ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΣΤΑΤΙΣΤΙΚΑ ΜΟΝΤΕΛΑ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΤΩΝ ΟΜΑΔΩΝ ΣΤΟ ΕΥΡΩΠΑΪΚΟ ΠΡΩΤΑΘΛΗΜΑ ΠΟΔΟΣΦΑΙΡΟΥ

Θοδωρής Μπάζμπας

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Μάιος 2026

Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Π Ε Ι Ρ Α Ι Ω Σ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΣΤΑΤΙΣΤΙΚΑ ΜΟΝΤΕΛΑ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΤΩΝ ΟΜΑΔΩΝ ΣΤΟ ΕΥΡΩΠΑΪΚΟ ΠΡΩΤΑΘΛΗΜΑ ΠΟΔΟΣΦΑΙΡΟΥ

Θοδωρής Μπάζμπας

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Μάιος 2026

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από τη ΓΣΕΣ του Τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμ. …….. συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Στατιστική

Τα μέλη της Επιτροπής ήταν:
- Καθηγητής Κωνσταντίνος Πολίτης (Επιβλέπων)
- Καθηγητής Γεώργιος Τζαβελάς
- Αναπληρωτής Καθηγητής Ιωάννης Τρανταφύλλου

Η έγκριση της Διπλωματική Εργασίας από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

UNIVERSITY OF PIRAEUS

School of Finance and Statistics

Department of Statistics and Insurance Science

POSTGRADUATE PROGRAM IN
APPLIED STATISTICS

STATISTICAL MODELS FOR ANALYSING THE PERFORMANCE OF TEAMS IN EUROPEAN FOOTBALL CHAMPIONSHIPS

By

Thodoris Bazmpas

MSc Dissertation

submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfilment of the requirements for the degree of Master of Science in Applied Statistics

Piraeus, Greece
May 2026

Ευχαριστίες

Η διπλωματική αυτή εργασία αποτελεί το επιστέγασμα των μεταπτυχιακών μου σπουδών και δε θα μπορούσε να ολοκληρωθεί χωρίς την πολύτιμη συμβολή ορισμένων ανθρώπων. Θα ήθελα αρχικά να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Κωνσταντίνο Πολίτη, για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου το συγκεκριμένο θέμα, καθώς και για την καθοδήγηση μου, τη συνεχή υποστήριξη και τις πολύτιμες συμβουλές του σε όλη τη διάρκεια της εκπόνησης της εργασίας. Η επιστημονική του κατάρτιση, η υπομονή και η προθυμία του να συζητά κάθε απορία υπήρξαν καθοριστικές για την ολοκλήρωση αυτής της προσπάθειας. Ιδιαίτερες ευχαριστίες οφείλω στην οικογένεια μου για την αμέριστη συμπαράσταση, την υπομονή και την ηθική υποστήριξη που μου παρείχαν καθ' όλη τη διάρκεια των σπουδών μου. Χωρίς τη δική τους συμπαράσταση, η ολοκλήρωση αυτού του κύκλου θα ήταν πολύ πιο δύσκολη. Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου που στάθηκαν δίπλα μου, μοιράστηκαν τις αγωνίες μου και με ενθάρρυναν σε κάθε βήμα. Πάνω από όλους, όμως, ευχαριστώ τον Θεό που με προστάτευσε και με κρατούσε υγιή όλα αυτά τα χρόνια, χαρίζοντας μου τη δύναμη, τη θέληση και την υπομονή που χρειάστηκα για να φέρω εις πέρας αυτή την προσπάθεια. Σε Αυτόν ανήκει η δόξα.

Περίληψη

Η ραγδαία ανάπτυξη της τεχνολογίας και των υπολογιστικών συστημάτων έχει καταστήσει τη στατιστική ανάλυση και την επιστήμη των δεδομένων απαραίτητα εργαλεία σε πλήθος επιστημονικών και επαγγελματικών πεδίων. Η ικανότητα συλλογής, επεξεργασίας και ερμηνείας μεγάλου όγκου πληροφορίας επιτρέπει την εξαγωγή αξιόπιστων συμπερασμάτων και τη λήψη τεκμηριωμένων αποφάσεων. Οι μέθοδοι αυτές έχουν βρει πρόσφατα εφαρμογή και στον αθλητισμό, όπου η ποσοτική προσέγγιση συμβάλλει ολοένα και περισσότερο στη βελτιστοποίηση της απόδοσης.

Ειδικότερα στο ποδόσφαιρο, η ανάλυση δεδομένων έχει μετασχηματίσει τον τρόπο με τον οποίο οι ομάδες προσεγγίζουν τόσο την τακτική προετοιμασία όσο και τη διεξαγωγή των αγώνων. Από την αξιολόγηση της ατομικής απόδοσης των ποδοσφαιριστών έως την πρόβλεψη μελλοντικών αποτελεσμάτων, η χρήση στατιστικών μοντέλων και αλγορίθμων μηχανικής μάθησης παρέχει πολύτιμα εφόδια σε προπονητές, αναλυτές και σκάουτερ, επιτρέποντάς τους να εντοπίζουν ανταγωνιστικά πλεονεκτήματα και να λαμβάνουν στρατηγικές αποφάσεις.

Η παρούσα διπλωματική εργασία επικεντρώνεται στην πρόβλεψη της πρόκρισης των εθνικών ομάδων ποδοσφαίρου από τη φάση των ομίλων στο Ευρωπαϊκό Πρωτάθλημα (UEFA EURO). Για τον σκοπό αυτό συλλέχθηκαν δεδομένα από τις διοργανώσεις των ετών 2012, 2016, 2020 και 2024. Χρησιμοποιήθηκαν διαφορετικά μοντέλα όπως λογιστική παλινδρόμηση, αλγόριθμοι μηχανικής μάθησης και νευρωνικά δίκτυα προκειμένου να αξιολογηθεί η ικανότητά τους να προβλέπουν εάν μια ομάδα θα προκριθεί στην επόμενη φάση ή όχι. Τα αποτελέσματα της έρευνας ανέδειξαν τους σημαντικότερους παράγοντες που επηρεάζουν την πρόκριση. Η ανάλυση προσφέρει πολύτιμες πληροφορίες για την κατανόηση των κρίσιμων παραμέτρων που καθορίζουν την επιτυχία στη φάση των ομίλων, συμβάλλοντας στη βελτίωση της στρατηγικής ανάλυσης σε ένα τόσο ανταγωνιστικό πεδίο όπως είναι το Ευρωπαϊκό Πρωτάθλημα ποδοσφαίρου.

Abstract

The rapid advancement of technology and computational systems has established statistical analysis and data science as indispensable tools across a wide range of scientific and professional fields. The ability to collect, process, and interpret large volumes of information enables the extraction of reliable conclusions and the making of informed decisions. These methods have recently found application in sports as well, where a quantitative approach increasingly contributes to performance optimisation.

In football in particular, data analysis has transformed the way teams approach both tactical preparation and match execution. From evaluating individual player performance to predicting future outcomes, the use of statistical models and machine learning algorithms provides valuable resources for coaches, analysts, and scouts, allowing them to identify competitive advantages and make strategic decisions.

This thesis focuses on predicting whether national football teams advance from the group stage in the European Championship (UEFA EURO). For this purpose, data were collected from the tournaments of 2012, 2016, 2020, and 2024. Different models were employed, such as logistic regression, machine learning algorithms, and neural networks, in order to assess their ability to predict whether a team will progress to the next round. The results highlighted the most important factors influencing qualification. The analysis offers valuable insights into the critical parameters that determine success in the group stage, contributing to the improvement of strategic analysis in such a competitive field as the European Football Championship.

Contents

CHAPTER 1

1. Introduction

In this thesis, an analysis of the UEFA European Football Championship (EURO), the premier national team competition in Europe, is conducted using a combination of statistical techniques and machine learning methods. The study is based on numerical data that were manually collected from the official UEFA website and cover the tournaments from 2012 to 2024, as complete statistical data for previous years are not available. To achieve a thorough investigation, the dataset was divided into two levels of analysis: first, a binary variable indicating whether a team advanced beyond the group stage ("Advanced") or was eliminated ("GroupStage"); second, a three-level classification based on final performance: low-level teams (eliminated in the group stage), medium-level teams (eliminated in the round of 16 or the quarter-finals), and high-level teams (reaching the semi-finals or the final). By using these distinct classes, we aim to highlight the factors that influence a team's progression in the tournament and to investigate whether statistical performance metrics can serve as reliable predictors.

In the second chapter, an extensive historical overview of the European Championship is first presented, from the inaugural tournament in 1960 to its modern formats, describing the different structures implemented over time. Subsequently, a literature review is presented, with references to relevant scientific articles that examine performance in international football competitions, the conclusions of which serve as a reference point for this thesis.

In the third chapter, descriptive measures for the explanatory variables are recorded in detail, both at the tournament aggregate level and at the per-match level. In parallel, graphical representations of the data are produced using boxplots and evolution charts per tournament, in order to highlight the differences between teams of different performance levels and to make the results more comprehensible and comparable.

In the fourth chapter, the normality of the quantitative variables is first tested using the Lilliefors test, so that the appropriate parametric or non-parametric methods can be selected. Next, correlation coefficients (Pearson and Spearman) between the variables are computed to explore their relationships and to detect possible multicollinearity. Finally, tests for equality of means (ANOVA, Kruskal-Wallis) are performed for the teams classified into the three performance categories, as well as by tournament year. For all tests, the significance level was set at 5%.

In the fifth chapter, we investigate which variables play a decisive role in advancing beyond the group stage and in being classified into three performance levels. For this purpose, appropriate generalized linear models are fitted: a binary logistic model for predicting advancement (variable Progress) and a multinomial logistic model for classification into three categories (Phase). The results are then interpreted, and model

fit is evaluated using appropriate diagnostic checks (VIF, likelihood ratio test, Hosmer-Lemeshow test, pseudo-R^2).

In the sixth chapter, through the use of machine learning techniques and appropriate data preprocessing, we aim to extract useful knowledge and compare predictions with the statistical models. Specifically, after first selecting the most important features using the Boruta method, classification algorithms (k-NN, SVM, neural networks) are applied to predict advancement from the group stage. Model evaluation is performed using 10-fold cross-validation and metrics such as accuracy, sensitivity, and F1-score.

In the seventh and final chapter, the findings of the previous chapters are summarised, the results from the different methodological approaches are compared, and the final conclusions of the thesis are presented.

CHAPTER 2

2. Historical Background and Literature Review

This chapter aims to provide a historical overview of the UEFA European Championship (Euro), tracing its evolution and format, while also reviewing relevant scientific literature on the topic published in international journals.

2.1 UEFA European Championship

2.1.1 History and Evolution of the Tournament

The UEFA European Football Championship (commonly known as the EUROS) is one of the top international football competitions in the world. It has been held every four years since 1960, in which the men's national teams of UEFA members compete to be crowned the champions of Europe. After a series of qualifying matches, the 24 best teams (since 2016) compete in the final tournament to win the trophy. The final tournament is divided into two distinct phases: the group stage and the knockout stage.

In the group stage, the teams are divided into six groups of four teams each. The teams are ranked into pots based on the points they earned in the EURO qualifiers. The draw for the group stage is made randomly from these pots. The top two teams from each group, along with the four best third-placed teams, advance to the knockout stage. Team standing in the groups are determined by points. The knockout stage is played in single-elimination matches, where a winner must be decided in each game. It begins with the "Round of 16", which, as the name suggests, consists of 16 teams. This is followed by the quarter-finals, the semi-finals, and the grand final. The teams that lose in the semi-finals compete in a third-place play-off (or consolation final) for third place.

UEFA selects the host(s) based on financial and logistical criteria, such as stadiums, infrastructure, and security, and not on sporting merit, such as the strength of a national team. In recent years, the EURO has been co-hosted by multiple countries. For example, EURO 2028 will be held in the United Kingdom and Ireland. Overall, the rules and formats are continually evolving, as seen in the 2016 EURO, and this trend is expected to persist in the future. Let's examine the overall evolution up to now.

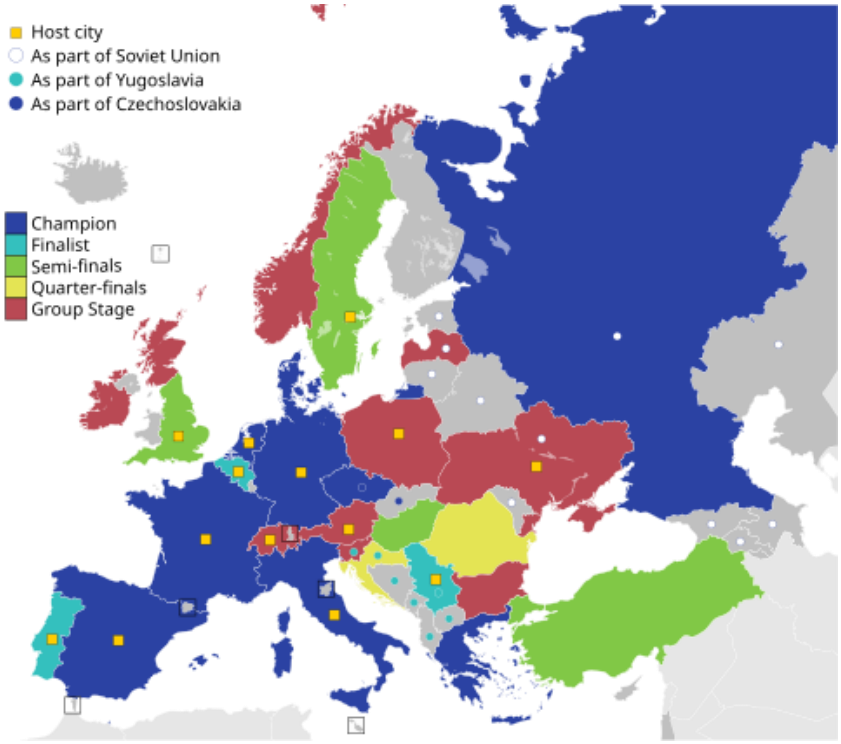

Figure 2.1: Countries with the best results and host countries (Source: Wikipedia.org)

International football made its worldwide debut at the 1908 Olympic Games in London. The concept of establishing a European championship for national teams dates back to 1927, attributed to the Frenchman Henri Delaunay, a former footballer and referee, who served at the time as FIFA's vice-president. However, the circumstances were not yet favourable for its realization. Finally, in July 1960, the first tournament was held, with 17 countries participating.

2.1.2 The different formats over the years

Many different formats followed over the years. For example, in 2016, the tournament's format was modified. This indicates that further changes are likely to happen, underscoring the fact that the European Championship is a continuously evolving tournament. However, let us first review what preceded these changes.

The first official tournament, known as the "European Nations Cup", was held in 1960. Seventeen teams participated in the qualifiers, with the final four (USSR, Yugoslavia, Czechoslovakia, France) competing in the semi-finals and final in Paris, France, where the USSR claimed the first title. This structure unchanged for the next four tournaments (1964, 1968, 1972, 1976). Significant events of this period were Spain's triumph on home soil in 1964, Italy's first victory in 1968 (the year extra time and a penalty shootout were introduced to decide the winner if necessary), and West Germany's first win in 1972. The period culminated with Czechoslovakia's dramatic win over West Germany in a penalty shootout in 1976.

The first major format change came in 1980 when the tournament was expanded to 8 teams for the final stage, divided into two groups of four. The group winners advanced

directly to the final, while the runners-up competed for the third place. West Germany emerged as the champion. The format remained stable from 1984 to 1992, with 8 teams in the final tournament. However, a key improvement was introduced: the top two teams from each of the two groups now advanced to semi-finals, creating a more exciting knockout stage. This era saw memorable victories for France (1984), the Netherlands (1988), and the surprising triumph of Denmark (1992).

From 1996 to 2012, the European Championship expanded to 16 teams, divided into four groups of four. The top two teams from each group advanced to a new quarter-final stage, before the semi-finals and the final. This period saw victories for Germany (1996), France (2000), Greece (in 2004, as a major surprise) and Spain (who won two consecutive titles in 2008 and 2012).

A significant expansion occurred in 2016, when the EURO changed its format, with 24 teams participating in the final stage, divided into six groups of four. The top two teams from each group and the four best third-placed teams advanced to the new round of 16, before the quarter-finals, semi-finals and the final. Portugal won the title that year. The same format was used in EURO 2020, which was hosted in 11 different cities to celebrate the tournament's 60th anniversary, with Italy emerging as the champion. EURO 2024 in Germany also featured the 24-team format, and it was won by Spain. It has been demonstrated that the EURO remains a dynamic and evolving tournament, which adapts to the needs and realities of modern football.

In summary, the following image shows the countries and the number of times they have won the European Championship, with Germany's record incorporating the titles and history of West Germany. It is noteworthy, however, that on what is now present-day German territory, three distinct national teams have participated in international football competitions in the past: West Germany, East Germany, and the Saarland national football team, which existed from 1950 to 1956.

Figure 2.2: Countries with the most UEFA EURO (Source: Chitchart.com)

This diploma thesis aims to address key questions, including: What are the significant predictive factors influencing team performance in the UEFA European Championship (EURO)? Is the style and nature of football consistent across different tournaments, and if not, which factors vary in their impact on result? Finally, the study will explore the feasibility of developing effective statistical and machine learning models utilizing various methodologies to predict the expected number of goals, classify teams based on their strength, and ultimately forecast their success.

The following section reviews existing literature with comparable aims, featuring studies from international sources that focus on the EURO Cup as well as other major international football tournaments.

2.2 Literature Review

This section surveys scientific articles and research pertinent to the thesis topic, reviewing the advancements achieved in this field of study.

The study titled "Offensive Transitions in High-Performance Football: Differences Between UEFA EURO 2008 and UEFA EURO 2016" by Ruben Maneiro, Claudio A. Casal, et al. (2019) was conducted with a dual objective:

a) to describe and compare offensive transitions between the EURO 2008 and 2016 tournaments, and
b) to develop predictive models to assess whether the relationship between various factors and the success of these transitions changed between the two tournaments.

The researchers employed a systematic data collection approach, analyzing video recordings from the tournaments. They meticulously recorded and categorized specific in-game events using a predefined and standardized observation instrument. This method involved directly watching the matches and logging each offensive transition event based on a set of criteria, without any intervention in the game itself.

A total of 1,533 offensive transitions from the quarter-final, semi-final, and final matches of both tournaments were analyzed. The following methods were applied for statistical analysis:

1) Analysis of Proportions and Chi-square Test: To identify statistically significant differences in the frequencies of variables (e.g. when and how transitions occur) between 2008 and 2016.
2) Logistic Regression Models: To investigate which factors (e.g. zone of the pitch, type of initiation) predict the success of a transition phase and whether this predictive relationship differs from one EURO to another.

The findings showed statistically significant differences ($p < 0.05$) between the two tournaments:

- The number, manner, or frequency of offensive transitions changed significantly within 8 years.
- The factors that predicted the success of a transition phase in EURO 2008 were not necessarily the same as those in EURO 2016. The logistic regression models "predicted" differently for each tournament, suggesting a genuine evolution in tactics and playing style.

In their study titled "Success factors in national team football: an analysis of the UEFA EURO 2020" Vincent Renner, Konstantin Gorgen, et al. (2024) aimed to identify the success factors in European national team tournaments (EURO), with a primary focus on EURO 2020 and a comparative analysis against EURO 2016. The research sought to distinguish between factors that are generally important (stable across tournaments) and factors whose significance depends on the prevailing tactical environment and playing style of each tournament.

Data from 51 matches per tournament (EURO 2016 & 2020) were used, providing 102 team-level observations. The data included traditional match statistics (e.g. shots, passes, dribbles) and contextual variables (e.g. market value ratio, home advantage, distance covered difference).

To address the small sample size and high multicollinearity among the numerous variables, an advanced statistical technique, "LASSO Crossfitted Stability-Selection" was employed. Two distinct approaches were used to measure success:

- Goal-based: The dependent variable was goal difference (a continuous variable)
- Result-based: The dependent variable was the match outcome (Win/Draw/Loss), analyzed using Proportional Odds logistic regression.

Three factors were found to be consistently significant across both tournaments, regardless of tactical trends:

- The ratio of the market value of a team's starting lineup to that of its opponent. The higher the ratio, the better the result.
- The difference in distance covered by a team compared to its opponent. Running more than the opponent is important.
- The percentage of shots on target.

Other factors were significant only in specific EURO tournaments, reflecting prevailing tactical trends:

- For EURO 2020 (more offensive, higher average goals): Positive effects from dribbles, save rate (goalkeeper effectiveness), and long passes. Negative effects from tackles attempted (an indicator of defensive disposition) and crosses.
- For EURO 2016 (more defensive, lower average goal): The most significant factor was shots from fast-break (counter-attacks), a characteristic of more defensive teams relying on counter-attacks.

Conclusion: The study concludes that success factors in national team tournaments can be categorized into two types: (a) general and stable factors (economic/qualitative

superiority, physical conditioning, accuracy in scoring), and (b) factors dependent on the tactical context of each tournament. The relative importance of tactical factors changes based on the evolution of the game and how teams adapt to prevailing trends. The use of the "LASSO Crossfitted Stability-Selection" method proved crucial for reliably identifying these factors despite the small sample size and data complexity. The result demonstrates that football is a dynamic system where successful strategies continually evolve.

In the study titled “Hybrid Machine Learning Forecasts for the UEFA EURO 2020”, A. Groll, L. M. Hvattum, et al. (2021) aimed to develop and evaluate hybrid machine learning models for predicting the outcomes of UEFA EURO 2020 by combining multiple data sources and methods. The primary goal is to estimate winning and progression probabilities for all teams, as well as to compare the predictive performance of different models.

The research was based on the comprehensive collection and analysis of four distinct categories of data:

1. Traditional Variables: Economic (e.g. GDP per capita), demographic (e.g. population), and team structure indicators (e.g. market value, average age, number of players abroad).
2. Historical Match Results: An extensive dataset of past matches was used to estimate the current competitive strength of each team through Poisson models.
3. Win probabilities from Bookmakers: Odds from numerous betting agencies were collected and processed to analyze the consensus opinion of the betting market.
4. Individual Player Ratings: Evaluation metrics based on the performance of individual players within their club teams were incorporated.

Hybrid Machine Learning Models were developed, combining ensemble learning algorithms (Random Forest with the cforest and ranger implementations, and XGBoost) with a regularized regression approach (Lasso-Poisson). These models were enriched with the strength estimates derived from the three aforementioned methods (Poisson ranking, bookmaker consensus, player ratings). For the full simulation of the tournament, 100,000 Monte Carlo simulations were performed. In each simulation, match outcomes were generated stochastically based on the estimated parameters of the models, allowing for the precise calculation of the probability of progression and victory for each team at every stage of the tournament.

The detailed analysis of the results highlighted the following conclusions:

- From the comparison of the models, the cforest algorithm (conditional inference forest) demonstrated the highest overall predictive capability in forecasting match outcomes (win/draw/loss)
- Variable importance analysis revealed that the most informative factors were the market value of the teams and the consensus abilities derived from bookmakers.
- Predictions for the tournament winner ranked France as the clear favorite with a probability of 14.8%, followed by England (13.5%) and Spain (12.3%).

