing sequences and their role in goal prediction throughout football matches, using data analytics techniques together with machine learning models. The study focused on an in-depth analysis of passing patterns and sequences while developing predictive models to present a view of the complicated dynamics of the game.

We explored various machine learning approaches to predict goal-scoring sequences in football, using a dataset of passing events from Europe's top five football leagues (season 2015-2016). Through a comprehensive analysis of traditional machine learning models, Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs), we tried to understand how each model performs in predicting whether a sequence of passes results in a goal.

We found that traditional models like Logistic Regression and Random Forest outperformed GNNs, while GNNs, in turn, performed better than RNNs. The RNN struggled to produce strong results. This performance limitation was not exclusive to the RNN but was shared by all models due to the dataset focusing only on passing sequences and excluding other critical factors such as shooting, dribbling, or defensive actions. As a result, all models struggled to deliver strong predictive performance. Additionally, another factor affecting the results is the architecture of the deep learning models. The RNN's architecture, in particular, may not have been tuned to its optimal configuration, which further contributed to its weaker performance. This highlights the importance of hyperparameter tuning and model architecture when using deep learning techniques.

From the beginning, we understood that this was a difficult problem with no guarantee of success. As highlighted before, predicting goals based solely on passing sequences is inherently challenging, given that arguably the most discriminative event, the shot, is missing. Despite this, the results show that our models were still able to learn some patterns from the passing sequences. This suggests that, even with limited information, passing sequences contain important details that can contribute to goal prediction models, such as Expected Goals (xG). Although the task is challenging, the models' ability to demonstrate predictive power indicates that passing data can be valuable, especially when combined with additional features in future research.

6.1 Limitations and Future Work

Future research could expand on this work by exploring the following directions:

- Integrating additional data sources, such as player tracking data or physical performance metrics, defensive aspects, team formations, dribbling, and shot characteristics to further refine the predictive models. Also, using a similar approach with other team sports could offer fresh perspectives into how team-based games work.
- Exploring alternative machine learning models, including other deep learning architectures, that could capture more complex patterns in football data.
- Extending the analysis to other football leagues and competitions and also different seasons to confirm that the findings may apply more broadly.

6.2 Closing Remarks

In conclusion, the main objectives of this thesis have been achieved; however, further exploratory analysis may always uncover additional findings. Throughout this process, I have gained a deeper understanding of machine learning, particularly how various models handle complex, real-world data.

Appendix A

Full Description of the Dataset

The first structure, named *passes_sequences*, includes detailed information regarding each passing sequence derived from the event data. This structure has multiple columns, each serving to delineate the specific attributes of the passing events.

- *sequence_id*: A unique identifier for each passing sequence.
- *pass_id*: An identifier unique to each pass within a sequence.
- *aerial_won*: A boolean value indicating whether the pass was won aerially.
- *angle*: The angle of the pass in radians.
- *assisted_shot_id*: The unique identifier for the shot that the pass assisted.
- *body_part*: Specifies the body part used to execute the pass.
- *cross*: Indicates whether the pass was a cross.
- *duration*: Represents the duration of the pass in seconds.
- *end_location*: An array denoting the x and y coordinates at which the pass ended.
- *goal_assist*: A boolean value indicating whether the pass resulted in an assist to a goal.
- *height*: Specifies the height at which the pass was made.
- *id*: The unique identifier for the pass event.
- *length*: Denotes the length of the pass in yards.
- *location*: An array representing the x and y coordinates at which the pass commenced.
- *match_id*: Identifies the match to which the pass event belongs.
- *minute*: Indicates the minute during which the pass event occurred.

- *off_camera*: A boolean value indicating whether the pass event occurred off-camera.
- *outcome_y*: Represents the outcome of the pass event.
- *period*: Specifies the period of the match in which the pass event occurred.
- *play_pattern*: Indicates the play pattern associated with the pass.
- *player*: The name of the player executing the pass.
- *player_id*: The identifier of the player executing the pass.
- *position*: Indicates the position of the player executing the pass.
- *possession*: Represents the unique possession number within the game.
- *possession_team_id*: The identifier of the team in possession during the pass.
- *recipient*: The name of the player receiving the pass.
- *related_events*: A comma-separated list of identifiers for events related to the pass.
- *second*: Denotes the second at which the pass event occurred.
- *shot_assist*: Indicates whether the pass was an assist for a shot.
- *switch*: A boolean value indicating whether the pass constituted a switch of flank.
- *technique*: Specifies the technique employed in executing the pass.
- *timestamp*: Represents the timestamp of the pass event.
- *type*: Denotes the type of pass event.
- *under_pressure*: A boolean value indicating whether the pass event occurred under pressure.