- Evaluation of the model on historical data (EURO 2004-2016) showed that its results largely agreed with the predictions of professional bookmakers, confirming the reliability of the approach. However, the research explicitly acknowledged the limitations imposed by external factors, such as the Covid-19 pandemic and the absence of spectators, admitting that such events can drastically affect athletic performance and reduce the accuracy of any predictive model.

In the study titled "The impact of imbalanced groups in UEFA Euro 1980–2024 and comparison with the FIFA World Cup", Michael A. Lapre and Julia G. Amato (2025) aimed to evaluate the imbalance between groups in UEFA EURO tournaments (1980-2024) and compare it with the corresponding imbalance in the FIFA World Cup. Additionally, it examined how this imbalance affects teams' probability of advancement and assessed the effectiveness of the ranking systems used by UEFA. Although UEFA does not face the problem of fairly distributing slots among continents (all teams are European), does it succeed in creating balanced groups? According to the research, the answer is no.

The method that was followed is as follows:

- Measuring Team and Group Strength: Elo ratings of all teams at the time of the draw were used. Group strength was measured in two ways: (a) as the average Elo rating of all 4 teams in the group and (b) as the average Elo rating of the 3 strongest teams in the group (excluding the weakest), a method that is more realistic when only 2 teams advance.
- Measuring Imbalance: Imbalance was calculated as the variation in group strength within the same tournament. A large range indicates significant imbalance.
- Impact on Advancement: To measure how imbalance affects success, logistic regression was applied. The dependent variable was advancement to the quarter-finals and semi-finals. The main independent variable was the average Elo rating of a team's opponents in its group (group opponents rating).
- Comparison of Ranking Systems: To determine which system is better, the predictions of the official UEFA ranking were compared with those of the Elo rating in 148 matches (where neither team was a host and the match had winner).

The results that were obtained were:

- Imbalance Exists: Significant imbalance was found between groups in both the EURO and the World Cup
- Imbalance is Similar: The degree of imbalance in the EURO is statistically indistinguishable from that in the World Cup. That is, UEFA performs just as poorly as FIFA in creating balanced groups.
- Impact on Advancement: Imbalance significantly affects the probability of advancement. A team facing stronger opponents in the group has a lower probability of reaching the quarter-finals. This effect is the same for both tournaments.

- UEFA's System is Inferior: The Elo rating is a clearly superior predictor compared to the official UEFA ranking. Models based on Elo had higher predictive accuracy (higher Pseudo R^2 and Area under Curve (AUC) in ROC).

CHAPTER 3

3. Descriptive Statistics

In the present chapter, we introduce and describe the data that will be used in this study. Following their presentation, we proceed with a descriptive statistical analysis and the visualization of the data. The aim is to understand their characteristics and how they are distributed across the various phases of the event organization, depending on the outcome, the team's potential (or strength), and the date.

3.1 The Data

The data that will be used in the present study for our analysis are from UEFA (uefa.com) and pertain to EURO tournaments from 2012 onwards. The reason for this is that prior to that year, complete data are not available, but only some basic statistics which, however, do not meet the requirements of this specific study.

We have two datasets that will be used: one containing the data of the teams participated in each EURO tournament overall, and one containing the data of the teams on per-match basis. The data we have pertain from the group stage onwards, which have almost identical columns.

In total, the two primary datasets are presented below:

Variable Name	Description
Year	The year of the EURO tournament
Teams	The name of the national team
Final Rank	The final position of the team in the tournament
Matches played	The total number of matches played by the team in the tournament
Won	The number of matches won
Drawn	The number of matches drawn (tied)
Lost	The matches of matches lost
Goals	The total number of goals scored by the team
Right foot	Goals scored with the right foot

Left foot	Goals scored with the left foot
Head	Goals scored with the head
Other	Goals scored by another body part
Goals inside area	Goals scored from inside the penalty area
Goals outside area	Goals scored from outside the penalty area
Penalties scored	Goals scored from penalty kicks
Total Attempts	Total number of shot attempts
Attempts on Target	Number of shot attempts that were on target
Attempts off Target	Number of shot attempts that were off target
Blocked	Number of shot attempts that were blocked
Passing Accuracy (%)	The percentage of successful passes
Passes Attempted	The total number of passes attempted
Passes Completed	The total number of passes successfully competed
Possession (%)	The average percentage of ball possession during matches
Crossing Accuracy (%)	The percentage of successful crosses
Crosses Attempted	The total number of crosses attempted
Crosses Completed	The total number of crosses successfully
Free-kicks Taken	The number of free-kicks taken
Attacks	The number of attacking plays built by the team
Assists	The number of assists leading to a goal
Corners Taken	The number of corner kicks taken
Offsides	The number of offside offenses committed
Tackles	The number of tackle attempts
Tackles Won	The number of successful tackles

Tackles Lost	The number of tackles where the dribbler got past the defender
Saves	The number of saves made by the goalkeeper
Goals Conceded	The total number of goals conceded by the team
Own Goals Conceded	The number of own goals scored by the team
Saves from Penalties	The number of penalty kicks saved by the goalkeeper
Clean Sheets	The number of matches in which the team did not concede a goal
Fouls Committed	The number of fouls committed by the team
Fouls Suffered	The number of fouls suffered by the team
Yellow Cards	The number of yellow cards received
Red Cards	The number of red cards received

Table 3.1: The team-level variables

The variables Year and Final Rank were created to serve the needs of the analysis. For the second dataset, we have exactly the same variables, with the exception of Won, Lost, Drawn, Final Rank, Passing Accuracy and Possession. These variables are divided by the number of matches each national team played in each tournament. Therefore, we have the same columns as in the first dataset, but in the second dataset, these variables are expressed on a per-match basis for each team, rather than as aggregate totals.

3.2 Descriptive Statistical Analysis

In the following section, we will present some descriptive measures for the variables that will be used in our analysis. These measures will help us gain an initial understanding of the data and provide useful insights for the subsequent stages of our study. This analysis focuses primarily on quantitative variables, which constitute the majority of the dataset, while categorical variables will play a very useful role in segmenting the quantitative data and in providing initial indications regarding the importance of match characteristics. Naturally, the results presented in this chapter should be interpreted as indicative, as these methods alone do not allow us to draw definitive statistical conclusions. However, they serve as a solid foundation for stablishing objectives and guiding the direction of our analysis.

The primary tool we will use here for visualizing the results is the boxplot. The boxplot, also known as the five-number summary diagram, is constructed as follows:

- The lower base of the rectangle is positioned at Q1 (the first quartile) and the upper base at Q3 (the third quartile).

- The median (δ) is represented by a horizontal line segment inside the rectangle, while the length of the rectangle's bases is chosen arbitrarily.
- The upper and lower whiskers are shaped like a T and an inverted T, respectively. Depending on the type of boxplot, they extend to different boundary values:
    a) In the skeletal boxplot, they extend to the maximum and minimum observation.
    b) In the schematic (full) boxplot, they extend to the largest observation less than or equal to the upper inner fence ($Q3 + 1.5 \times IQR$) and the smallest observation greater than or equal to the lower inner fence ($Q1 - 1.5 \times IQR$), where $IQR = Q3 - Q1$.
- Values that lie beyond the inner fences are marked as outliers. Those located between the inner fence and the outer fence ($Q1 - 3 \times IQR$ or $Q3 + 3 \times IQR$) are characterized as mild outliers, while values beyond the outer fence are characterized as extreme outliers *(Dawson, 2011).*

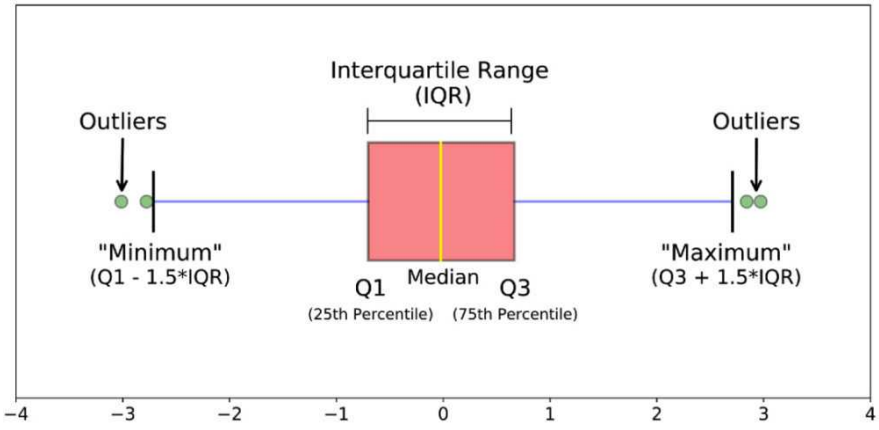

Figure 3.1: BoxPlot Description (Source: https://www.kdnuggets.com/2019/11/understanding-boxplots.html)

3.3 Descriptive Statistical Analysis of Characteristics by Tournament Phase.

Of particular interest is examining the extent to which team characteristics differ depending on the tournament phase in which they occur. We therefore continue by determining the status of these characteristics across the tournament phases. We have categorized the tournament phases as follows: the group stage represents the low level, the Round of 16 and the quarter-finals represent the medium level, and the semi-finals and the two finalists represent the high level. It should be recalled that the phase for each team was determined based on the final position they achieved in each tournament.

For this descriptive analysis, it was necessary to normalize our data. Specifically, we divided all characteristics in the dataset by the number of matches each team played, thus having data per game. At this stage, we will primarily use comparative box plots, along with other types of diagrams where appropriate, to make the comparisons more comprehensible. We begin with the offensive variables.

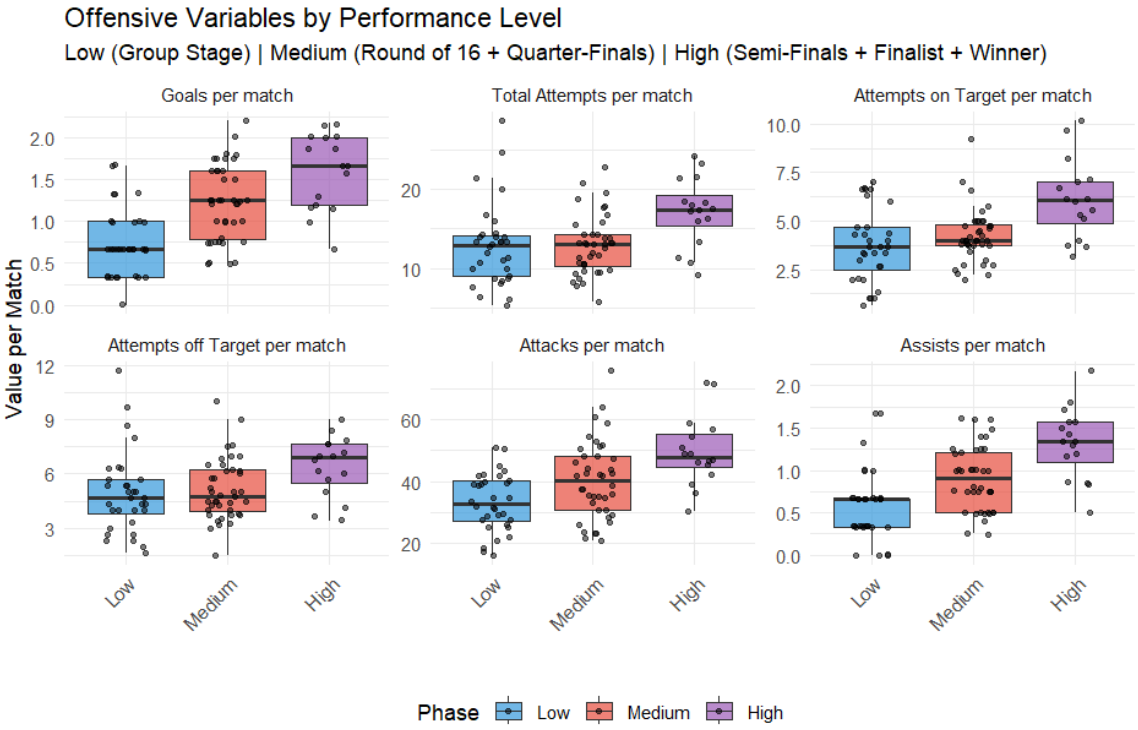

Figure 3.2: Boxplots for the Offensive Variables by Performance Level

Visually, we observe that characteristics such as goals, attacks and assists increase as the performance level of the teams rises. It appears that the median values become progressively higher with each ascending performance tier. Regarding the remaining offensive variables, specifically total shots, shots on target and shots off target, we note that teams in the group stage and those advancing up to the quarter-finals do not exhibit

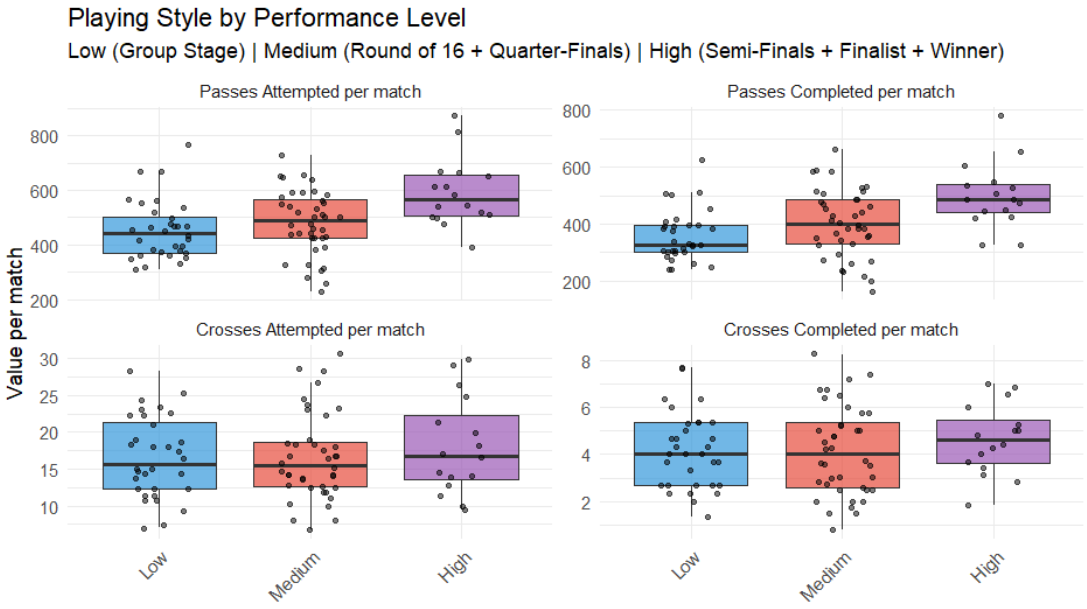

Figure 3.3: Boxplots for the Playing Style by Performance Level

substantial differences. However, teams reaching the semi-finals and the final demonstrate a higher number of total shots, as well as more shots both on and off target. We now proceed to a number of variables related to the teams' playing style.

In terms of pass attempts and successful passes (looking at Figure 3.3 as well), we observe that here too, the higher level of the teams, the greater these variables are. As for crosses (attempts and successful ones), we note that there are no significant differences across all three team levels.

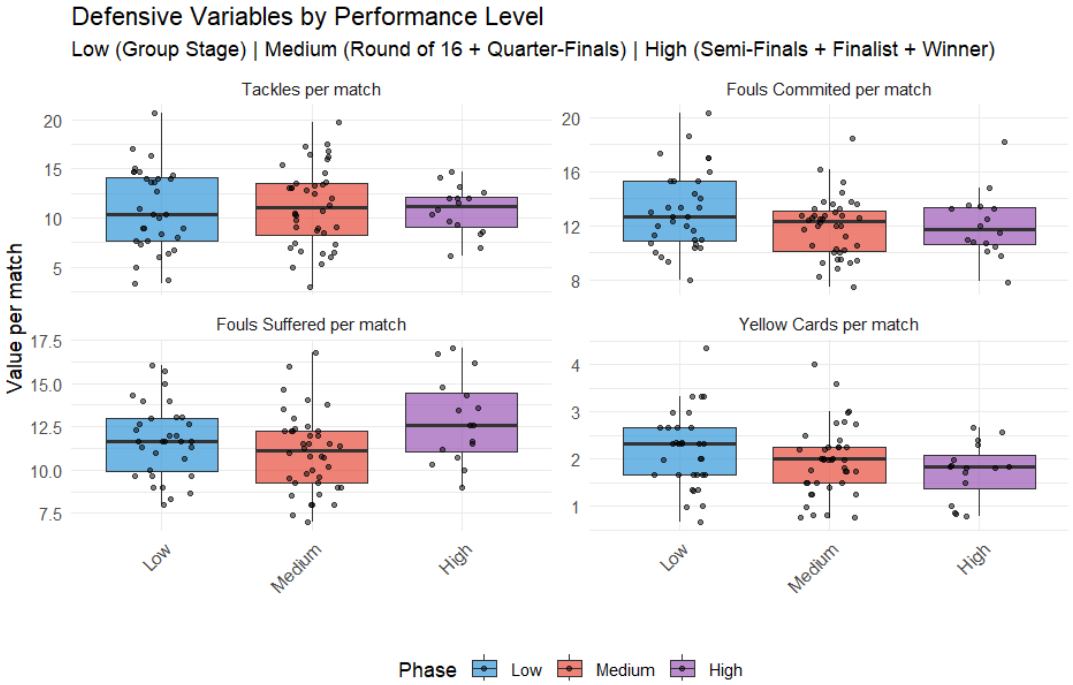

Figure 3.4: Boxplots for the Defensive Variables by Performance Level

In the defensive variables, we do not observe major differences between the teams. However, we could say that the high-level teams suffer more fouls. Furthermore, the

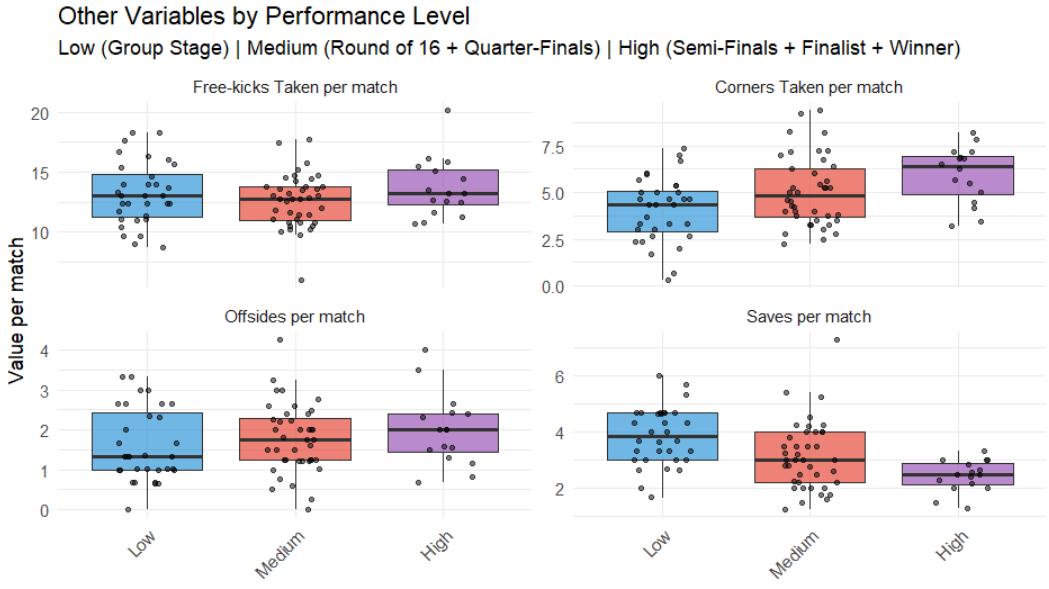

Figure 3.5: Boxplots for Different Variables by Performance Level

higher the level of the teams, the fewer cards they receive. Finally, we will continue with the observations of the teams' remaining characteristics.

We observe that in free-kicks taken and offsides, there is no significant difference regarding the teams' level. Where we see a difference is in corners, where the higher the level of the teams, the more corners they have. This is because they attack more. Furthermore, we see that as teams advance to higher levels, the number of saves their goalkeepers decreases. The following table compares the mean values of the variables we are studying in each tournament phase, highlighting what we observe graphically.

Variables	Low Level Teams	Medium Level Teams	High Level Teams
Goals	0.74	1.22	1.59
Total Attempts	12.8	12.9	17.1
Attempts On Target	3.8	4.2	6.1
Attempts Off Target	4.9	5.1	6.4
Attacks	33.2	40.4	49.7
Assists	0.6	0.9	1.3
Passes Attempted	453.8	480.5	591.5
Passes Completed	358.6	402.2	500.2
Crosses Attempted	16.6	16.6	18.1
Crosses Completed	4.2	4.1	4.6
Yellow Cards	2.2	2.0	1.7
Tackles	10.9	11.2	10.8
Fouls Commited	13.1	11.9	12.1
Fouls Suffered	11.7	11.0	12.8
Free-kicks Taken	13.2	12.6	13.7
Corners Taken	4.1	5.1	5.9
Saves	3.9	3.1	2.4
Offsides	1.6	1.8	2.0

Table 3.2: Average Values of Team Characteristics by Performance Level

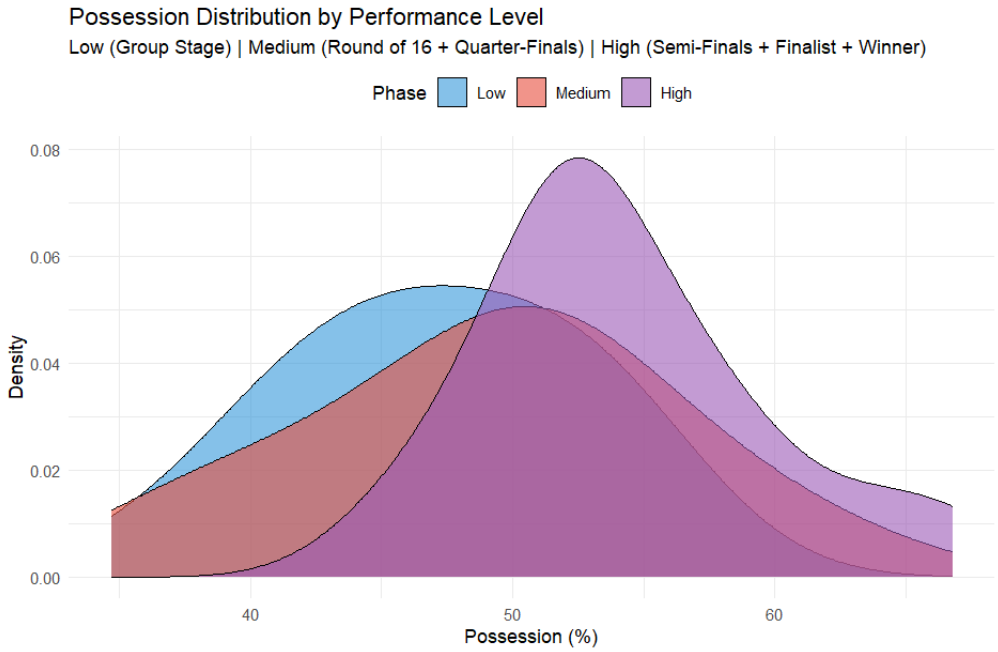

Figure 3.6: Possession Distribution by Performance Level

To make teams' possession comparable, we constructed a diagram to compare the distributions according to the level of the tournament phase. We observe that the teams at the high level have greater possession and greater homogeneity. That is, most of the high-level teams clustered around the same possession percentages. Among the medium-level teams, we see that they have higher possession percentages only compared to the low-level teams, and we observe heterogeneity; meaning the curve of the medium level is more spread out than the others, so we understand that there is a diversity in possession tactics. Below we will examine the passing accuracy.

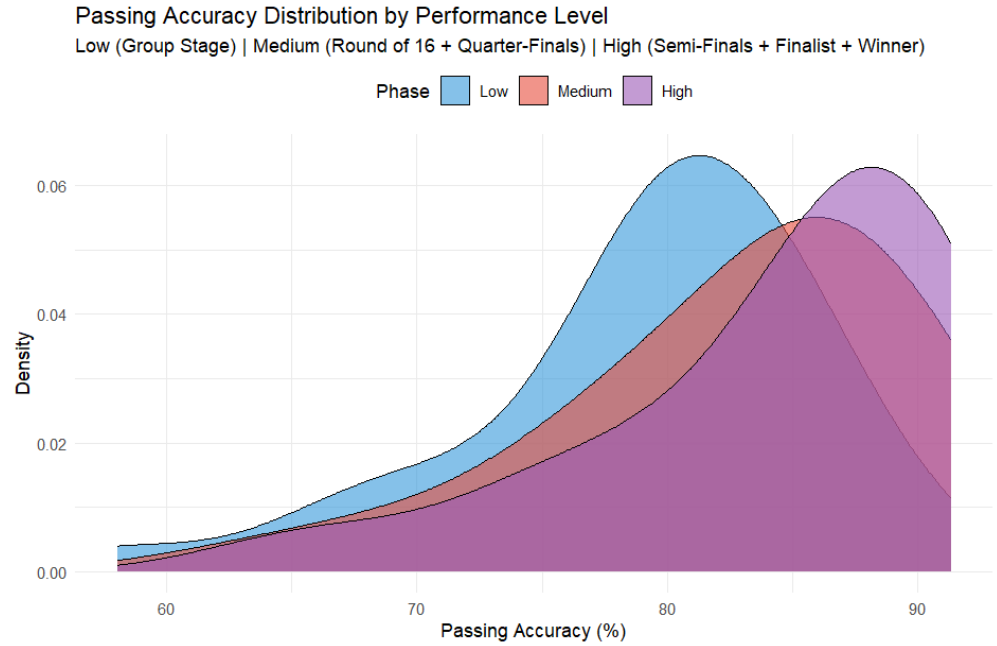

Figure 3.7: Passing Accuracy Distribution by Performance Level

Regarding passing accuracy, we see that the higher the ranking of the teams, the greater their passing accuracy. Finally, we will also examine crossing accuracy.

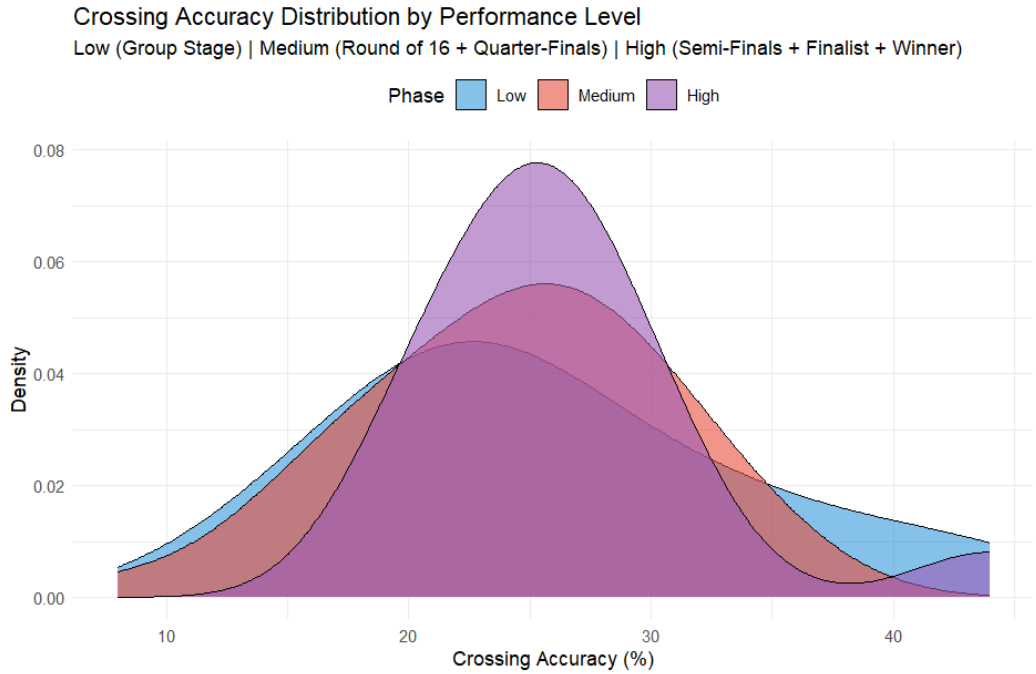

Figure 3.8: Crossing Accuracy Distribution by Performance Level

Here we observe that crossing accuracy is broadly similar across all performance levels, with substantial overlap between distributions. However, high-level teams exhibit greater consistency, as indicated by a more concentrated distribution, while low- and medium-level teams display greater variability, suggesting less consistent crossing performance.

3.3.1 The evolution of characteristics per tournament and team category

Continuing the descriptive analysis, in this section we will gain an initial overview of how our characteristics evolve per tournament and team category (Low/Medium/High) This part is particularly interesting because, for example, we will examine whether goals increase or decrease for each category and tournament. Our study continues along the same lines as before, using charts and other basic features to draw some preliminary insights. To enrich our analysis and examine the comparison of variables from a different perspective, in this section we will use comparative multi-panel line charts. These will provide an initial overview of how the variables evolve per tournament, depending on the performance of teams from different tiers. Here too, our metrics will be normalized by the number of matches played by each team.

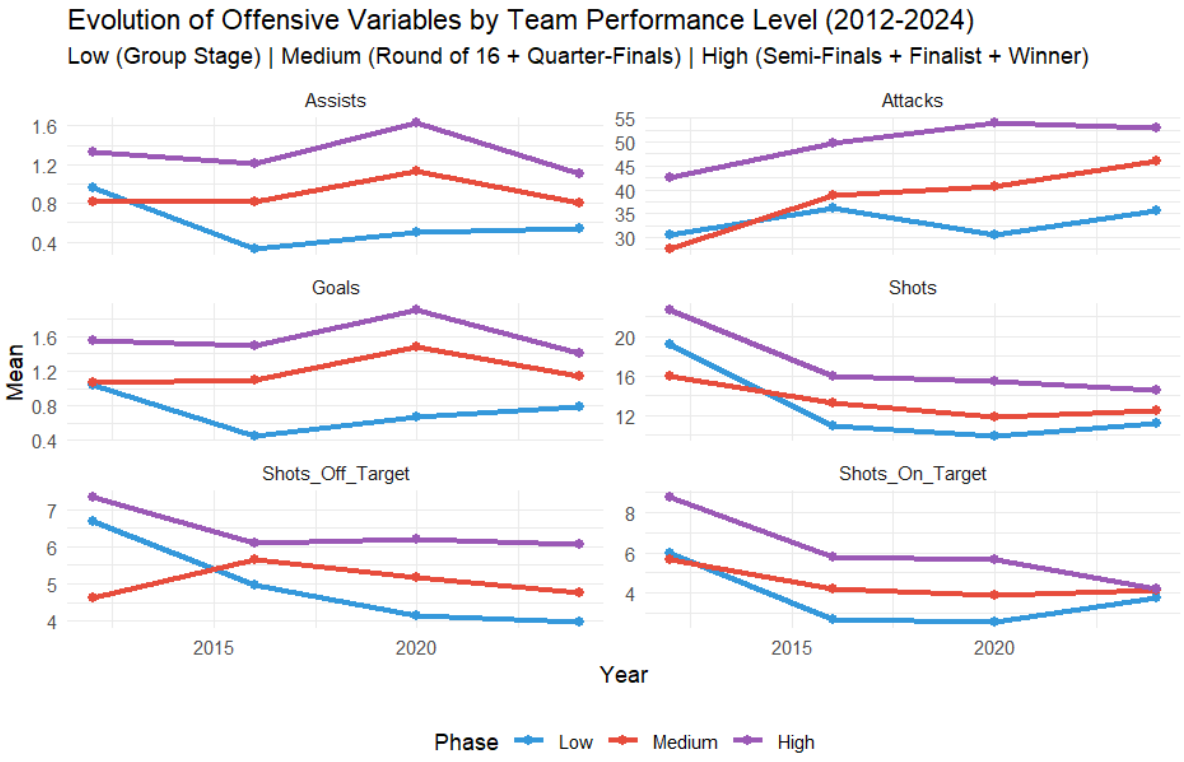

Figure 3.9; Multi-panel Line Charts for Evolution of Offensive Variables by Team Performance Level

From the multi-panel charts for the offensive variables, we observe that shots for each team performance level decrease per tournament, with the most shots for each having been taken in 2012 across all three team performance levels. The same is observed for shots on target. The situation differs slightly for shots off target, as mid-tier teams were the most inaccurate in 2016. The exact opposite is observed in attacks, where the most

attacks are recorded across all three team performance levels in 2024. It should be noted that attacks express the total number of offensive phases a team carried out during the tournament. An attack is generally defined as a phase that starts from the halfway line or deeper and ends in the opponent's area, with the aim of creating a goal-scoring opportunity. Furthermore, we see that goals and assists are not stable but instead show some minor fluctuations, with a peak for high and medium-tier teams in 2020. For low-level teams, goals and assists also show minor ups and downs, with the best year for this group being 2012 and the worst being the subsequent tournament in 2016, after which there is a slight increase up until the most recently organized tournament. Below we will continue with the passing game variables.

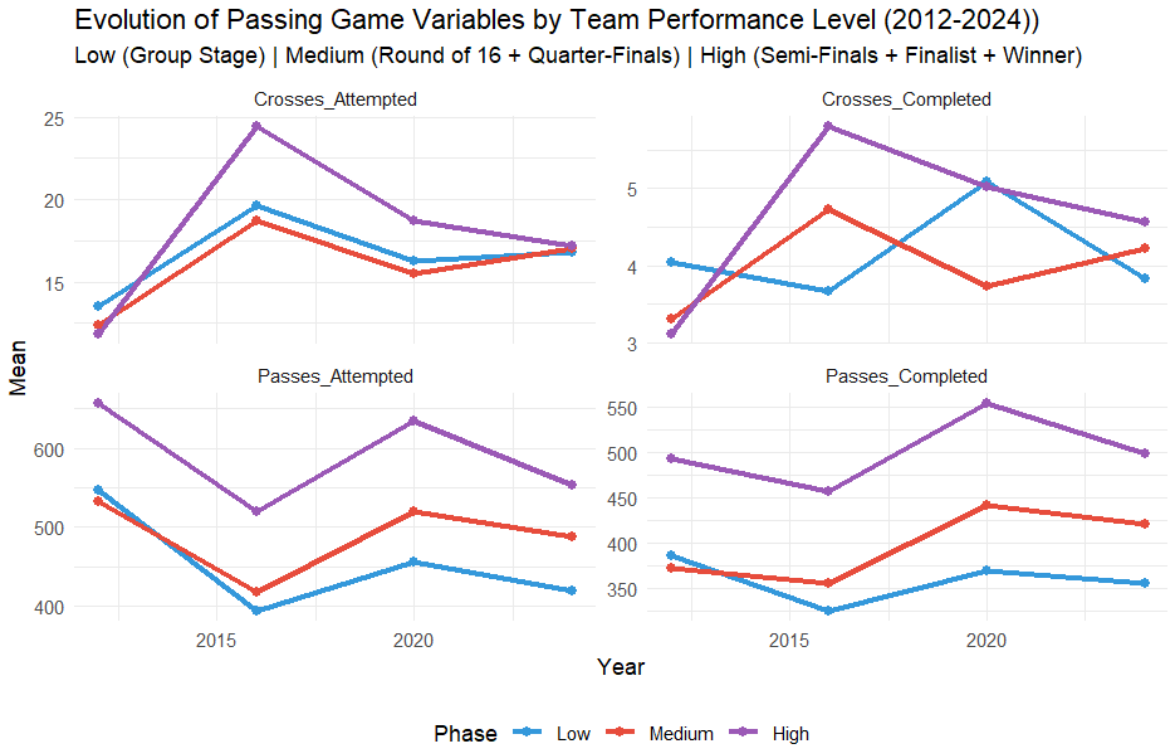

Figure 3.10: Multi-line Panel Charts for Evolution of Passing Game Variables by Team Performance Level

In the evolution of passes for each team level per tournament, we observe continuous fluctuations, with the most pass attempts having been made in 2012 across all three team performance levels. Similarly, continuous fluctuations are observed in crosses attempted, with the highest number occurring in 2016. Regarding pass accuracy, where there are also continuous fluctuations, we observe that it was highest in 2020 for teams at the high and medium performance levels. In contrast, for the low-level teams, the most accurate passes were in 2012. The same pattern is observed in crosses accuracy, with the only difference being that the low-level teams were more accurate in 2020, while the high and medium-tier groups were more accurate in 2016.

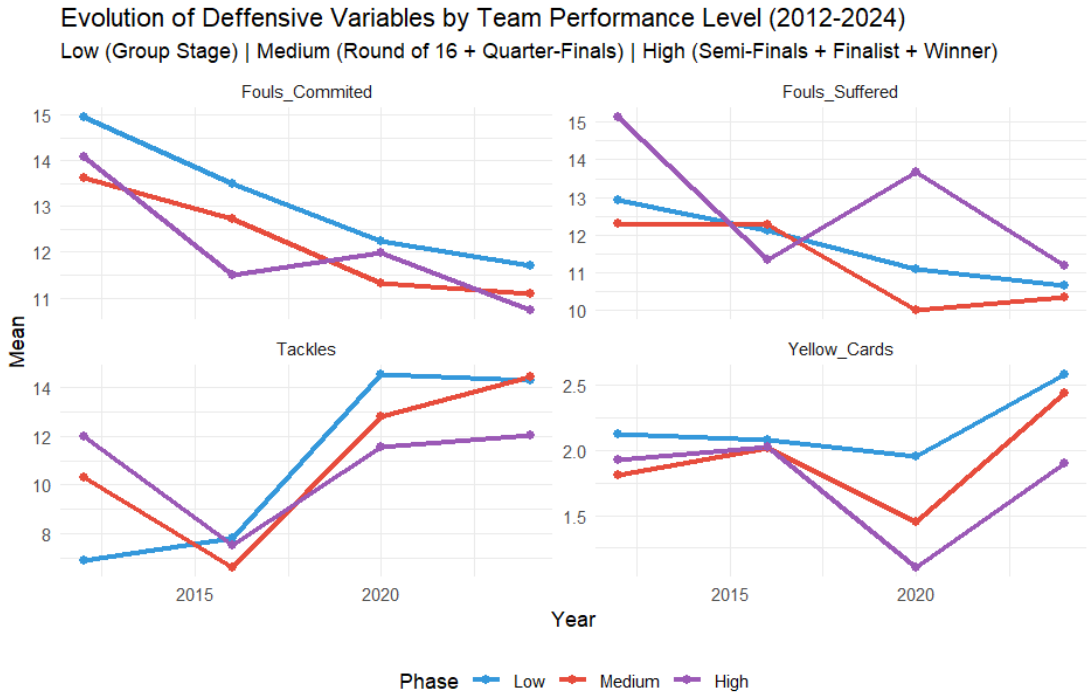

Figure 3.11: Multi-panel Lines Charts for Evolution of Defensive Variables by Team Performance Level

The aforementioned visualizations indicate a consistent decline in fouls across tournament. Furthermore, tackling appears to be gaining preference as a defensive tactic. A concurrent increase in yellow card incidents is also evident, potentially influenced by the implementation of VAR (which has been used since 2016 and onwards) in recent editions, resulting in more stringent officiating and review of challenges.

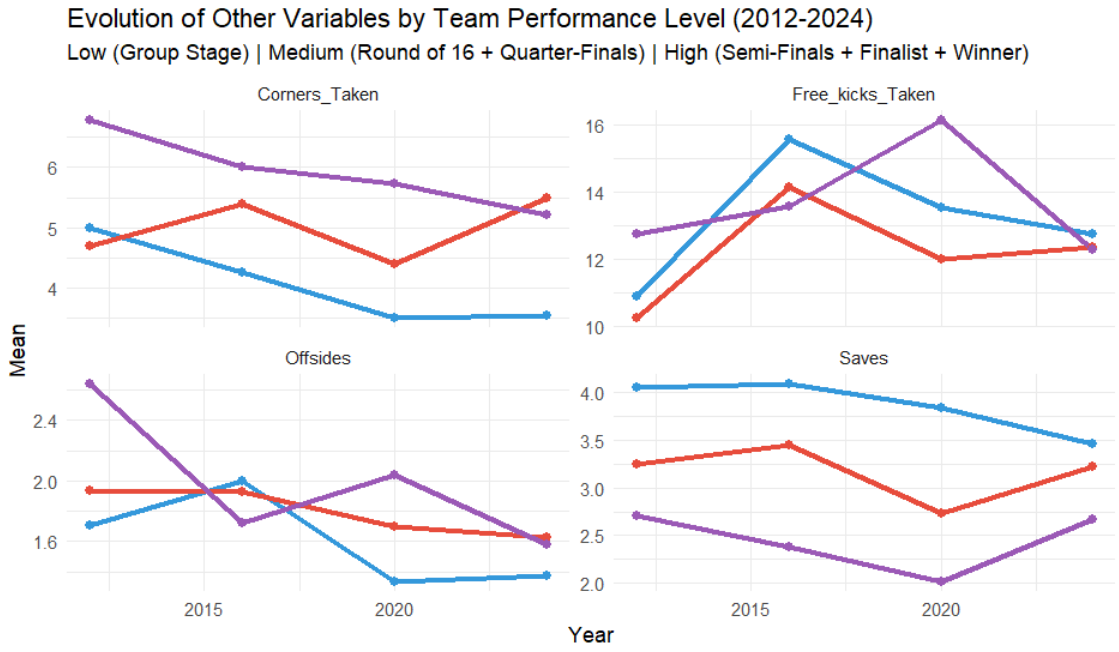

Figure 3.12: Multi-panel Lines Charts for Evolution of Other Variables by Team Performance Level

The data indicates a consistent decline in offside calls across all team performance tiers with each tournament. A similar decreasing pattern is evident in corner kicks, with the exception of medium-tier teams, which exhibit fluctuations within a certain range. Regarding saves, we see that they decrease per tournament for the low-level teams, while for the other teams (high and medium-level) there are fluctuations. Furthermore, we also observe significant in free-kicks taken.

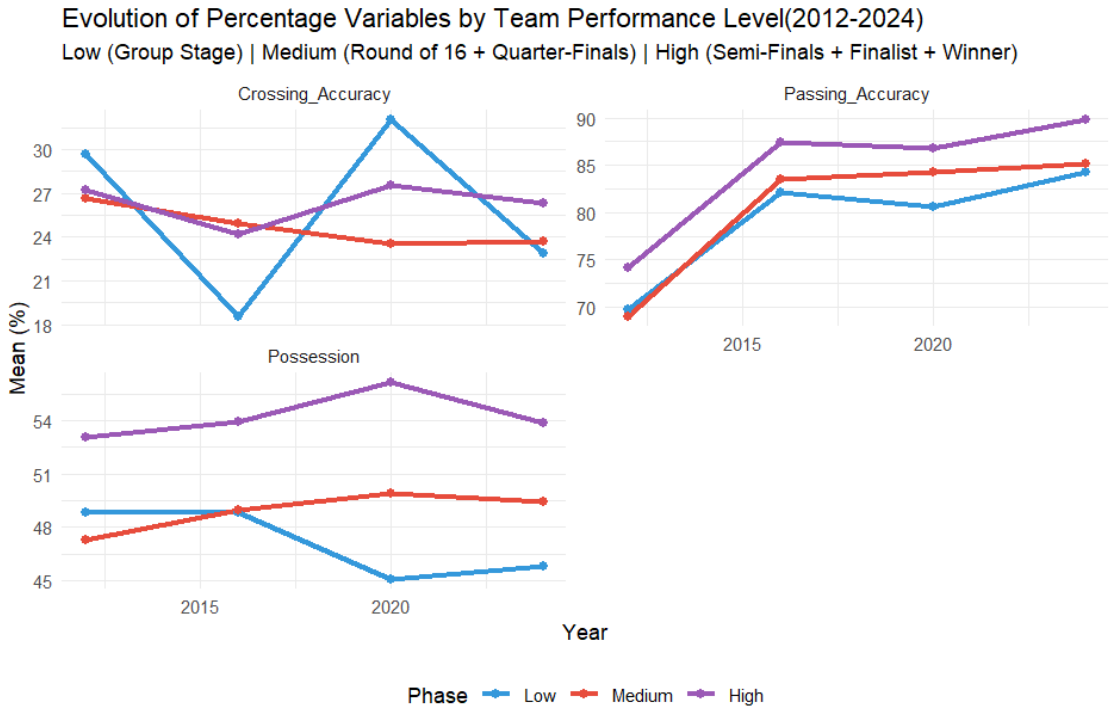

Figure 3.13: Multi-panel Lines Charts for Evolution of Percentage Variables by Team Performance Level

Our findings indicate a tournament-by-tournament improvement in passing accuracy for every competitive level. While crossing success rates remain almost stable for the high and low-level teams, the lowest-performing teams exhibit significant fluctuations per tournament. Possession percentages show remarkable stability throughout the period examined, with no substantial changes across any of the team classifications.

3.4 Evolution of Football Performance Metrics per tournament

In this section, we create graphs that depict the evolution of key football statistics. We specifically analyze the percentage of goals by type (right foot, left foot, head, other), the distribution of goals inside/outside the area, the percentages of attempts on target, off target and blocked and the successful tackle percentage. First, we will look at the goal percentages by type.