The second structure, named *sequences_outcome*, includes only two columns:

- *sequence_id*: A unique identifier for each passing sequence.
- *outcome_x*: A Boolean value indicating whether the passing sequence resulted in a goal (TRUE) or not (FALSE).

Bibliography

[1] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[4] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[5] Shun Cao. Passing path predicts shooting outcome in football. *Scientific Reports*, 13, October 2023. doi:10.1038/s41598-024-60183-7.

[6] Paolo Cintia, Salvatore Rinzivillo, and Luca Pappalardo. A network-based approach to evaluate the performance of football teams. 09 2015.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[8] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[9] Tom Decroos, Jan Van Haaren, and Jesse Davis. Actions speak louder than goals: Valuing player actions in soccer. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1851–1861, 2019.

[10] Harm Eggels, Ruud van Elk, and Mykola Pechenizkiy. Explaining soccer match outcomes with goal scoring opportunities predictive analytics. In *MLSA@PKDD/ECML*, 2016.

[11] Enes Eryarsoy and Dursun Delen. Predicting the outcome of a football game: A comparative analysis of single and ensemble analytics methods. In *Hawaii International Conference on System Sciences*, 2019.

[12] PyTorch Geometric. torch_geometric.nn.conv.nnconv, 2021.

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[14] Bruno Gonçalves, Diogo Coutinho, Sérgio Santos, Carlos Lago-Peñas, Saúl Jiménez, and Jaime Sampaio. Exploring team passing networks and player movement dynamics in youth association football. *Journal of Sports Sciences*, 35(2):174–181, 2017.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[16] Mat Herold, Floris Goes-Smit, Stephan Nopp, Pascal Bauer, Chris Thompson, and Tim Meyer. Machine learning in men's professional football: Current applications and future directions for improving attacking play. *International Journal of Sports Science & Coaching*, 14:849–858, October 2019. doi:10.1177/1747954119879350.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[18] David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, New York, NY, USA, 3rd edition, 2013.

[19] Jongwon Kim, Nic James, Nimai Parmar, Besim Ali, and Goran Vučković. The attacking process in football: A taxonomy for classifying how teams create goal scoring opportunities using a case study of crystal palace fc. *Frontiers in Psychology*, 10:1–8, 10 2019. doi:10.3389/fpsyg.2019.02202.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Proceedings of ICLR 2017*, 2017.

[21] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi:10.1038/nature14539.

[22] Hongyou Liu, Will G Hopkins, and Maria-Angeles Gómez. Modelling relationships between match events and match outcome in elite football. *European Journal of Sport Science*, 16 (5):516–526, 2016. doi:10.1080/17461391.2015.1042527.

[23] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990.

[24] Medium. Redes neurais roots 2: Treinamento, 2018.

[25] Tom M Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997.

[26] Fahad Mostafa and Minjun Chen. Computational models for predicting liver toxicity in the deep learning era. *Frontiers in Toxicology*, 5, 2024. ISSN: 2673-3080. doi:10.3389/ftox.2023.1340860.

[27] Javier López Peña and Raúl Sánchez Navarro. Who can replace xavi? a passing motif analysis of football players, 2015.

[28] Aditya Rana. Event detection in football using graph convolutional networks. 01 2023. doi:10.48550/arXiv.2301.10052.

[29] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[30] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[31] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20 (1):61–80, 2009.

[32] Towards Data Science. Mcculloch-pitts model, 2019.

[33] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[35] StatsBomb. Github - statsbomb event data, 2020.

[36] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Interspeech*. ISCA, 2012.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[38] Andrea Tacchetti, H. Francis Song, Pedro A. M. Mediano, Vinicius Zambaldi, Neil C. Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W. Battaglia. Relational forward models for multi-agent learning, 2018.

[39] Ekansh Tiwari, Prasanjit Sardar, and Sarika Jain. Football match result prediction using neural networks and deep learning. In *2020 International Conference on Reliability, Infocom Technologies, and Optimization (Trends and Future Directions)*, pages 229–231, 2020. doi:10.1109/ICRITO48877.2020.9197811.

[40] Shaobo Wang, Pan Zhao, Biao Yu, Weixin Huang, and Huawei Liang. Vehicle trajectory prediction by knowledge-driven lstm network in urban environments. *Journal of Advanced Transportation*, 2020:1–20, 11 2020. doi:10.1155/2020/8894060.