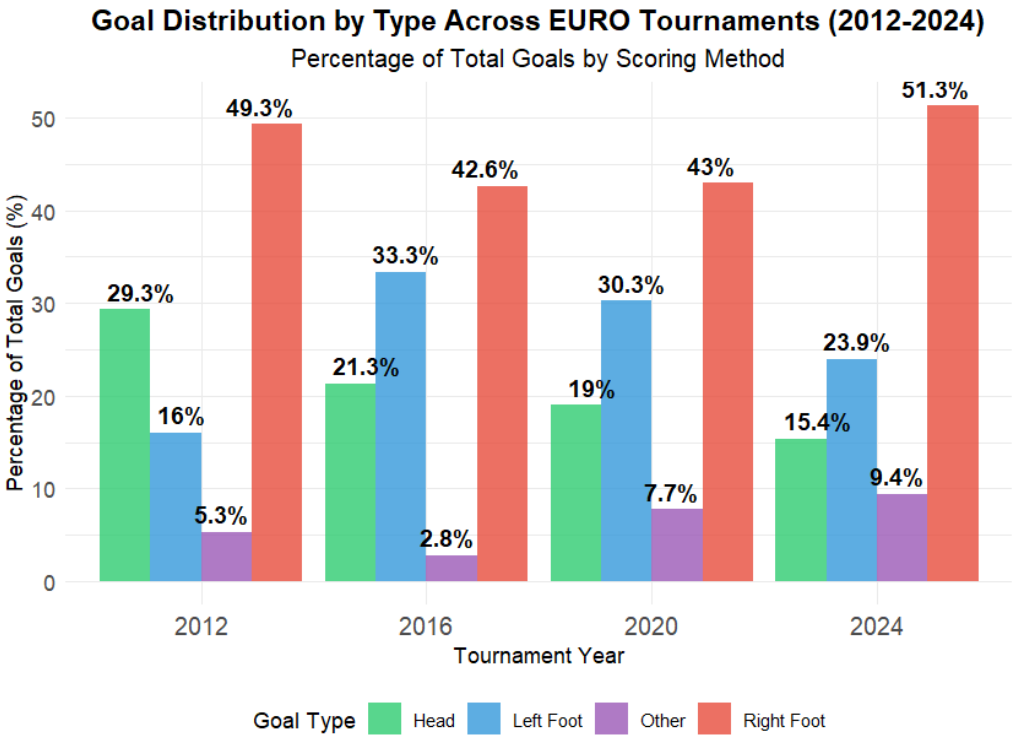

Figure 3.14: Bar Chart for Goal Distribution by Type Across

We can see that in the last four EURO tournaments, the majority of goals are scored with the right foot. Nearly half of all goals are scored with the right foot. In 2012, we observe that headers were the second most frequent scoring method, with the left foot being the third. However, this pattern changes in the subsequent tournaments, where we note that left becomes the second most common way to score. Furthermore, we

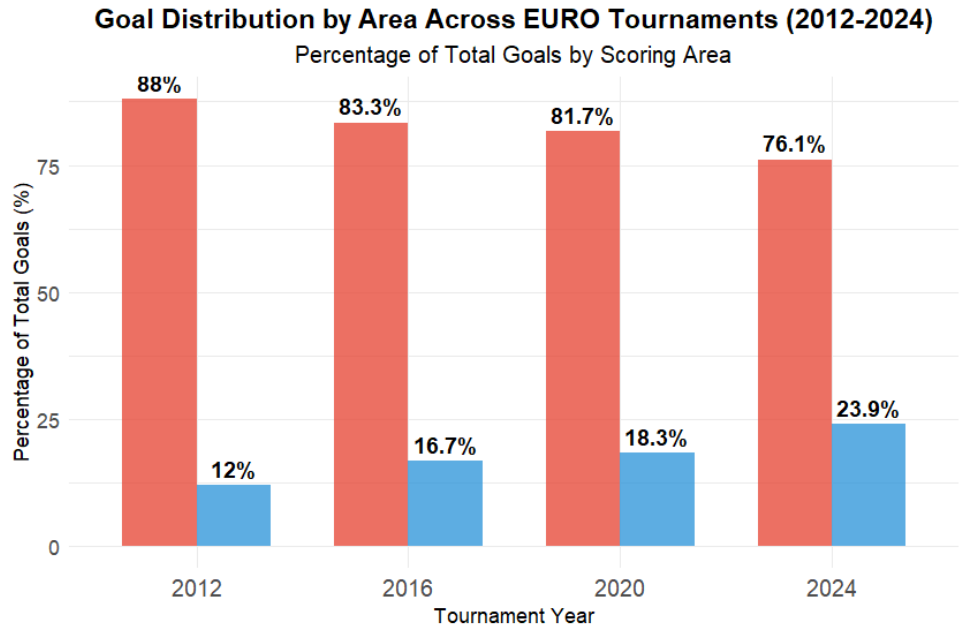

Figure 3.15: Bar Chart for Goal Distribution by Area Across

observe that scoring with headers decreases tournament by tournament. Next, we will see whether most goals are scored from inside or outside the area.

Of course, most goals have been scored inside the area, as it is easier to score from inside the area than from outside it. Furthermore, we observe that the percentages of goals scored inside the area show a very slight decrease tournament by tournament.

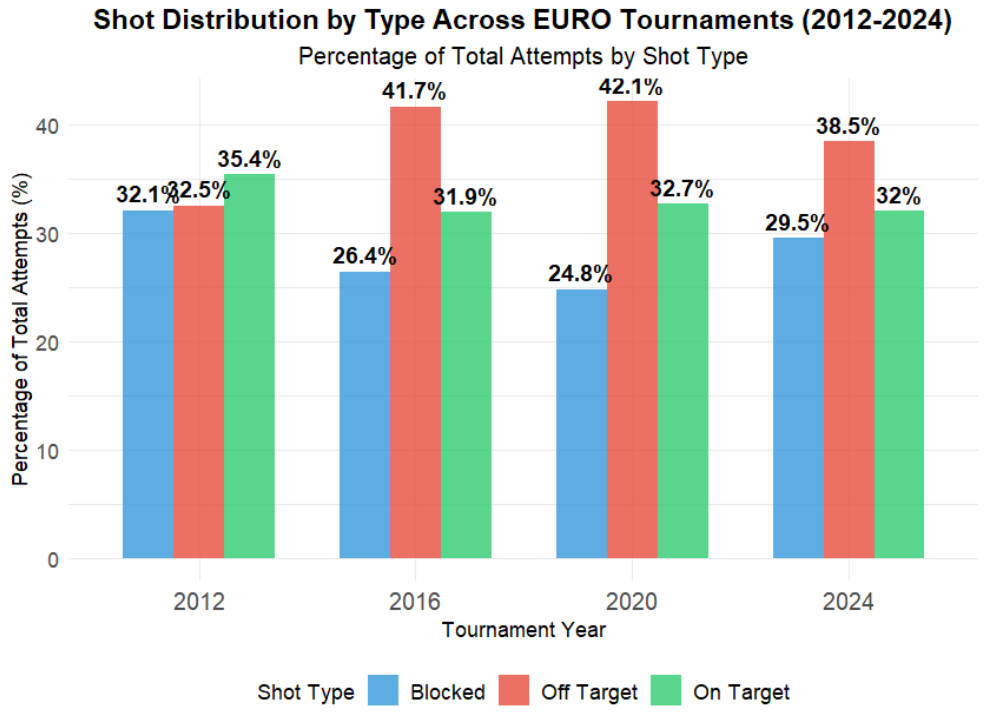

Figure 3.16: Bar Chart for Shot Distribution by Type Across

Regarding shots, we now observe that in 2012 there was a higher percentage of shots on target, whereas in the subsequent tournaments we see the opposite trend.

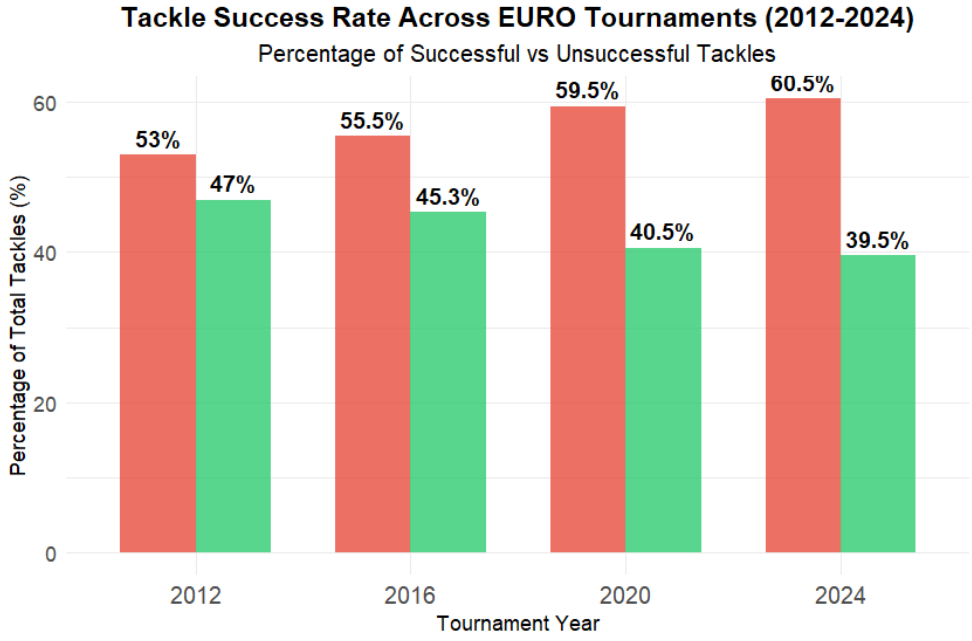

Figure 3.17: Bar Chart for Tackle Success Rate Across

Furthermore, we see that most shots are either on or off target and are not blocked by the opposing defense.

From the above graph, we can see that the percentage of tackles lost increases tournament by tournament, while the percentage of tackles won decreases. This could be happening because teams are finding ways to break down defenses, as we also saw earlier that teams are making more attacks per tournament. Additionally, this could also be influenced by VAR, as many decisions are changed due to its intervention.

3.5 Scatter plots

Scatter plots are graphical representations used to examine the potential relationship between two variables. Each point on the plot represents an observation. The position of each point in the Cartesian coordinate system indicates the values for two different variables. This method is particularly useful for revealing the relationships between variables as well as how strongly they are correlated. Scatter plots are important tools in statistical analysis, providing an immediate visual understanding of data that might not be apparent from examining the numbers alone. In these plots, it is common practise to also use a trend line. The line in a scatter plot typically represents a linear fit and is also known as a trend line or regression line. It is used to predict the direction and strength of a correlation between the two variables. This line attempts to describe the relationship that may exist between the data points and can indicate whether there is a positive, negative, or no correlation. The trend line can be calculated using various methods, with the least squares method being the most common approach. Presented below are several examples of scatter plots for various values of the correlation coefficient.

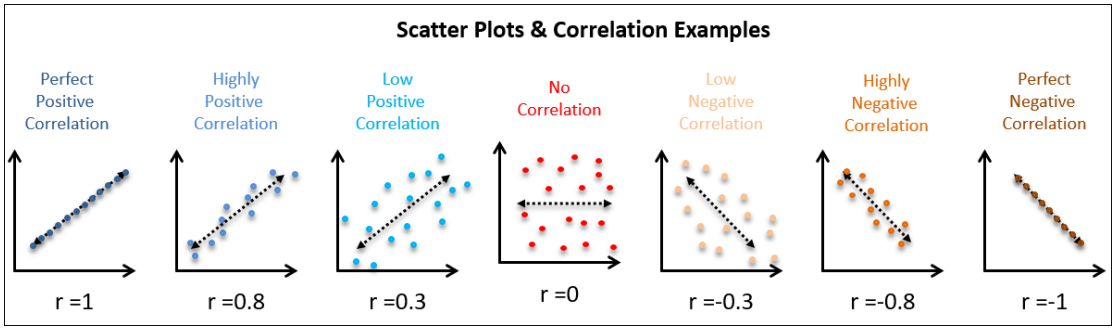

Figure 3.18: Scatter Plot for the Different Values of the Correlation Coefficient (Source: cqeacademy.com)

In our data, because they consist of data from teams from the last 4 EUROS, we want to see the real correlation in performance. For this reason, we have normalized our data, because some teams have played more matches than others. We selected several pairs of variables that are more likely to be correlated with each other, and the resulting graphs are as follows:

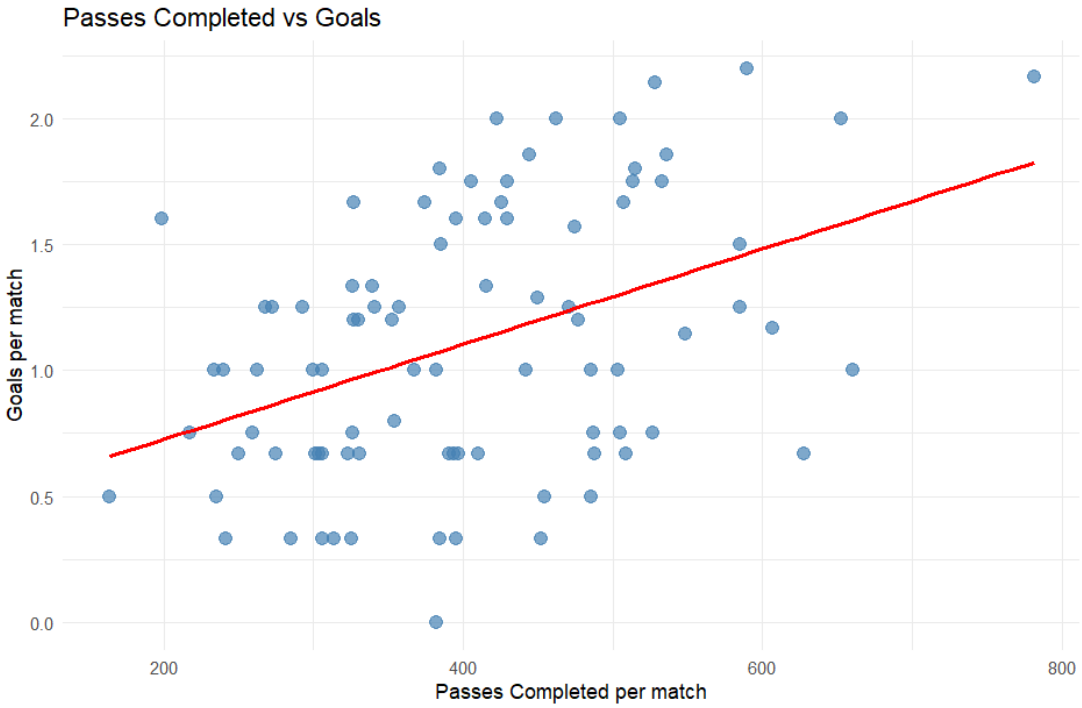

Figure 3.19: Scatter Plot for the Variables Passes Completed and Goals

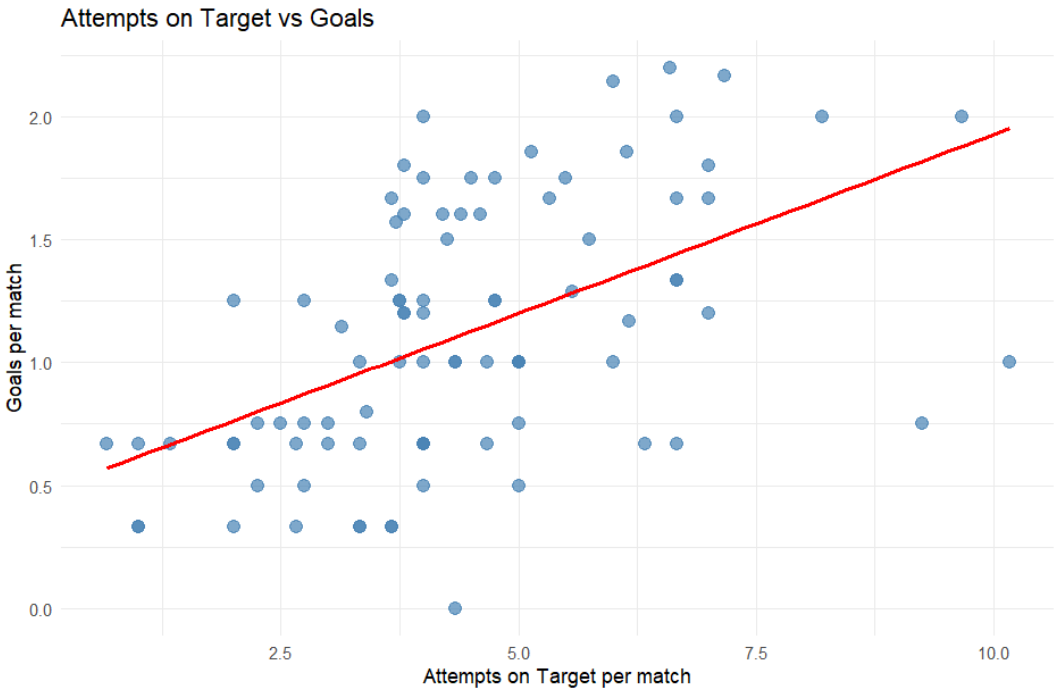

Figure 3.20: Scatter Plot for the Variables Attempts On Target and Goals

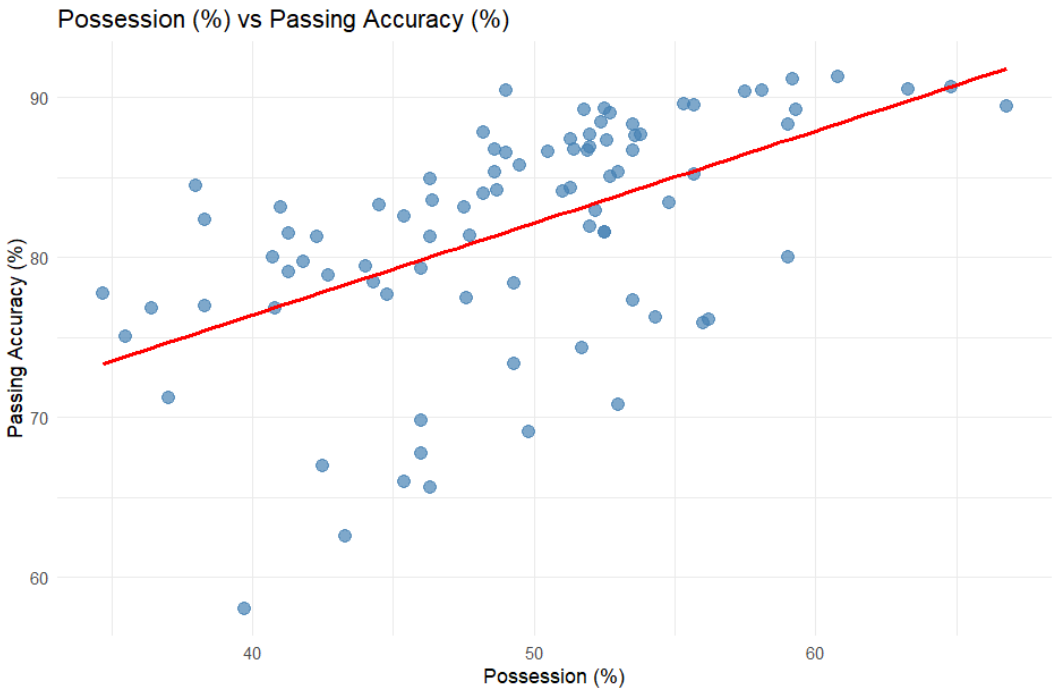

Figure 3.21: Scatter Plot for the Variables Possession (%) and Passing Accuracy (%)

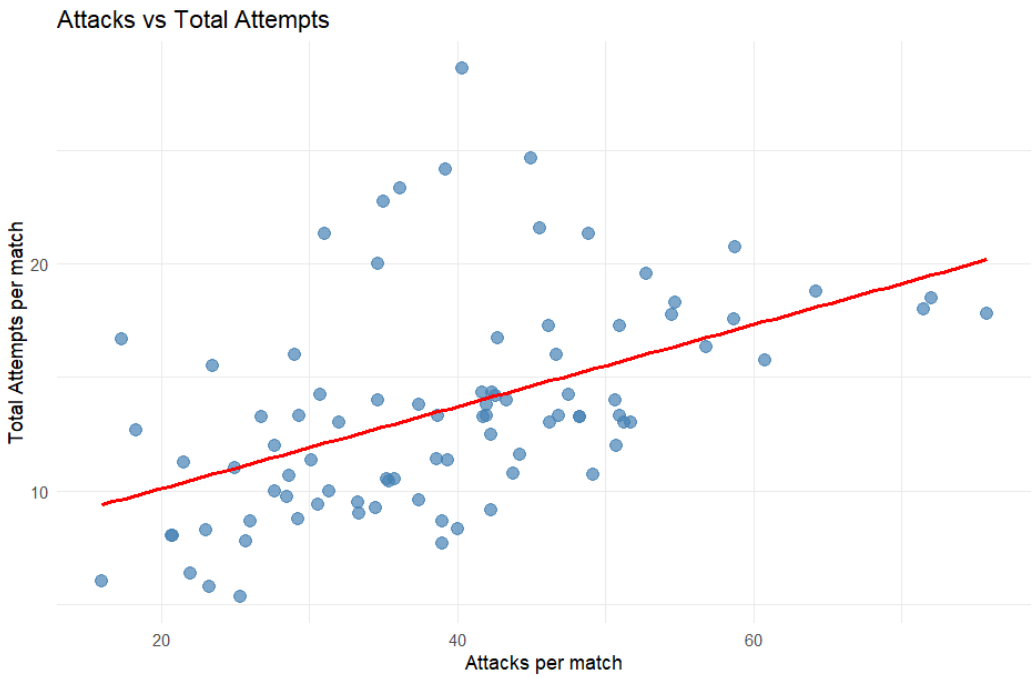

Figure 3.22: Scatter Plot for the Variables Attacks and Total Attempts

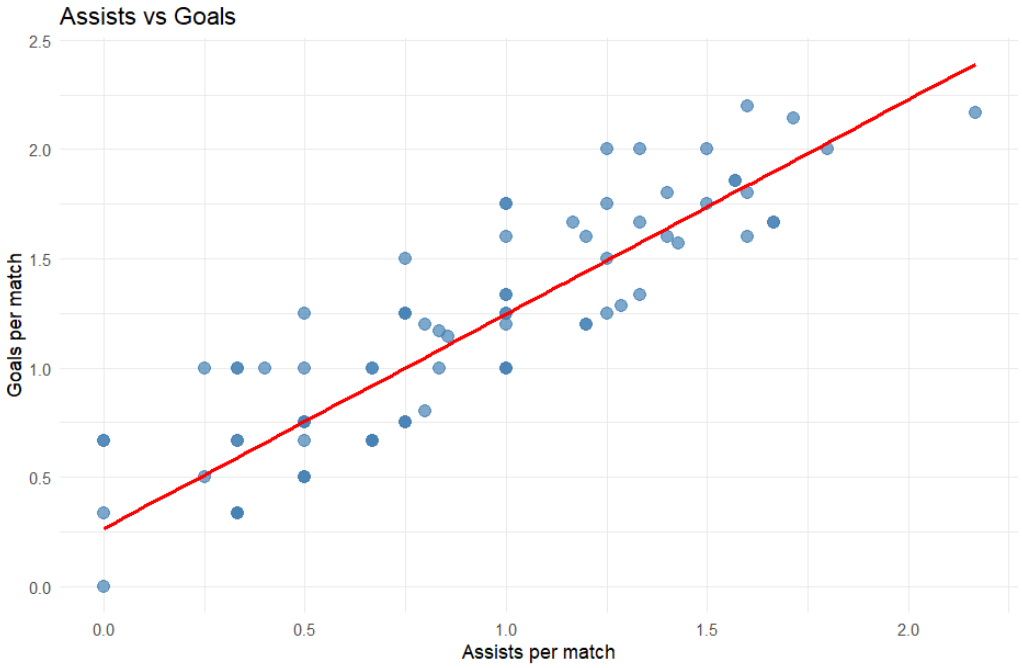

Figure 3.23: Scatter Plot for the Variables Assists and Goals

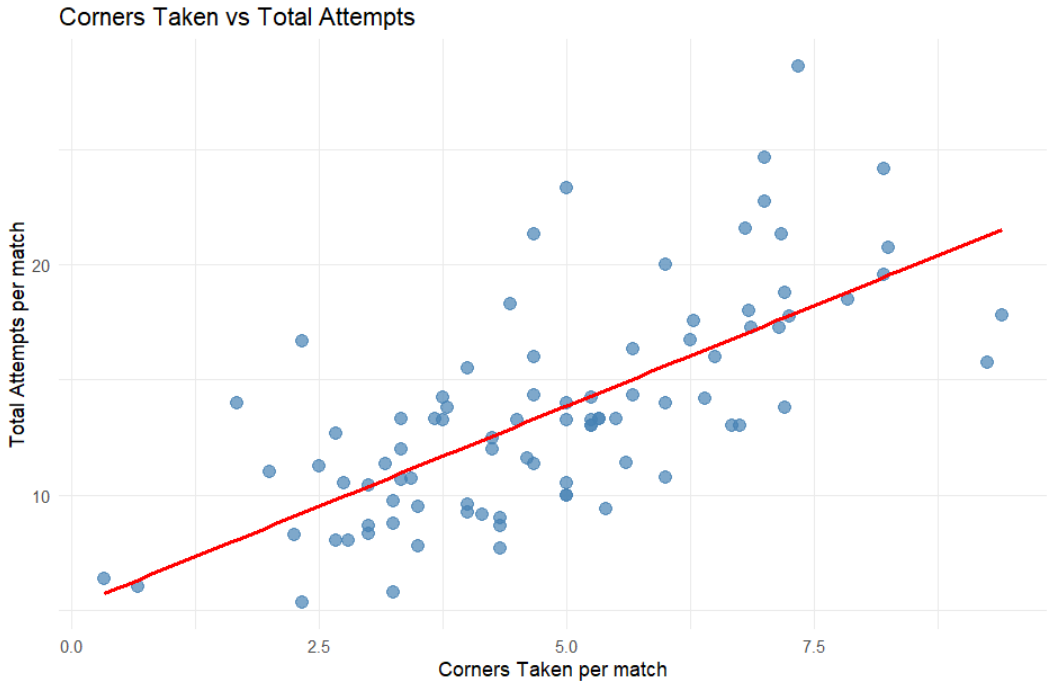

Figure 3.24: Scatter Plot for the Variables Corners Taken and Total Attempts

Based on the specific sets of variables we selected, we observe a trend of positive correlation between them. We will proceed with further tests to verify the validity of the indications we mentioned. A useful tool for measuring the relationship between variables is the Pearson correlation coefficient, which is estimated by the sample correlation coefficient and is optimally estimated in the case of normal samples. For this reason, before we discuss the correlation coefficients, contingency tables, and other tools we will use later, we will examine the normality of the variables in our sample.

3.6 Normality Test

In the statistical analysis of any dataset, assessing the normality of variables constitutes a crucial step for evaluating the nature of the data and, particularly, the statistical methods that may subsequently be applied. For this purpose, various normality tests can be examined, each with its own characteristic properties suited to different types of data. The Shapiro-Wilk test, proposed by Shapiro and Wilk (1965), is a well-known and powerful regression test of normality. It gives good results for small sample sizes, and its accuracy is claimed for sample sizes from 3 to 5,000. The Kolmogorov-Smirnov test is an empirical distribution function test that requires the distribution to be completely specified with known parameters. When the parameters of the normal distribution are not known in advance and must be estimated from the data, the Lilliefors test – a modification of the Kolmogorov-Smirnov test – is more appropriate, because using the original KS statistic in such situations tends to reduce the probability of a Type I error. Other tests, such as the Anderson-Darling and the Jarque-Bera tests, focus on different aspects of data distribution. The Anderson-Darling test gives more weight to the tails of the distribution compared to the Cramér-von Mises test, while the Jarque-Bera test is based on the sample skewness and kurtosis. *(Ahmad & Sherwani, 2015)*

As previously mentioned, for our data we employed the Lilliefors test. The test performed is as follows:

- H_0: The data follow a normal distribution
- H_1: The data do not follow a normal distribution

First, let us examine in more detail the test we will be using. The key difference between the Lilliefors test and the classical Kolmogorov-Smirnov test is that the former does not require the mean and variance of the distribution to be known in advance. Instead, these parameters are estimated from the sample data. The test statistic for the Lilliefors test is calculated as the maximum absolute difference between the empirical cumulative distribution function of the sample and the theoretical cumulative distribution function of the normal distribution with the estimated parameters. Small values of the test statistic suggest that the empirical distribution closely matches the normal distribution, while large values indicate a significant deviation from normality.

The p-value accompanying the test statistic is used to decide whether to reject or not the null hypothesis of normality. A small p-value (typically < 0.05) suggests rejection of the null hypothesis, meaning that the data distribution is not normal.

The tests we performed using R showed that at a 5% significance level, the vast majority of our variables have a p-value less than 0.05, providing strong evidence that most of our quantitative variables do not follow a normal distribution. Below are the results from our complete dataset. And here we used the columns which are divided by the games played by each team.

Variables	Lilliefors Statistic	Lilliefors p-value
Red Cards	0.4846	0.0000
Saves from Penalties	0.4812	0.0000
Own Goals Conceded	0.4473	0.0000
Penalties Scored	0.4257	0.0000
Other	0.4380	0.0000
Goals outside area	0.2780	0.0000
Lost	0.2755	0.0000
Drawn	0.2733	0.0000
Clean Sheets	0.2099	0.0000
Left Foot	0.2142	0.0000
Won	0.2418	0.0000
Matches Played	0.2411	0.0000
Head	0.2385	0.0000
Attempts on Target	0.1070	0.0146
Blocked	0.1046	0.0187
Assists	0.1057	0.0166
Right Foot	0.1069	0.0148
Total Attempts	0.1332	0.0006
Attempts off Target	0.1052	0.0175
Fouls Suffered	0.0609	0.5839
Free-Kicks Taken	0.0676	0.4123
Crosses Attempted	0.1023	0.0238
Corners Taken	0.0519	0.8102
Goals	0.1261	0.0015
Goals inside area	0.0986	0.0344
Attacks	0.0480	0.8856
Clearances Attempted	0.1440	0.0008
Fouls Commited	0.1025	0.0231
Passes Attempted	0.0521	0.8060
Goals Conceded	0.1253	0.0015
Passes Completed	0.0745	0.2657
Yellow Cards	0.0913	0.0671
Crosses Completed	0.0823	0.1498
Tackles Won	0.1010	0.0271
Passing Accuracy (%)	0.1066	0.0151
Saves	0.1164	0.0050
Offsides	0.1124	0.0080
Crossing Accuracy (%)	0.0913	0.0675
Tackles	0.0740	0.2758
Possession (%)	0.0791	0.1910
Tackles Lost	0.0914	0.0669

Table 3.3: Results of Normality Tests for the Per Match Data

We observe that the variables Crossing Accuracy, Tackles, Tackles Lost, Possession, Crosses Completed, Yellow Cards, Passes Completed, Passes Attempted, Attacks, Corners Taken, Fouls Suffered and Free-kicks Taken follow a normal distribution because their p-value is greater than 5% significant level. All other variables do not

follow a normal distribution. The above results are very useful for the development of this study, as they will help us later to decide which statistical tests to use in each case.

To conclude, a very interesting area that the several researchers have explored in the past is the distribution of goals in a match. Specifically, as shown by Karlis D. and Ntzoufras I. (2000), goals follow a Poisson distribution, a finding which paves the way for using specific methods to predict them. We perform a x^2 goodness-of-fit test on our data to investigate whether such a finding is confirmed in our own case, with the results shown below.

Test	P-value
x^2 goodness-of-fit	0.0273

Table 3.4: Poisson Distribution Test for Teams Goals per Match

Based on the test, we see that the resulting p-values is less than 5%, which leads us to conclude that in our specific case, the goals scored by teams per match do not follow a Poisson distribution.

CHAPTER 4

4. Statistical tests and correlations between the variables

4.1 Introduction

Continuing our analysis and our effort to identify the factors influencing European football, we are now called upon to examine a dataset with multiple characteristics. This is a positive development, as there are numerous factors that affect a tournament-especially when discussing the UEFA Euro, where every detail can be scrutinized. On the other hand, this large volume of information creates the need for an initial "screening" of these characteristics. The goal is to identify which ones most significantly impact the subjects of our study, as well as to find those that most strongly correlated with one another. In this chapter, we will investigate the correlations among the variables under examination. Subsequently, we will perform hypothesis tests on the means of our variables across various elements of the tournament.

4.2 Correlations between the variables

In statistics, the description of the relationship between variables is called correlation. The types of correlations we encounter are linear and non-linear. In the first case, as the values of one variable increase, the values of the other variable either increase or decrease. In the second case, we have relationships between variables that cannot be described linearly and are modeled using different patterns.

The key characteristics of correlations are their type, direction and strength. To investigate these correlations, we use correlation coefficients, which are selected based on the nature of our data.

Correlation coefficients are statistical measures used to quantify the relationship between two or more variables. Depending on the nature of the data and the relationship being studied, different types of correlation coefficients are used. The most common coefficients are Pearson's, Spearman's, and Kendall's.

Pearson Correlation Coefficient (r): The Pearson correlation coefficient measures the linear relationship between two random variables. It is denoted by the letter ρ and is calculated as follows:

$\rho_{XY} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$, where $cov(X,Y)$ is the covariance of X and Y.

The sample correlation coefficient, denoted by the letter r, is calculated using the estimators of the above quantities as follows:

$r_{XY} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$, where n is the sample size, x_i and y_i are the i-th sample observations for two variables and $\bar{x}$ and $\bar{y}$ are the sample means of the two variables.

Regarding the interpretation of the coefficient, we have the following:

- If $r = \pm 1$, there is a perfect linear correlation.
- If $-0.3 \leq r \leq 0.3$, there is no linear correlation. However, this does not mean there is no other type of correlation between the two variables.
- If $-0.5 \leq r \leq -0.3$ or $0.3 \leq r \leq 0.5$, there is a weak linear correlation.
- If $-0.7 \leq r \leq -0.5$ or $0.5 \leq r \leq 0.7$, there is a moderate linear correlation.
- If $-0.8 \leq r \leq -0.7$ or $0.7 \leq r \leq 0.8$, there is a strong linear correlation.
- If $-1 \leq r \leq -0.8$ or $0.8 \leq r \leq 1$, there is a very strong linear correlation. *(Androulakis, 2024)*

It is used for linear relationships and is suitable for continuous, normally distributed data, but it is sensitive to outliers. Among its advantages are the fact that it is simple and easy to calculate and interpret, while it provides an immediate indication of the strength and direction of the linear relationship. Its limitations include the fact that it is not suitable for non-linear relationships and it is influenced by the presence of outliers.

Spearman's Rank Correlation Coefficient (ρ or r_s): Spearman's correlation coefficient is used to assess the relationship between two variables, regardless of linearity. It is based on the ranks of the data and is calculated as follows:

$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$, where $d_i = Rg(x_i) - Rg(y_i)$, with $Rg(x_i)$ and $Rg(y_i)$ being the ranks of the sorted observations x_i and y_i respectively.

This coefficient also takes values from -1 to 1 and is interpreted as follows:

- If $r_s = 1$, then $d_i = 0$ and there is a positive dependence between X and Y.
- If $r_s = -1$, then there is a negative dependence between X and Y.
- If $r_s = 0$, then there is no dependence between X and Y. *(Khawla Ali Abd Al-Hameed, 2022)*

It is used for non-linear, monotonic relationships and for ordinal data. It is suitable for data that does not follow a normal distribution. Its advantages include robustness to outliers and the ability to detect non-linear relationships. Its limitations are found in the fact that it can lose information related to the absolute difference between values, as it is based on rankings.

4.2.1 Correlations between Quantitative Variables

The correlations between the quantitative variables are presented below. These were tested using the Pearson method for those that follow a normal distribution. Otherwise, for those that do not follow a normal distribution, we used Spearman's correlation. All quantitative variables were selected, and due to their large number, a heatmap was used to facilitate the reading of the results. Thus, we see the heatmap of correlations between quantitative variables:

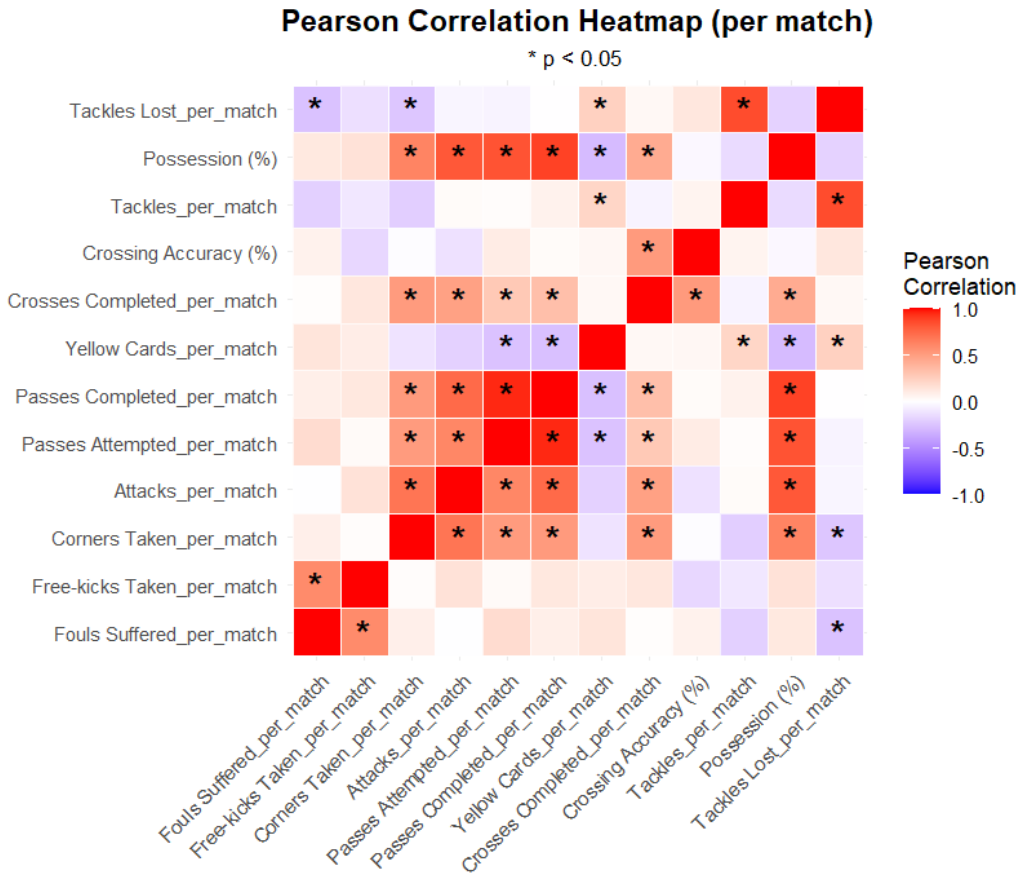

Figure 4.1: Statistically significant (*) Pearson correlations

From the Pearson correlation matrix between technical football variables per match, several strong and interesting statistical relationships emerge that reflect fundamental characteristics of modern football. The most notable correlation is observed between Tackles per match and Tackles Lost per match, with a Pearson coefficient of r = 0.856. This exceptionally strong positive relationship suggests that teams that attempt many defensive interventions inevitably tend to lose many of them as well. This finding can be interpreted as an indicator of a specific defensive style based on a high number of interventions, high-intensity defense, and possibly lower technical accuracy in duels. It may also signal a strategy of aggressive pressing where the team attempts to regain possession higher up the pitch, albeit with the risk of failing in these attempts. In the offensive domain, we identify very strong positive correlations that characterize high-possession teams. The relationship between Passes Attempted and Passes Completed (r = 0.953) is nearly perfect, confirming that these two variables essentially measure the same characteristic: the team's overall passing volume. Possession (%) shows extremely high correlations with both Passes Completed (r = 0.886) and Passes Attempted (r = 0.830), revealing the direct and powerful connection between possession percentage and a team's passing volume. This pattern extends to other offensive variables: Possession correlates significantly with Attacks per match (r = 0.812) and with Corners Taken (r = 0.626), indicating that teams that dominate possession create more attacking phases and set-piece opportunities. Furthermore, a coherent relationship is observed between offensive parameters. Corners correlate positively with both Attacks (r = 0.690) and Crosses Completed (r = 0.515), suggesting

that attacking phases, particularly those developed from the wings, often lead to corners. The connection between Crosses Completed and Crossing Accuracy (r = 0.517) is also logical, as teams with better crossing technique complete more of these deliveries. In the defensive domain, the negative correlation between Yellow Cards and Possession (r = -0.291) reveals an interesting pattern: teams with less possession tend to receive more yellow cards. This is likely because they find themselves more frequently in defensive phases and compelled into more interventions that may result in bookings. The negative correlation between Fouls Suffered and Tackles Lost (r = -0.261) offers an interesting perspective, suggesting that teams that lose fewer tackles perhaps suffer more fouls, possibly as a result of the opposition attempting to counter their better defensive organization. Below, we will examine the Spearman correlations

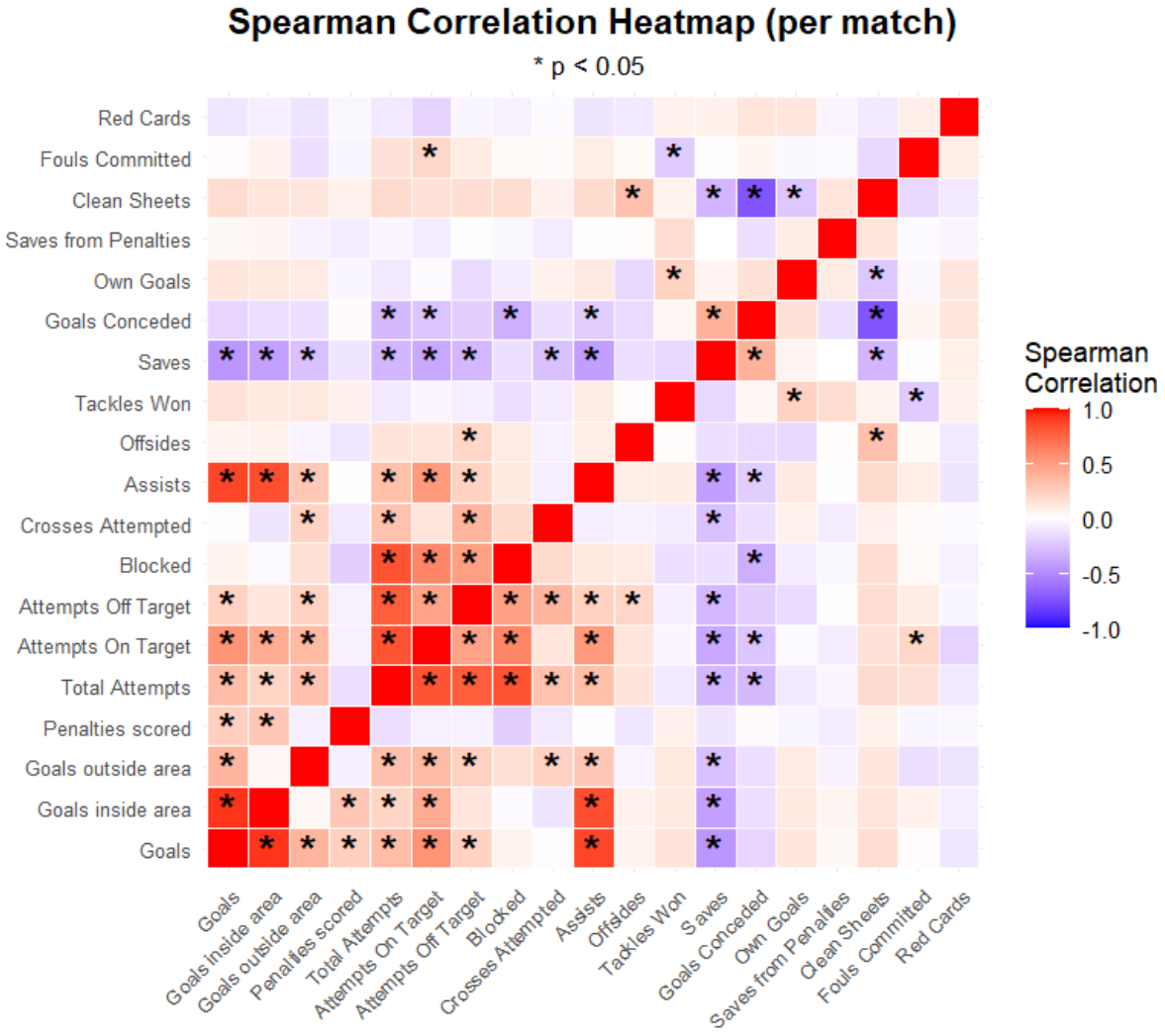

Figure 4.2: Statistically significant (*) Spearman correlations

of the remaining variables that do not follow a normal distribution.

From the Spearman correlation matrix between various technical variables, many interesting statistical relationships emerge that reveal fundamental patterns in how attacking play develops and concludes in modern football. Among the strongest correlations, the one between Goals and Goals inside the area stands out, with a Spearman coefficient of r = 0.927. This almost perfect positive relationship suggests that the overwhelming majority of goals are scored from inside the area, confirming the

statistical reality that the most effective chances are created close to the goal. Also, the very strong correlation between Goals and Assists (r = 0.878) reveals the fundamental relationship between chance creation and finishing, as teams that score many goals also tend to have many assists. In the same context, Assists show an equally strong correlation with Goals inside the area (r = 0.846), emphasizing the importance of creating chances at close distances from the goal. In the area of attacking attempts, strong positive correlations are also observed. Shots on Target are strongly linked to Total Attempts (r = 0.827), showing that teams that take many total attempts also tend to have many shots on target. Similarly, Blocked Shots are strongly correlated with Total Attempts (r = 0.828), indicating that teams with a high volume of shots often see many of their attempts blocked. Among the moderate to strong correlations, we find that Goals have a moderate positive correlation with Shots on Target (r = 0.553), confirming that the ability to score goals is linked to the ability to put shots on target. Also, Goals show a correlation with Goals outside the area (r = 0.387), indicating that goals from long distances contribute significantly but not dominantly to overall scoring. On the other hand, a negative correlation is observed between Goals and Saves (r = -0.454), showing that teams that score many goals tend to face fewer saves, likely because they have possession and limit the opposing team's chances. Assists, in turn, show a moderate positive correlation with Shots on Target (r = 0.521), indicating that they are linked to the ability to create dangerous chances that lead to shots on target. Among the significant negative correlations, the very strong negative correlation between Clean Sheets and Goals Conceded (r = -0.736) stands out, which is completely logical, since the fewer goals a team concedes, the more clean sheets it has. At the same time, Saves show a positive correlation with Goals Conceded (r = 0.399), showing that teams that concede many goals also tend to have many saves, as they are often under pressure. Regarding the interesting relationships between various types of shot attempts, we observe that Total Attempts are very strongly correlated with Shots off Target (r = 0.793), indicating that teams with a high shot volume tend to also have many shots off target. Also, Shots on Target have a strong correlation with Blocked Shots (r = 0.616), a fact which suggests that shots on target often lead to blocked attempts as well.

The next step following the above is to examine the statistical significance of the correlations between the variables. If we look closely at the above two heatmaps, we will notice asterisks in some of the tiles. This tells us that the correlations of these variables are statistically significant at a 5% significance level.

Before we proceed to the correlations between quantitative and qualitative variables, a crucial connecting link can be the study of the correlations between our variable and the match outcome (Won, Drawn, Lost). In this case, the outcome variables are treated as quantitative. Specifically, we will divide them by the number of matches each team has played, effectively normalizing them like all others columns. In this particular case, we will only calculate the correlations between the variable Won and all the others, because if we have a positive correlation with the variable Won, we will have a negative correlation with the variable Loss. Therefore, below, only the correlations for the variable Won will be displayed. Thus, we have the following:

Variables	Won	Variables	Won
Goals	0.662	**Free-Kicks**	-0.073
Goals Inside	0.653	**Attacks**	0.295
Goals outside	0.182	**Assists**	0.684
Penalties scored	0.139	**Corners**	0.310
Total Attempts	0.259	**Offsides**	0.183
Attempts on target	0.347	**Tackles**	-0.083
Attempts off target	0.169	**Saves**	-0.438
Blocked	0.163	**Clean Sheets**	0.555
Passing Accuracy	0.171	**Fouls Committed**	-0.104
Possession	0.278	**Fouls Suffered**	0.096
Crossing Accuracy	0.133	**Yellow Cards**	-0.198
Red Cards	-0.192	**Goals Conceded**	-0.521
Own Goals	-0.018	**Saves from Penalties**	0.167

Table 4.1: Spearman's rank correlation coefficients (ρ) between the variables and the Won variable

Based on the correlation coefficients, we observe that assists exhibit the strongest correlation (0.684), making goal-scoring opportunities the most important statistical indicator for victory. This underscores the importance of a collective and creative approach. Goals (0.662) and especially goals from inside the area (0.653) also show a strong correlation, which was expected, as goals, particularly those from high-quality chances, directly lead to wins. Furthermore, clean sheets (0.555) display a very strong correlation, confirming that defensive reliability and preventing goals are equally critical to scoring effectiveness. On the other hand, indicators such as corners (0.31), shots on target (0.347), possession (0.278), and attacks (0.295) reflect approach and pressure. They are positively correlated with winning, but not as directly as final effectiveness, such as goals and assists. A team can have high values in these indicators without necessarily winning if the quality in the final action is lacking. A significant negative correlation is observed with goals conceded (-0.521), which constitutes the most important factor for defeat, acting as the inverse of clean sheets. Concurrently, saves (-0.438) show a moderately negative correlation. It is important to note that this does not imply poor goalkeeper performance, but rather that teams forced to make many saves usually face sustained pressure and are likely to lose. Finally, cards (yellow -0.198 and red -0.192) display a weak negative correlation. Aggression and numerous dismissals slightly reduce the chances of winning, likely due to forced tactical changes or the team's reduced numerical strength on the pitch. Precisely the inverse relationships characterize losing outcomes. The following table presents the statistical significance levels (p-values) for the correlations discussed above.

Variables	Won	Variables	Won
Goals	0.000	**Free-Kicks**	0.395
Goals Inside	0.000	**Attacks**	0.006
Goals outside	0.087	**Assists**	0.000
Penalties scored	0.613	**Corners**	0.004
Total Attempts	0.033	**Offsides**	0.127
Attempts on target	0.001	**Tackles**	0.513
Attempts off target	0.127	**Saves**	0.000
Blocked	0.479	**Clean Sheets**	0.000
Passing Accuracy	0.140	**Fouls Committed**	0.297
Possession	0.004	**Fouls Suffered**	0.326
Crossing Accuracy	0.581	**Yellow Cards**	0.075
Red Cards	0.050	**Goals Conceded**	0.000
Own Goals	0.791	**Saves from Penalties**	0.252

Table 4.2: p-values for the Spearman's rank correlation coefficients between the variables and the Won variable.

The analysis of statistical significance reveals that at the 5% level, several variables demonstrate significant correlations with match outcomes. For wins, statistically significant correlations are found with goals scored, assists, goals from inside the area, clean sheets, goals conceded, saves, shots on target, corners, possession, and attacking plays.

4.2.2 Correlations Between Quantitative and Qualitative Variables

In this final analytical segment, we investigate the relationships between quantitative and categorical variables, utilizing Spearman's correlation coefficient as previously justified. The findings are displayed below.

Variables	Final Rank	Year	Variables	Final Rank	Year
Goals	-0.616	-0.001	**Free-Kicks**	-0.024	-0.019
Goals Inside	-0.489	-0.079	**Attacks**	-0.423	0.261
Goals outside	-0.477	0.285	**Assists**	-0.592	-0.095
Penalties scored	-0.131	0.081	**Corners**	-0.352	-0.127
Total Attempts	-0.307	-0.354	**Offsides**	-0.184	-0.194

Attempts on target	-0.401	-0.312	**Tackles**	0.009	0.666
Attempts off target	-0.242	-0.234	**Saves**	0.460	-0.149
Blocked	-0.165	-0.337	**Clean Sheets**	-0.527	-0.015
Passing Accuracy	-0.313	0.538	**Fouls Committed**	0.131	-0.357
Possession	-0.319	-0.028	**Fouls Suffered**	-0.077	-0.390
Crossing Accuracy	-0.081	-0.043	**Yellow Cards**	0.158	0.115
Red Cards	0.149	0.042	**Goals Conceded**	0.559	0.048
Own Goals	-0.014	0.292	**Saves from Penalties**	-0.206	-0.086

Table 4.3: Spearman's rank correlation coefficients (ρ) between the per-match quantitative variables and two qualitative factors: Final Rank and Tournament Year.

The aforementioned correlation matrix clearly reflects several characteristics that were also visually depicted in the box plots of the previous chapter. Specifically, the negative correlations between the quantitative variables and the qualitative 'Final Rank' indicate that as a team progresses further in the tournament (in the dataset, the ranking ascends, with 1 representing the tournament winners and 6 those eliminated in the group stage), the more negative the correlations become with all variables related to the offensive aspects of matches. This means that the stronger teams in the tournament demonstrate superior values across all these variables (Goals, Assists, Goals Inside and Outside Area, Attempts on target Passing Accuracy, Possession, Attacks, Corners, Clean Sheets), which, as observed earlier, show a strong correlation with success. We also see that the variable Goals conceded correlates positively with the Final Rank, which makes us understand that the stronger teams concede fewer goals. Furthermore, the correlations between the variables and the years in which the tournaments were held present no surprises, as most are weak. Specifically, where we observe larger positive correlations are with tackles (0.666). This means that as the tournaments progress, more tackles are being made. This could be happening due to increased intensity and pressure in the game. We also have a positive correlation with passing accuracy (0.538). This means that as the tournaments progress through the years, the teams improve in their passing accuracy. Additionally, there are some moderate negative trends, such as with Fouls Suffered (-0.390) and Fouls Committed (-0.357). This means that tournament after tournament, fewer fouls are being committed. Finally, we observe a negative correlation between the year and total attempts (-0.354), from which we conclude that as the tournaments progress, the teams' scoring attempts decrease.

Variables	Final Rank	Year	Variables	Final Rank	Year
Goals	0.000	0.991	**Free-Kicks**	0.827	0.860
Goals Inside	0.000	0.464	**Attacks**	0.000	0.014
Goals outside	0.000	0.007	**Assists**	0.000	0.381

Penalties scored	0.225	0.456	**Corners**	0.001	0.238
Total Attempts	0.004	0.001	**Offsides**	0.086	0.070
Attempts on target	0.000	0.003	**Tackles**	0.931	0.000
Attempts off target	0.023	0.028	**Saves**	0.000	0.165
Blocked	0.125	0.001	**Clean Sheets**	0.000	0.887
Passing Accuracy	0.003	0.000	**Fouls Committed**	0.223	0.001
Possession	0.002	0.797	**Fouls Suffered**	0.473	0.000
Crossing Accuracy	0.454	0.692	**Yellow Cards**	0.141	0.287
Red Cards	0.149	0.042	**Goals Conceded**	0.000	0.658
Own Goals	0.896	0.006	**Saves from Penalties**	0.054	0.426

Table 4.4: p-values for the Spearman correlation coefficients in Table 4.3

Here we observe that there is a statistically significant correlation at the 5% significance level between Fouls Suffered, Fouls Committed, Total Attempts, Passing Accuracy, Goals outside and Tackles with the variable Years. The finding had also been captured in the descriptive analysis we conducted. Additionally, regarding the correlation of the variables mentioned above with the Final Rank variable, we see that they are also statistically significant at a 5% significance level.

4.3 Hypothesis Tests for the Equality of Means of Two Samples

4.3.1 Introduction and Theory of Mean Value Tests

At the end of previous chapter, we conducted a test for the normality of the data. This was carried out through hypothesis testing regarding the sample's normality. In general, hypothesis tests in statistics are used to examine whether one or more parameters of one or more populations satisfy a certain hypothesis against another, which we call the alternative. These types of tests are also the most useful when examining the equality of means in two populations. Test are divided into parametric and non-parametric. Parametric tests have certain requirements for our data, such as independence, normality, and constant variance. If any of these conditions are violated, non-parametric tests are used instead. The types of such tests referred to as t-test vary, and a brief explanation is provided below with the help of an image.

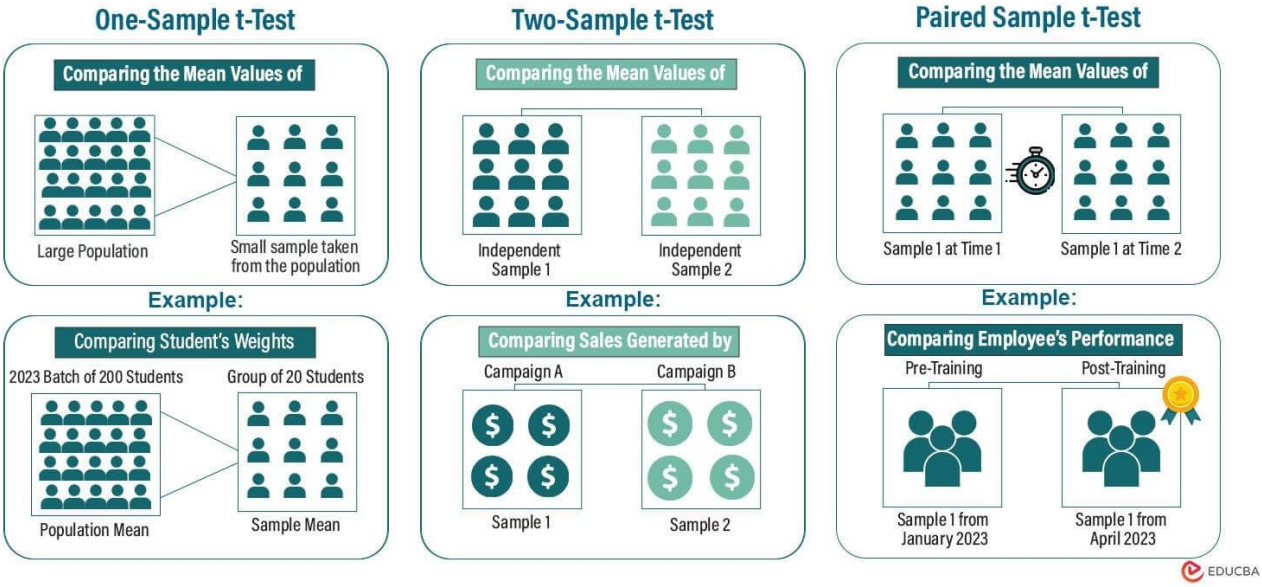

Figure 4.3: Different types of t-test (Source: https://www.educba.com/t-test-formula/)

In its most basic form, the first case we encounter is the one-sample t-test, which examines whether the mean of a population is equal to a given value. Next, in the second case, we encounter the independent samples t-test, where, as the name implies, it examines whether two independent populations have equal means. Finally, in the last form, the paired samples t-test, we have two samples from the same population, studying the effect of a certain factor before and after an intervention.

In our data, we observed that some variables follow a normal distribution. Therefore, we can use the aforementioned tests. However, since in Chapter 3 we divided the dataset into three categories: Low (Group Stage), Medium (Round of 16, Quarter-Finals), and High (Semi-Finals, Finalist, and Winner) we will use ANOVA, which will be explained below.

4.4 Tests of hypotheses for the equality of means of three samples – ANOVA

Here, in contrast to earlier, the means we want to test are more than two. Therefore, it is not possible to use the same methodology as before; instead, we, need a generalization of it. The solution to our problem is provided by tests for means involving more than two populations. Naturally, the most well-known methodology for these cases is Analysis of Variance (ANOVA). However, it has some strict assumptions, such as the normality of the data and the homogeneity of variances (homoscedasticity). The hypotheses of ANOVA are:

- H_0: All population means are equal.
- H_1: At least one population mean differs from the others.

The test is performed using the test statistic F, which, when the null hypothesis is true, follows Snedecot's F-distribution with $k-1$ and $N-k$ degrees of freedom, where $N = n1 + n2 + \cdots + nk$. *(Evaggelaras, 2024)*

4.4.1 Analysis of Variance (ANOVA) for detecting differences between group means.

Let us remind you that we are using the data which we have divided by the number of games each team has played. Thus, we apply the above method to our data, and the results we obtain are the following:

Lilliefors Test for Normality			
Variables	Medium	Low	High
Fouls Suffered	0.698	0.766	0.754
Free-kicks Taken	0.735	0.621	0.376
Corners Taken	0.187	0.259	0.402
Attacks	0.901	0.204	0.480
Passes Attempted	0.554	0.141	0.537
Passes Completed	0.963	0.051	0.447
Possession (%)	0.704	0.328	0.216

Table 4.5: Results of the Lilliefors Normality Test for variables by Final Rank category

First, we need to check if the assumption of normality holds for the three groups we mentioned earlier. We can see that for the variables Fouls Suffered, Free Kicks Taken, Corners Taken, Attacks, Passes Attempted, Passes Completed and Possession the normal distribution assumption is valid for all three groups. Immediately after, we will check for the homoscedasticity of our data.

Levene's Test for Homogeneity of Variances		
Variables	Levene F value	p-value
Fouls Suffered	0.4169	0.6604
Free-Kicks Taken	0.5132	0.6004
Corners Taken	0.3937	0.6758
Attacks	1.2236	0.2993
Passes Attempted	0.3115	0.7332
Passes Completed	1.5992	0.2081
Possession (%)	1.1242	0.3297

Table 4.6: Results of Levene's Test for Homogeneity of Variances

And here we see that for our variables, at a significance level of 5%, the hypothesis of homoscedasticity is not rejected with the help of Levene's test. Now we can use the ANOVA test since we are okay with the assumptions.

ANOVA Test		
Variables	F Value	p-value
Fouls Suffered	3.8074	0.0261
Free-Kicks Taken	1.3592	0.2624
Corners Taken	6.4065	0.0026
Attacks	11.7124	0.0000

Passes Attempted	7.7384	0.0008
Passes Completed	9.1713	0.0002
Possession (%)	6.9165	0.0016

Table 4.7: Results of the One-way Analysis of Variance (ANOVA) for determining statistically significant differences in the means between the categorization groups.

At a 5% significance level, for all variables except Free Kicks Taken we conclude that the means of the groups differ. Therefore, for the variables where this holds, we will proceed with a Post-Hoc Analysis (Tukey HSD) to determine which specific groups differ from each other.

4.4.1.1 Post-Hoc Analysis (Tukey HSD) to determine which means differ across groups

The Tukey HSD (Honest Significant Difference) test is a parametric method used to perform multiple comparisons between groups after an initial Analysis of Variance (ANOVA), as we applied.

The Analysis of Variance (ANOVA), as we saw, assesses whether there are statistically significant differences between three or more groups, based on the assumption that the data follow a normal distribution and there is homogeneity of variances. However, it does not identify which specific groups differ from each other. For this purpose, the Tukey HSD test is applied as a post-hoc test, which compares all possible pairs of groups to identify the specific differences.

The Tukey HSD method is a Family-Wise Error Rate (FWER) control method used to check the probability of false positive results (Type I errors) when multiple comparisons are performed simultaneously. As the number of comparisons increases, so does the probability of committing a Type I error.

The Tukey HSD test procedure involves the following steps:

1. **Calculation of Differences:** For every possible pair of groups, the difference between their means is calculated.
2. **Calculation of Critical Value (HSD):** A critical value (HSD) is calculated based on the Studentized Range distribution (Q), the Mean Square Error (from the ANOVA), and the group sizes. This value acts as a universal significance threshold.
3. **Comparison:** The absolute difference between the means of each pair is compared to the HSD critical value.
4. **Decision:** If the absolute difference between two means is greater than or equal to the HSD critical value, then that specific pair of groups is considered to have a statistically significant difference. *(Abdi-Williams, 2010)*

So, the Tukey HSD method helps us to identify statistically significant differences between specific group pairs after an initial ANOVA. The results follow:

POST-HOC ANALYSIS (Tukey HSD)		
Categories	Diff	p-value
Fouls Suffered		
Low-High	-1.1440	0.2235
Medium-High	-1.8170	0.0202
Medium-Low	-0.6729	0.4179
Corners Taken		
Low-High	-1.8616	0.0025
Medium-High	-0.8820	0.2140
Medium-Low	0.9796	0.0552
Attacks		
Low-High	-16.5396	0.0000
Medium-High	-9.3300	0.0174
Medium-Low	7.2096	0.0229
Passes Attempted		
Low-High	-137.6795	0.0006
Medium-High	-110.9711	0.0051
Medium-Low	26.7083	0.6001
Passes Completed		
Low-High	-141.5979	0.0001
Medium-High	-98.0046	0.0081
Medium-Low	43.5933	0.2107
Possession (%)		
Low-High	-7.1375	0.0011
Medium-High	-5.0425	0.0216
Medium-Low	2.0950	0.3418

Table 4.8: Results of the Post-hoc analysis using the Tukey HSD method for variables that showed a statistically significant difference in the ANOVA

Based on the above, we see for the variable "Fouls Suffered" that at a 5% significance level, the High and Medium groups differ in their mean values. Therefore, based on their difference, we can say that the High teams, on average, suffer more fouls compared to the Medium teams. Also, for this variable, the Low teams do not differ from the other two groups.

For the variable "Corners Taken", we can say that the Low teams and the High teams differ. According to their difference, we can say that the High teams, on average, take 1.8616 more corners. Otherwise, there is no other statistically significant difference for this variable.

The next variable is "Attacks". Here, all groups differ from each other. According to their differences, we can say that the High teams, on average, make 16.5396 more attacks than the Low teams. Furthermore, the High teams make, on average, 9.33 more attacks than the Medium teams and the Medium teams make, on average, 7.2096 more attacks than the Low teams.

For pass attempts, we see that the teams in the high group have on average 137.67 more attempts per match compared to the teams in the low group. Also, the high group of

teams has on average 110.97 more pass attempts per match compared to the medium teams. The difference between medium and low teams is not statistically significant. Also, for completed passes we see almost the same results as with pass attempts regarding the significance of the differences. The high teams have on average 141.5979 more completed passes compared to the low teams, and on average 98 more completed passes than the medium teams.

Finally, we will discuss the variable "Possession", where we see a statistically significant difference in the pairs Low-High and Medium-High. Based on the difference, we can say that the High teams have, on average, 7.1375% more possession than the Low teams and, on average, 5.0425% more possession than the Medium teams.

4.4.2 Kruskal-Wallis for detecting differences between group means.

In some cases, the one-way ANOVA test cannot be performed because its assumptions are not met. For this reason, we will use the non-parametric equivalent of ANOVA, the Kruskal-Wallis test, to identify the differences in the mean values. Subsequently, we will use Dunn's test with the Holm correction to determine how our variables differ based on the result.

The Kruskal-Wallis test is the non-parametric equivalent of the one-way ANOVA. It is used to determine if there are statistically significant differences between the medians of three or more independent groups. It is applied when the assumptions for ANOVA (normality, homogeneity of variances) are violated.

- H_0: All k samples originate from the same population, or equivalently, the medians of the groups are equal.
- H_1: At least one sample originates from a population with a different median. In other words, not all group medians are equal.

The test statistic H is calculated as:

$$H = \frac{12}{n(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(n+1)$$

where k is the number of groups, n_i is the number of observations in the i-th group, n is the total number of observations and R_i is the sum of the ranks for the i-th group. we reject the hypothesis that all groups are the same if our calculated H value is greater than the critical value from the Chi-square table with k-1 degrees of freedom and a significance level α. *(Evaggelaras, 2024)*

Therefore, we apply the above method to some of the variables for which it is meaningful. We remind you that we are using the variables that we divided by the number of matches each team played.

Kruskal-Wallis Test		
Variables	H Value	p-value
Goals	30.9311	<0.001
Total Attempts	11.4187	0.0033
Attempts off Target	8.0723	0.0177
Attempts on Target	14.1591	<0.001
Assists	26.6097	<0.001
Offsides	2.1833	0.3357
Passing Accuracy (%)	10.5159	0.0052
Goals Conceded	26.2826	<0.001

Table 4.9: Results of the Non-Parametric Kruskal-Wallis Test for variables where the assumptions for ANOVA were violated.

According to the above results at a 5% significance level, we see that there are statistically significant differences between the team groups in all variables except for offsides. Therefore, for the variables where there are differences between the groups, we will conduct an additional test to determine where exactly these differences occur.

4.4.2.1 Dunn's test with Holm correction for finding the ranking of medians across team groups.

The next logical question that arises in these cases is which mean values change and how they differ depending on the outcome. To answer such a question, there are some methodologies that can help us. The Dunn's test is a non-parametric method used for performing multiple comparisons between groups following an initial Kruskal-Wallis analysis, as we applied. As we saw, the Kruskal-Wallis analysis evaluates whether there are statistically significant differences between three or more groups, without assuming that the data follow a normal distribution. However, it does not identify which specific groups differ from each other. For this purpose, the Dunn's test is applied as a post-hoc test, which compares pairs of groups to pinpoint the specific differences. The Holm correction is a method for adjusting for multiple comparisons, used to control the probability of obtaining false positive results (Type I errors). As the number of comparisons increases, so does the probability of a Type I error.

The Holm correction procedure involves the following steps:

- Ranking: The p-values from the pairwise comparisons are ranked in ascending order.
- Multiplication: The first p-value is multiplied by the total number of tests, the second p-value is multiplied by the total number of tests minus one, and so on.
- Comparison: Each adjusted p-value is compared to the significance level α (e.g., 0.05) to determine if the comparisons are statistically significant.

In summary, the method we will use, the Dunn's test with the Holm correction, allows us first to identify statistically significant differences between specific pairs of groups

following an initial Kruskal-Wallis analysis, and then to adjust the p-values for multiple comparisons, thereby reducing the risk of false positive results. *(Dinno, 2015)*

Dunn's test with the Holm correction		
Categories	Z Value	p-value
Goals		
High-Low	5.1484	<0.001
High-Medium	2.0303	0.0423
Low-Medium	-4.1143	<0.001
Total Attempts		
High-Low	3.1449	0.0050
High-Medium	3.0650	0.0044
Low-Medium	-0.2373	0.8124
Attempts off Target		
High-Low	2.7763	0.0165
High-Medium	2.3528	0.0373
Low-Medium	-0.6497	0.5159
Attempts on Target		
High-Low	3.7471	<0.001
High- Medium	2.8347	0.0092
Low-Medium	-1.3020	0.1929
Assists		
High-Low	5.0252	<0.001
High- Medium	2.6244	0.0087
Low-Medium	-3.2142	0.0026
Passing Accuracy (%)		
High-Low	2.9583	0.0093
High- Medium	1.0801	0.2801
Low-Medium	-2.4720	0.0269
Goals Conceded		
High-Low	-5.0185	<0.001
High- Medium	-2.7049	0.0068
Low-Medium	3.1053	0.0038

Table 4.10: Results of the Post-hoc Dunn's test with Holm correction following Kruskal-Wallis

Using the method we described, we obtain the results presented above. In summary, we see that for goals, there are statistically significant differences across all three groups. Specifically the results reveal a clear hierarchical pattern in goal-scoring efficiency: High > Medium > Low. High-performing teams score the most goals per match, Medium-performing teams score significantly fewer goals than High performers but significantly more than Low performers, Low-performing teams score the fewest goals per match. We draw the same conclusion for assists as well. For total attempts, we can say that the results reveal a differentiated pattern in shooting attempts: High > Medium = Low. High-performing teams generate significantly more shooting attempts than both other groups, Medium and Low-performing teams show similar levels of total attempts per match. The key distinction is between High performers versus all others. We observe the same pattern for both attempts off target and attempts on target. For passing accuracy, we can say that High and Medium-performing teams show similar levels of

passing accuracy. Low-performing teams have significantly worse passing accuracy than both other groups. Finally, for goals conceded we have that the results reveal a clear hierarchical pattern in defensive performance: High < Medium < Low. High-performing teams concede the fewest goals per match. Medium-performing teams concede significantly more goals than High performers but significantly fewer than Low performers. Low-performing teams concede the most goals per match.

4.4.3 Analysis of Variance (ANOVA) of variables by tournament year

This section concerns the comparison of the mean values of our variables to test their equality, based on the year of the tournament. We would like to remind you that our data comes from the EURO tournaments from 2012 up to 2024. For this analysis, we are using the data that has been aggregated by the matches each team has played. We will begin by examining whether the assumption of normality holds for the variables in these four years.

Lilliefors Test for Normality				
Variables	2012	2016	2020	2024
Fouls Suffered	0.638	0.797	0.699	0.407
Free-Kicks	0.173	0.529	0.591	0.631
Corners	0.365	0.355	0.738	0.997
Attacks	0.956	0.547	0.851	0.692
Passes Attempted	0.398	0.551	0.582	0.270
Passes Completed	0.627	0.328	0.645	0.173
Possession (%)	0.394	0.577	0.264	0.264

Table 4.11: Results of the Lilliefors Normality Test for variables by tournament year.

We can see that for the variables Fouls Suffered, Free Kicks Taken, Corners Taken, Attacks, Passes Attempted, Passes Completed and Possession the normal distribution assumption is valid for all years. Immediately after, we will check for the homoscedasticity of our data.

Levene's Test for Homogeneity of Variances		
Variables	Levene F value	p-value
Fouls Suffered	1.3354	0.2684
Free-Kicks Taken	0.2785	0.8407
Corners Taken	0.2406	0.8678
Attacks	0.4072	0.7482
Passes Attempted	0.4026	0.7515
Passes Completed	0.2873	0.8344
Possession (%)	0.3709	0.7741

Table 4.12: Results of Levene's Test for Homogeneity of Variances for comparison between years.

And here we see that for our variables, at a significance level of 5%, the hypothesis of homoscedasticity is not rejected with the help of Levene's test. Now we can use the ANOVA test since we are okay with the assumptions.

Anova Test		
Variables	F Value	p-value
Fouls Suffered	6.3835	0.0006
Free-Kicks Taken	8.2508	0.0001
Corners Taken	1.2284	0.3045
Attacks	2.4689	0.0676
Passes Attempted	5.5591	0.0016
Passes Completed	1.7111	0.1709
Possession (%)	0.0553	0.9828

Table 4.13: Results of the One-way Analysis of Variance (ANOVA) for determining statistically significant differences in the means between the four tournament years.

At a 5% significance level, we conclude that for the variables Fouls Suffered, Free Kicks Taken, and Passes Attempted, the means across the years in which the EURO tournaments are held differ. Therefore, for the variables where this holds true, we will proceed with a Post-Hoc Analysis (Tukey HSD) to determine which specific groups differ from each other.

POST-HOC ANALYSIS (Tukey HSD)		
Categories	Diff	p-value
Fouls Suffered		
2016-2012	-1.2457	0.2720
2020-2012	-2.3424	0.0053
2024-2012	-2.7253	0.0008
2020-2016	-1.0967	0.2854
2024-2016	-1.4796	0.0821
2024-2020	-0.3828	0.9238
Free-Kicks Taken		
2016-2012	3.3449	0.0000
2020-2012	2.0168	0.0254
2024-2012	1.2924	0.2592
2020-2016	-1.3282	0.1547
2024-2016	-2.0526	0.0081
2024-2020	-0.7244	0.6554
Passes Attempted		
2016-2012	-144.7897	0.0013
2020-2012	-53.3475	0.4907
2024-2012	-95.6626	0.0602
2020-2016	91.4422	0.0389
2024-2016	49.1271	0.4651
2024-2020	-42.3151	0.5914

Table 4.14: Results of the Post-hoc analysis using the Tukey HSD method for variables that showed a statistically significant difference in the ANOVA by year.

Based on the above, we observe for the variable “Fouls Suffered” that, at a 5% significance level, the 2012 and 2020 editions differ in their mean values. Therefore, based on their difference, we can say that in 2012, teams suffered on average 2.3424 more fouls compared to 2020. We also see that in 2012, teams suffered on average 2.7253 more fouls compared to 2024. These two differences in means are statistically significant; the others are not, so there is no point in commenting on them.

For the variable “Free Kicks Taken”, we have three differences in means that are statistically significant at a 5% significance level, which we will analyze below. We observe that in 2016, there were on average 3.3449 more free kicks compared to 2012. Furthermore, 2020 also had on average 2.0168 more free kicks compared to 2012. The last statistically significant difference is between 2024 and 2016, where in 2024 there were on average 2.0526 fewer free kicks.

Finally, we have the variable “Passes Attempted”. Here, we have two differences that are statistically significant at a 5% significance level, which will be mentioned next. In 2016, there were on average 144.7897 fewer pass attempts compared to the year 2012, and in 2020, there were on average 91.4422 more pass attempts compared to 2016.

4.4.4 Kruskal-Wallis for variables by tournament year

Concluding this chapter, we will use the Kruskal-Wallis test on the same variables that were used in the previous test for the group of teams. The only difference is that the test will now be performed on the mean ranks based on the years in which the tournaments took place. Thus, we have the following table:

Kruskal-Wallis Test		
Variables	H Value	p-value
Goals	4.5991	0.2036
Total Attempts	20.5994	<0.001
Attempts off Target	4.8152	0.1858
Attempts on Target	19.9101	<0.001
Assists	7.1279	0.0679
Offsides	3.5173	0.3189
Passing Accuracy (%)	37.7109	<0.001
Goals Conceded	5.3066	0.1507

Table 4.15: Results of the Non-Parametric Kruskal-Wallis Test for comparing distributions between the four tournament years for variables where the assumptions for ANOVA were violated.

Based on the above results at a 5% significance level, we observe that the statistically significant differences between the years in which the tournaments were held are in the variables Total Attempts, Attempts on Target, and Passing Accuracy. Therefore, for the variables where there are differences between the years, we will perform an additional test to determine exactly where these differences occur. This test will once again be the Dunn's test with the Holm correction.

Dunn's test with the Holm correction		
Categories	Z Value	p-value
Total Attempts		
2012-2016	3.4776	0.002
2012-2020	4.2282	<0.001
2016-2020	0.8392	1
2012-2024	3.7480	<0.001
2016-2024	0.3023	0.7624
2020-2024	-0.5369	1

Attempts on Target		
2012-2016	3.6383	0.0014
2012-2020	4.1593	<0.001
2016-2020	0.5825	1
2012-2024	3.4840	0.002
2016-2024	-0.1725	0.863
2020-2024	-0.7550	1
Passing Accuracy (%)		
2012-2016	-4.7552	<0.001
2012-2020	-4.5783	<0.001
2016-2020	0.1977	0.8432
2012-2024	-5.8973	<0.001
2016-2024	-1.2769	0.4033
2020-2024	-1.4746	0.421

Table 4.16: Results of the Post-hoc Dunn's test with Holm correction for variables that showed a statistically significant difference in the Kruskal-Wallis test by year.

Based on the above results, we see that for the variable "Total Attempts", the differences are primarily due to 2012, which had a significantly different number of attempts compared to all other years. The years 2016, 2020, and 2024 do not differ significantly from each other. Additionally, we conclude that the year 2012 is the one with the most attempts. The pattern is identical to the "Total Attempts" variable. The year 2012 stands out with a significantly different number of shots on target, while the remaining years (2016, 2020, and 2024) form a group with no significant internal differences. The same applies here. For passing accuracy, 2012 shows extremely significant differences ($p < 0.001$) with all subsequent years. The subsequent years (2016, 2020, 2024) do not differ significantly from each other. Furthermore, 2012 is the year with the worst passing accuracy.

CHAPTER 5

5. Generalized Linear Models

In this chapter, we will apply two generalized linear models. The first is a binary logistic regression model that predicts whether a team advances beyond the group stage or not. The second is a multinomial logistic regression model that classifies teams into the three detailed performance categories: Low Level (teams that were eliminated in the group stage), Medium Level (teams that advanced up to the round of 16 and the quarter-finals), and High Level (teams that reached the semi-finals and the final). To successfully address the two research questions, we must appropriately fit generalized linear models that will have as response variables, first, the categorical variable Progress and, subsequently, the three-level classification of teams based on their final position, Phase.

5.1 Introduction

Generalized Linear Models (GLMs) constitute a fundamental and powerful statistical methodology that extends the boundaries of classical linear models. In the classical linear model, the dependent (response) variable is assumed to follow a normal distribution, and its relationship with the independent variables is directly linear. However, many real-world datasets do not conform to these prerequisites. This is where GLMs come in to fill the gap, offering a unified framework for analysis.

The core idea of GLMs is the use of a link function, $g(*)$, which connects the mean, $\mu = E(Y)$, of the response variable Y to a linear combination of the explanatory variables $X_1, X_2, \dots, X_n$. This connection is expressed by the fundamental relationship:

$$g(\mu) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Where $\beta_0, \beta_1, \dots, \beta_n$ are the unknown model parameters that must be estimated. This flexibility allows the model to handle data that follow various distributions from the broad exponential family of distributions, such as the binomial (for binary or proportional data), the Poisson (for count data), the negative binomial, the gamma, and, of course, the normal distribution. Consequently, GLMs only require the assumption that the distribution of the response variable belongs to this family, without needing assumptions about the distribution of the errors or the constancy of their variance (homoscedasticity).

The advantages of using GLMs are significant. Beyond the ability to model non-normal data, the parameter estimators are derived via the maximum likelihood method, so that they possess desirable asymptotic properties (asymptotic unbiasedness, consistency, efficiency, asymptotic normality). Furthermore, the unified theoretical framework of GLMs eliminates the need for different approaches depending on the nature (quantitative or qualitative) of the explanatory variables.

In practice, the choice of the appropriate GLM depends on the nature of the response data. For example, for the analysis of count data, such as the number of events within a time period (e.g., number of goals in a football match), Poisson regression is the natural and often necessary approach. Unfortunately, we have seen that this is not the case in our specific situation. It is essential, prior to applying any GLM, to conduct a preliminary check for the presence of multicollinearity among the explanatory variables, in order to ensure valid and interpretable estimates of the model coefficients.

In summary, Generalized Linear Models offer a powerful and flexible way to describe and examine relationships between variables across a wide range of research conditions where traditional linear models fail to provide a reliable analysis

5.2 Logistic Regression

Logistic regression is the most fundamental and widely used Generalized Linear Model (GLM) for the analysis of binary responses, meaning where the dependent variable Y can only take two possible values (e.g., 0/1, Yes/No, Success/Failure). It is based on the assumption that the response follows a binomial distribution, and therefore, it models the probability of the event of interest ($Y = 1$) occurring.

The heart of the method is the link function, which transforms the probability $p = P(Y = 1|X)$, restricted to the interval [0,1], into a linear combination of the independent variables that can take any real value. The choice of the appropriate link function leads to different models, the most common of which are:

- **Logit Model**
  It uses the logit link function. It is the most popular model due to the straightforward interpretation of its coefficients. Its formula is:

$$logit(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

  The term $\frac{p}{1-p}$ is called the odds. Thus, the coefficient β_i shows how the logarithm of the odds changes for a one-unit increase in the variable X_i. By rearranging the formula, we can also express the probability itself:

$$p = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_n X_n}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_n X_n}}$$

- **Probit Model**
  It is based on the probit function, which is the inverse of the cumulative distribution function of the standard normal distribution. The model has the form:

$$probit(p_i) = \Phi^{-1}(p) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

where Φ^{-1} is the inverse function of the stamdard normal cdf. The results of the probit model are often very similar to those of the logit model, but the coefficient estimates are interpreted based on the slopes of the normal curve.

- **Complementary Log-Log Model**
  It uses the non-symmetric cloglog function, which is defined as:

$$cloglog(p) = \log(-\log(1-p)) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

  This model is particularly useful when the probability p approaches 1 (success) very quickly as the independent variables increase, but approaches 0 (failure) very slowly. It is suitable for data with pronounced asymmetry. *(Politis, 2024)*

5.3 Multinomial Logistic Regression

Multinomial Logistic Regression is the natural generalization of binary logistic regression for cases where the response variable Y has more than two categories ($J > 2$). It is one of the most fundamental and widely used models for analyzing multi-categorical data across many scientific fields.

Since the J probabilities $\pi_1, \pi_2, \dots, \pi_j$ sum to one, the modeling strategy is to model the relative probabilities of each category in relation to a chosen reference category (baseline category). Without loss of generality, we usually choose the first category ($j = 1$) as the reference. Below are some models of multinomial regression presented:

- **Multinomial Logit Model**
  It uses the generalized logit link function. It is the natural extension of binary logistic regression for response variables with more than two categories. The formula is for a reference category (usually $j = 1$), the model for each other category $j = 2, \dots, J$ is:

$$log\left(\frac{\pi_j}{\pi_1}\right) = \boldsymbol{X}^T\boldsymbol{\beta}_j = \beta_{j0} + \beta_{j1}\chi_1 + \cdots + \beta_{jp}\chi_{jp}, \;\; j = 2, \dots, J$$

  The term π_j/π_1 is called the relative odds of category j against the reference category. Thus, the coefficient β_{ji} shows how the logarithm of the relative odds changes for a one-unit increase in the variable X_i. The probabilities for each category can be expressed as:

$$\pi_1 = \frac{1}{1 + \sum_{i=2}^{J} e^{\boldsymbol{X}^T\boldsymbol{\beta}_i}}, \qquad \pi_j = \frac{e^{\boldsymbol{X}^T\boldsymbol{\beta}_j}}{1 + \sum_{i=2}^{J} e^{\boldsymbol{X}^T\boldsymbol{\beta}_i}}, \qquad j = 2, \dots, J$$

- **Multinomial Probit Model**
  It assumes $\varepsilon = (\varepsilon_1, \dots, \varepsilon_j)^T \sim N_j(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a covariance matrix. Consequently, the vector of utilities follows a multivariate normal distribution:

$$U = (U_1, \dots, U_j)^T \sim N_j(\boldsymbol{Bx}, \boldsymbol{\Sigma})$$

Where $\boldsymbol{B}$ is a $J \times (p+1)$ matrix with rows $\beta_1^T, \ldots, \beta_J^T$.

- **Heteroskedastic Logit Model**

  This model assumes the errors are independent but not identically distributed. Specifically:

$$\varepsilon_j \sim \mathcal{EV}(0, \sigma_j), \qquad \sigma_j > 0, \;\; j = 1, \ldots, J$$

  where $\mathcal{EV}(0, \sigma_j)$ denotes the extreme value (Gumbel) distribution with scale parameter σ_j. Its cumulative distribution function is:

$$F_j(x) = e^{-e^{-x/\sigma_j}}, \;\; x \in R$$

- **Logit Models for Ordinal Variables**

  In many problems, the $J \geq 2$ categories of the response variable have an inherent ordering (e.g., very low/low/moderate/high/very high). In such cases, we can use specialized models that exploit this structure and are more parsimonious (fewer parameters) and interpretable.

  1. **Proportional Odds Model**

     This is the most widely used model for ordinal responses. Instead of modeling the probabilities of individual categories, it models the cumulative probabilities (i.e., the cumulative distribution function).

$$logit\{P(Y \leq j)\} \equiv log\left(\frac{P(Y \leq j)}{P(Y > j)}\right) = \alpha_j + X^T\beta, \qquad j = 1, \ldots, J-1$$

     Where $P(Y \leq j) = \pi_1 + \cdots + \pi_j$ is the cumulative probability up to category j, α_j is the threshold (intercept) for the j-th logit transformation and β is the common vector of coefficients for all j.

  2. **Adjacent Categories Logits**

$$\log\left(\frac{\pi_{j+1}}{\pi_j}\right) = \alpha_j + X^T\beta, \qquad j = 1, \ldots, J-1$$

  3. **Continuation Ratio Logits**

$$\log\left(\frac{\pi_1 + \cdots + \pi_j}{\pi_{j+1}}\right) = \alpha_j + X^T\beta$$

Or

$$\log\left(\frac{\pi_j}{\pi_{j+1} + \cdots + \pi_j}\right) = \alpha_j + X^T\beta$$

(Iliopoulos, 2024)

5.4 Goodness-of-Fit Measures and Statistical Significance Tests for Variables and Overall Model Adequacy

After becoming familiar with the fundamental models of logistic and multinomial logistic regression, it is crucial to know how to evaluate the quality of their fit and to test the statistical significance of the variables they include. In contrast to linear regression, where the coefficient of determination R^2 provides a direct indication of the model's adequacy, in multinomial logistic regression we employ specialized measures and statistical tests that are appropriate for categorical response variables.

5.4.1 Akaike's Information Criterion (AIC)

To identify the best model, one of the most widely used criteria is Akaike's Information Criterion, commonly known as AIC. For a model with k parameters, the AIC is defined as:

$$AIC = 2k - 2\ln(L)$$

where L denotes the maximum likelihood value of the model. In order to select the optimal model among the candidates, we look for the smallest AIC value. The AIC balances the model fit (through the log-likelihood) and its complexity (via the number of parameters k), thereby penalizing over-parameterization. It is an extremely useful tool for comparing models, especially when the models are not nested. However, AIC should not be used in isolation. Other important aspects that must be considered include the statistical significance of the variables used in our models as well as their overall adequacy. *(Politis, 2024)*

5.4.2 Statistical Significance of Variables: Wald Test and Likelihood Ratio Test (LRT)

After the appropriate selection of the model, it is necessary to test the significance of the variables that are useful and to explain their variability. Two of the most fundamental methods for evaluating the statistical significance of variables in logistic regression models are the Wald test and the Likelihood Ratio Test (LRT) . The Wald test is a statistical hypothesis test where the two alternatives are:

- $H_0: \beta = 0$
- $H_1: \beta \neq 0$

with the test statistic:

$$Z = \frac{\hat{\beta}}{s.e.(\hat{\beta})}$$

where $\hat{\beta}$ is the estimator of the parameter for each independent variable. When the null hypothesis $H_0: \beta = 0$ holds, the test statistic Z asymptotically follows the standard normal distribution $N(0,1)$. From estimation theory, the maximum likelihood estimators for the parameters of a generalized linear model asymptotically follow the

normal distribution. Consequently, for a large number of observations (typically over 30), we can use the normal distribution to examine whether a parameter β differs significantly from zero. *(Politis, 2024)*

An equally important and widely used method for testing the overall adequacy of a model, as well as for comparing nested models, is the Likelihood Ratio Test (LRT). The LRT is based on comparing the likelihood functions of two nested models:

- $\boldsymbol{H_0}$: the two models do not differ significantly
- $\boldsymbol{H_1}$: the larger model is significantly better than the other one

The test statistic is derived from the ratio of the maximum likelihood values of the two models:

$$G^2 = -2log\left(\frac{L(Model\ H_0)}{L(Model\ H_1)}\right) = 2[logL(H_1) - logL(H_0)]$$

where $L(H_0)$ and $L(H_1)$ are the maximum likelihood values of the simpler and fuller models, respectively. The statistic G^2 can equivalently be expressed through the deviance of the models, which is defined as:

$$D = -2logL$$

Therefore, the LRT statistic can be written as:

$$G^2 = D(H_0) - D(H_1)$$

where $D(H_0)$ and $D(H_1)$ are the deviances of the simpler and fuller model, respectively. Under the null hypothesis, the statistic G^2 follows a x^2distribution with degrees of freedom equal to the difference in the number of parameters between the two models:

$$G^2 \sim x^2_{df}, \quad df = p_1 - p_0$$

where p_1 and p_0 are the number of parameters in the fuller and simpler model, respectively. The p-value is calculated as:

$$p = P(x^2_{df} \geq G^2)$$

- If $p < \alpha$: We reject the simpler model in favor of the more complicated one.
- If $p \geq \alpha$: The simpler model is considered adequate. *(Iliopoulos 2024)*

5.4.3 Hosmer-Lemeshow Goodness-of-Fit Test

After evaluating the statistical significance of the variables, the next step is to examine the overall goodness-of-fit of our model. For logistic regression models, one of the most widely used tests for this purpose is the Hosmer-Lemeshow test.

This test examines whether our model fits the data satisfactorily, with the following hypotheses:

- $\boldsymbol{H_0}$: The observed values of the response variable Y do not differ significantly from the estimated values predicted by the model (the model has a good fit).
- $\boldsymbol{H_1}$: The observed values differ significantly from the estimated values (the model does not have a good fit).

The procedure for applying the test involves the following steps:

1) Ordering the observations based on the predicted probability of success calculated by the model.
2) Dividing the ordered observations into g groups (usually $g = 10$), where each group contains approximately an equal number of observations.
3) Recording, for each group, the observed number of successes and failures, thus forming a $g \times 2$ contingency table.

The Hosmer-Lemeshow test statistic, denoted as $\widehat{H}$, is calculated as the Pearson χ^2 statistic for the above table. Under the null hypothesis, $\widehat{H}$ follows (approximately) a χ^2 distribution with $g - 2$ degrees of freedom.

Rejection of the null hypothesis (i.e., p-value < α) indicates that there are statistically significant differences between the observed and expected values, which means that our model does not have a good fit to the data. Conversely, if we do not reject H_0, we can conclude that the model fits satisfactorily. *(Politis, 2024)*

5.4.4 McFadden's Pseudo R^2

For the qualitative assessment of the predictive ability and adequacy of a multinomial logistic regression model, one of the most commonly used measures is McFadden's Pseudo R^2. Unlike the traditional R^2 of linear regression, which does not directly apply to models with categorical responses, pseudo-R^2 measures provide a way to quantify how much better our model predicts the data compared to a null model. McFadden's Pseudo R^2 is defined as:

$$R^2_{McF} = 1 - \frac{\ln L_{model}}{\ln L_{null}}$$

Where $\ln L_{\text{model}}$ is the logarithm of the maximum likelihood of the model that includes the explanatory variables and $\ln L_{\text{null}}$ is the logarithm of the maximum likelihood of the null model. When McFadden's Pseudo R^2 equals zero, this indicates that our model offers no improvement compared to the null model. As McFadden's Pseudo R^2approaches one, this suggests that our model provides an almost perfect fit to the data. In applied research, values above 0.2 are generally considered satisfactory, demonstrating that the model explains a significant portion of the variability in the response. Values above 0.4 typically indicate very good model fit, suggesting that the explanatory variables substantially improve prediction accuracy relative to the null model. *(Smith & McKenna, 2013)*

5.4.5 Multicollinearity Testing

This specific test is essential before beginning any modeling procedure, because it is quite common in the data under examination to observe the phenomenon where the independent variables used are highly correlated with each other. This fact renders a model particularly unreliable, as potential high correlations can significantly alter its interpretation, leading to erroneous conclusions. If some independent variables in a multiple regression model have a very high correlation with each other, then we say that multicollinearity is present. *(Koutras, 2023).*

This check gains particular importance in the context of logistic regression as well, since this model belongs to the broader category of Generalized Linear Models (GLMs). The presence of multicollinearity affects the reliability and stability of the estimates, even when the dependent variable is binary or multi-class. Specifically, it can lead to inflation of standard errors, unstable and unpredictable coefficient estimates, difficulty in the separate interpretation of each variable's effect. The criterion for determining the presence or absence of multicollinearity in a model is the value of the VIF (Variance Inflation Factor) index. Considering a model with n independent variables, this factor for each variable is defined by the relation:

$$VIF_k = \frac{1}{1 - R_k^2}, \quad k = 1,2, \dots, n$$

where R_k^2 is the coefficient of determination (R-squared) of the model that uses the k-th variable as the dependent variable and the remaining variables as independent.

The VIF (Variance Inflation Factor) index has a lower bound of 1; the closer it is to this value, the more evidence we have in favour of independence among the predictors, while it has no upper bound. A VIF value equal to 1 indicates that the variables are not correlated. When the index values fall within the range of 1 to 5 ($1 \leq VIF \leq 5$), the variables are characterized as moderately correlated. Finally, when the index exceeds the value of 5 ($VIF > 5$), the variables are considered highly correlated, which is an indication of significant multicollinearity. *(Daoud, 2017)*

5.5 Application and Results of Logistic Regression for Predicting Advancement

We will now proceed to apply the above theory to our data for binary logistic regression. We will present our analysis in a somewhat reversed order, starting by first presenting the final model we arrived at. Then, we will conduct the evaluation of our model, and finally, we will present the conclusions we reached. In this section, we will use as the response variable the binary variable Progress. Teams that advanced beyond the group stage (round of 16, quarter-finals, semi-finals, final, and winner) belong to the "Advanced" category, while teams that were eliminated in the group stage belong to the "GroupStage" category. In this case, we will use the per-match data for the teams. We will begin by first presenting the model we arrived at, and then we will examine its reliability and the conclusions we will draw.

5.5.1 Binary Logistic Model for Team Progress Classification

In our search for the optimal classification model for team performance, a binary logistic model was estimated. The model, comprising key per-game performance metrics, aims to predict the probability of a team advancing beyond the group stage (Progress variable) based on the following explanatory variables: Attempts on Target per match, Passes Completed per match, Blocked per match, Attacks per match, Goals Conceded per match and Saves from Penalties per match. The model was implemented in the R statistical environment using the glm() function with binomial family. To achieve algorithm convergence, a maximum limit of 50 iterations (maxit = 50) was set, which was sufficient for the model to converge smoothly.

It is important to note that variables directly related to goals scored (such as Goals per match, Assists per match, Right foot per match, Left foot per match, Head per match, Goals inside area per match, Goals outside area per match, and Penalties scored per match) were intentionally excluded from the analysis. The reason for this exclusion is that the outcome of matches is ultimately determined by the number of goals each team scores. Including goal-related variables would lead to a trivial and self-evident conclusion: teams that score more goals advance. Such a finding would offer no meaningful insight into the underlying factors that contribute to a team's success. Instead, by omitting these variables, we aim to identify deeper, more fundamental performance indicators that precede goal-scoring, such as accuracy in the final attempt (Attempts on Target per match), ball circulation and possession (Passes Completed per match), the ability to create chances despite the opposing defense (Blocked shots per match), overall offensive activity (Attacks per match), defensive resilience (Goals Conceded per match), and effectiveness in critical situations (Saves from Penalties per match). All variables were manually checked for statistical significance in the model we want to build, and we ended up with these.

Logistic Regression				
Variables	Estimate	Std. Error	Z-value	P-value
Intercept	1.772	2.087	0.849	0.396
Attempts on Target	1.146	0.348	3.286	0.001
Passes Completed	0.001	0.005	0.283	0.777
Blocked	-1.534	0.428	-3.588	0.000
Attacks	0.103	0.049	2.124	0.034
Goals Conceded	-3.588	1.024	-3.504	0.000
Saves Penalties	13.109	4.958	2.644	0.008
Null deviance: 115.365 on 87 degrees of freedom				
Residual deviance: 58.246 on 81 degrees of freedom				
AIC: 72.195				

Table 5.1: Binary logistic regression results for predicting the probability of teams advancing beyond the group stage (Progress variable)

First, we note that according to the Wald test, all variables are statistically significant at the 5% significance level, except for the constant term and Passes Completed per match (p = 0.777). The resulting model is a binary logistic regression model that uses teams eliminated in the group stage (GroupStage) as the reference category (baseline category). The estimation results include the coefficients, standard errors, z-value, p-value, as well as the deviance and the AIC criterion. The model estimates the

probability of a team belonging to the Advanced category (advancement beyond the group stage) relative to the reference category GroupStage. We remind that only per-match variables were used. The equation of the model is formulated as follows:

$$log\left(\frac{P(Advanced)}{P(GroupStage)}\right) = 1.772 + 1.146(A) + 0.001(P) - 1.534(B) + 0.103(AT) - 3.588(G) + 13.109(S)$$

Note: A =Attempts on Target, P = Passes Completed, B = Blocked, AT = Attacks, G = Goals Conceded, S = Saves Penalties.

In the binary logistic regression model, the coefficients represent the change in the logarithm of the relative probability of success (advancement versus elimination).

- If we increase the value of the variable Attempts on Target per match by one unit, while keeping the values of the other variables constant, the logarithm of the odds of success (advancement versus elimination) is expected to increase by 1.146 units.
- If we increase the value of the variable Blocked per match by one unit, while keeping the values of the other variables constant, the logarithm of the odds of success (advancement versus elimination) is expected to decrease by 1.534 units.
- If we increase the value of the variable Attacks per match by one unit, while keeping the values of the other variables constant, the logarithm of the odds of success (advancement versus elimination) is expected to increase by 0.103 units.
- If we increase the value of the variable Goals Conceded per match by one unit, while keeping the values of the other variables constant, the logarithm of the odds of success (advancement versus elimination) is expected to decrease by 3.588 units.
- If we increase the value of the variable Saves from Penalties per match by one unit, while keeping the values of the other variables constant, the logarithm of the odds of success (advancement versus elimination) is expected to increase by 13.109 units.

5.5.2 Statistical Evaluation and Diagnostic Validation of the Model

After presenting the estimated logistic model, a necessary step is to evaluate its statistical reliability and diagnostic validity. This subsection aims to test the model's core assumptions and measure its absolute predictive accuracy.

5.5.2.1 Diagnostic Test for Multicollinearity

To diagnose potential multicollinearity, the Variance Inflation Factor (VIF) will be calculated for each predictor. This is essential to verify the absence of strong interdependencies that could compromise the stability and interpretability of the model's coefficients.

Variance Inflation Factor (VIF) Analysis	
Variables	VIF
Attempts on Target	3.299
Passes Completed	1.909
Blocked	4.942
Attacks	2.378
Goals Conceded	2.273
Saves Penalties	2.117

Table 5.2: Variance Inflation Factors (VIF) for the model's predictor variables

This VIF diagnostic procedure confirms the absence of problematic multicollinearity in the model. The general rule for interpreting VIF stipulates that values exceeding 5 or 10 indicate harmful multicollinearity that can destabilize coefficient estimates. In our case, all VIF values are below the threshold of 5, with the highest value observed for the variable Blocked per match (VIF = 4.942), which does not raise concern. The remaining variables exhibit even lower values, with the variable Passes Completed per match having the smallest value (VIF = 1.909), indicating practical independence from the other independent variables.

Therefore, we can conclude that multicollinearity is not a source of bias in our model. The stability and reliability of the coefficient estimates presented in the logistic regression table are assured. This validation allows us to confidently proceed with the subsequent stages of model evaluation, focusing on statistical significance and predictive performance.

5.5.2.2 Likelihood Ratio Test (LRT)

Following the successful multicollinearity check, the next critical step in model assessment is to evaluate its statistical significance and to determine whether the inclusion of the selected predictors offers a substantial improvement over a simpler baseline. This is achieved through the Likelihood Ratio Test (LRT).

Analysis of Deviance Table					
Variables	Df	Deviance	Resid. Df	Resid. Dev	P-value
NULL			87	115.365	
Attempts on Target	1	6.575	86	108.790	0.010
Passes Completed	1	3.901	85	104.889	0.048
Blocked	1	10.148	84	94.742	0.001
Attacks	1	6.553	83	88.188	0.010
Goals Conceded	1	21.182	82	67.006	0.000
Saves Penalties	1	8.810	81	58.195	0.003

Table 5.3: Likelihood Ratio Test (LRT) results for the statistical significance of the model variables

The analysis of each variable's individual significance via Likelihood Ratio Tests reveals a clear hierarchy in the predictive power of the six variables in the model. The variable Goals Conceded per match exhibits the greatest statistical significance, with the largest reduction in deviance (21.182 units) when included in the model. This finding confirms that defensive resilience and a team's ability to avoid conceding goals constitute a decisive factor in distinguishing between teams that advance and those that are eliminated. Equally important is the variable Blocked per match, which also shows

very high statistical significance (p = 0.001) with a deviance reduction of 10.148 units. The negative sign of the coefficient indicates that the more offensive attempts of a team are blocked by opponents, the more the probability of advancement decreases. This is expected, as blocked shots represent offensive actions that do not reach the goal. The variable Saves from Penalties per match exhibits a statistically significant contribution (p = 0.003) with a deviance reduction of 8.810 units. Although this phenomenon is relatively rare, the ability to save penalties appears to constitute a qualitative indicator that holds independent predictive value, particularly in critical moments of matches. The variables Attempts on Target per match and Attacks per match contribute to a similar degree, with deviance reductions of 6.575 and 6.553 units respectively (p = 0.010 for both). Accuracy in the final attempt and overall offensive activity significantly enhance the predictive power of the model, though to a lesser extent than defensive stability and blocked shots. Finally, the variable Passes Completed per match shows a marginally statistically significant contribution (p = 0.048) with a deviance reduction of 3.901 units. Although its effect is the smallest among all variables, ball circulation and completed passes add an additional layer of information, suggesting that possession and passing can contribute, albeit in a limited way, to distinguishing success.

5.5.2.3 Assessing Goodness-of-Fit Using the Hosmer-Lemeshow Test

To evaluate the overall goodness-of-fit of the binary logistic model, we use the Hosmer-Lemeshow Goodness-of-Fit Test. This test is one of the most widely used diagnostic tools for assessing the fit of logistic regression models, as it examines whether the observed values of the response variable differ significantly from the expected values predicted by the model.

Hosmer-Lemeshow Test	
Fit Measure	Value
X-squared (H-L)	4.405
Df	6
P-value	0.622

Table 5.4: Hosmer-Lemeshow goodness-of-fit test results for the binary logistic model

In our model, the test statistic value is X-squared = 4.405 with 6 degrees of freedom and p-value = 0.622. Given that the p-value is considerably larger than the 0.05 significance level, we fail to reject the null hypothesis. This means that there are no statistically significant differences between the observed and expected values, indicating that our model exhibits a good fit to the data.

The high p-value (0.622) confirms that our model is adequate and its predictions are consistent with the observations. This result, combined with the previous diagnostic tests (multicollinearity test and LRT), further reinforces the validity and reliability of our model for predicting the probability of team advancement.

5.5.2.4 Analysis of Predictive Accuracy using the Confusion Matrix

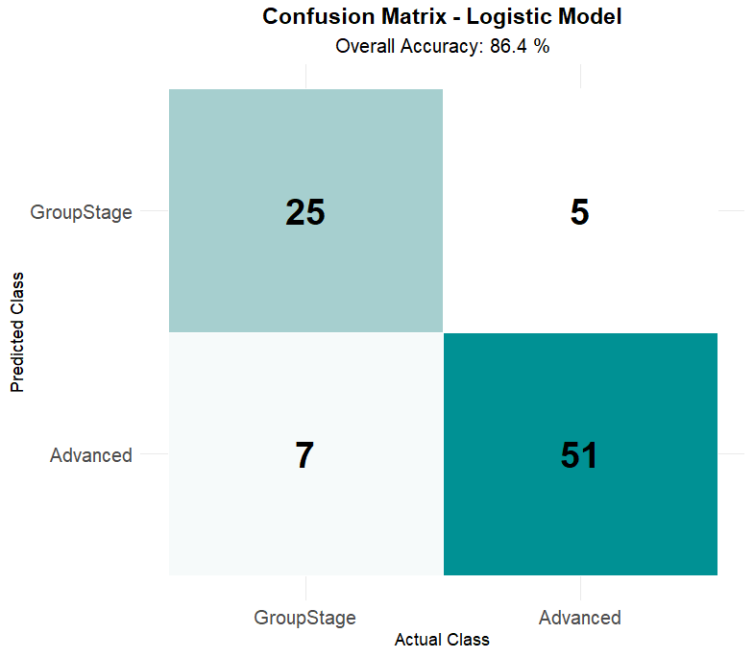

Figure 5.1: Confusion matrix heatmap for the logistic regression model

The final and decisive phase of the model's evaluation is testing its actual predictive ability in-sample. For this purpose, the Confusion Matrix is constructed, which compares the categories predicted by the model with the actual observed categories for all teams in the sample.

The model correctly classifies 86.4% of the cases (76 out of 88 teams). This percentage is very satisfactory for a binary logistic model, indicating strong practical utility in predicting whether a team advances beyond the group stage. The model exhibits high specificity for the GroupStage category, with 78.1% (25 out of 32) of truly eliminated teams correctly classified. The error pattern for this category shows that 7 teams that were actually eliminated were misclassified as Advanced, indicating a tendency to overestimate the probability of advancement for some weaker teams. For the Advanced category, the model demonstrates even higher performance, with 91.1% (51 out of 56) of truly advancing teams correctly identified. This high recall means the model is particularly reliable in recognizing teams that successfully progress beyond the group stage. The misclassification pattern shows that 5 advancing teams were incorrectly predicted as GroupStage, representing an underestimation error. The overall accuracy of 86.4% confirms that the selected per-match performance metrics (Attempts on Target, Passes Completed, Blocked shots, Attacks, Goals Conceded and Saves from Penalties) collectively provide strong predictive power for team advancement, while avoiding the trivial conclusion that would come from including goal-related variables.

5.6 Application and Results of the Multinomial Logistic Regression Model

Continuing with our analysis, we proceed to apply the multinomial logistic model, which is the natural extension of binary logistic regression for cases where the response

variable takes more than two categories. In the present study, the response variable is Phase, which classifies teams into three levels based on their final position in the tournament: Low Level for teams eliminated in the group stage, Medium Level for teams that advanced up to the round of 16 and the quarter-finals, and High Level for teams that reached the semi-finals and the final. The aim of the model is to investigate how almost the same key per-match performance metrics used in the binary model (Attempts on Target per match, Blocked per match, Attacks per match, Goals Conceded per match, and Saves from Penalties per match) affect the probability of a team being classified into each of the three performance levels. As in the case of binary logistic regression, variables directly related to goals were intentionally excluded from the analysis in order to avoid trivial conclusions and to focus on deeper, more fundamental performance indicators that precede goal-scoring. The model was estimated in the R statistical environment using the multinom() function from the "nnet" package, with Low Level as the reference category (baseline). To achieve algorithm convergence, a maximum limit of 1000 iterations (maxit = 1000) was set.

5.6.1 Statistical Evaluation of the Model

First of all, we will check for multicollinearity among the variables. According to our table, we do not have a multicollinearity issue with the variables, because none exceeds the value of 5.

Variance Inflation Factor (VIF) Analysis	
Variables	VIF
Attempts on Target	3.228
Blocked	4.853
Attacks	1.559
Goals Conceded	2.182
Saves Penalties	1.864

Table 5.5: Variance Inflation Factors (VIF) for the model's predictor variables

We will also immediately test whether the variables are significant using the Likelihood Ratio Test (LRT) method.

Analysis of Deviance Table			
Variables	LR Chisq	Df	P-value
Attempts on Target	28.212	2	0.000
Blocked	27.061	2	0.000
Attacks	11.238	2	0.003
Goals Conceded	29.473	2	0.000
Saves Penalties	10.613	2	0.005

Table 5.6: Likelihood Ratio Test (LRT) results for the multinomial logistic model

The analysis of the statistical significance of each variable through the Likelihood Ratio Test reveals that all variables are statistically significant in the multinomial model, confirming their overall contribution to the classification of teams into the three performance levels (Low, Medium, High).

To evaluate the overall explanatory power of the multinomial logistic model, we use McFadden's Pseudo R^2. This is one of the most widely accepted and conservative measures for quantifying the goodness-of-fit in models with a categorical dependent variable.

Fit Measure	Value
McFadden's Pseudo R^2	0.448

Table 5.7: McFadden's Pseudo R-squared value for the multinomial logistic model

Empirical benchmarks for McFadden's Pseudo R^2 suggest that values between 0.2 and 0.4 represent a model with substantial explanatory power, while values exceeding 0.4 indicate an exceptionally strong fit (Louviere et al., 2000; McFadden, 1974).The calculated value of 0.448 for our model decisively surpasses this higher benchmark. Consequently, we conclude that the multinomial logistic model, specified with key offensive and defensive metrics, demonstrates superior explanatory strength in accounting for the variation in a team's tournament progression (Phase). This result provides strong quantitative corroboration for the model's validity, confirming that the chosen predictors capture a major share of the underlying determinants of competitive success.

5.6.2 Multinomial Logistic Model for Team Phase Classification

As we mentioned, we will use the same variables as in the logistic model. The estimates obtained are presented in the following table.

Multinomial Logistic Model				
Parameter	Medium		High	
	Coefficients	Std. Error	Coefficients	Std. Error
(Intercept)	1.814	2.099	-1.008	2.880
Attempts on Target	1.073	0.350	1.980	0.482
Blocked	-1.511	0.429	-1.861	0.521
Attacks	0.109	0.040	0.140	0.050
Goals Conceded	-3.347	1.006	-6.299	1.573
Saves Penalties	13.087	4.693	15.818	6.196
Residual Deviance: 100.629				
AIC: 124.629				

Table 5.8: Multinomial logistic regression results for performance category prediction

The resulting model is a Multinomial Logit Model that uses the "Low" category as the baseline reference category. The estimation results provide for each of the two remaining categories ("Medium" and "High"): Coefficients, Std. Errors, Residual Deviance, AIC. The model estimates:

$$log\left(\frac{P(Medium)}{P(Low)}\right) = 1.814 + 1.073(A) - 1.511(B) + 0.109(AT) - 3.347(G) + 13.087(S)$$

$$log\left(\frac{P(High)}{P(Low)}\right) = -1.008 + 1.980(A) - 1.861(B) + 0.140(AT) - 6.299(G) + 15.818(S)$$

Note: A = Attempts on Target, B = Blocked, AT = Attacks, G = Goals Conceded, S = Saves Penalties.

In the multinomial logistic model, each set of coefficients represents the log-odds of a team being in a specific category (Medium or High) relative to the baseline category (Low). Therefore, the interpretation below corresponds to the outcome of being in the stated category versus the baseline.

For the Medium category (relative to Low):

- If we increase the value of the Attempts on Target per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds of a team belonging to the Medium category instead of the Low category is expected to increase by 1.073 units.
- If we increase the value of the Goals Conceded per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (Medium vs Low) is expected to decrease by 3.347 units.
- If we increase the value of the Blocked per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (Medium vs Low) is expected to decrease by 1.511 units. It is reminded that blocked shots represent offensive attempts by the team that are blocked by the opposing defense.
- If we increase the value of the Attacks per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (Medium vs Low) is expected to increase by 0.109 units.
- If we increase the value of the Saves from Penalties per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (Medium vs Low) is expected to increase by 13.087 units.

For the High category (relative to Low):

- If we increase the value of the Attempts on Target per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds of a team belonging to the High category instead of the Low category is expected to increase by 1.980 units.
- If we increase the value of the Goals Conceded per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (High vs Low) is expected to decrease by 6.299 units.
- If we increase the value of the Blocked per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (High vs Low) is expected to decrease by 1.861 units.
- If we increase the value of the Attacks per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (High vs Low) is expected to increase by 0.140 units.
- If we increase the value of the Saves from Penalties per match variable by one unit, while keeping the values of the other variables constant, the logarithm of the odds (High vs Low) is expected to increase by 15.818 units.

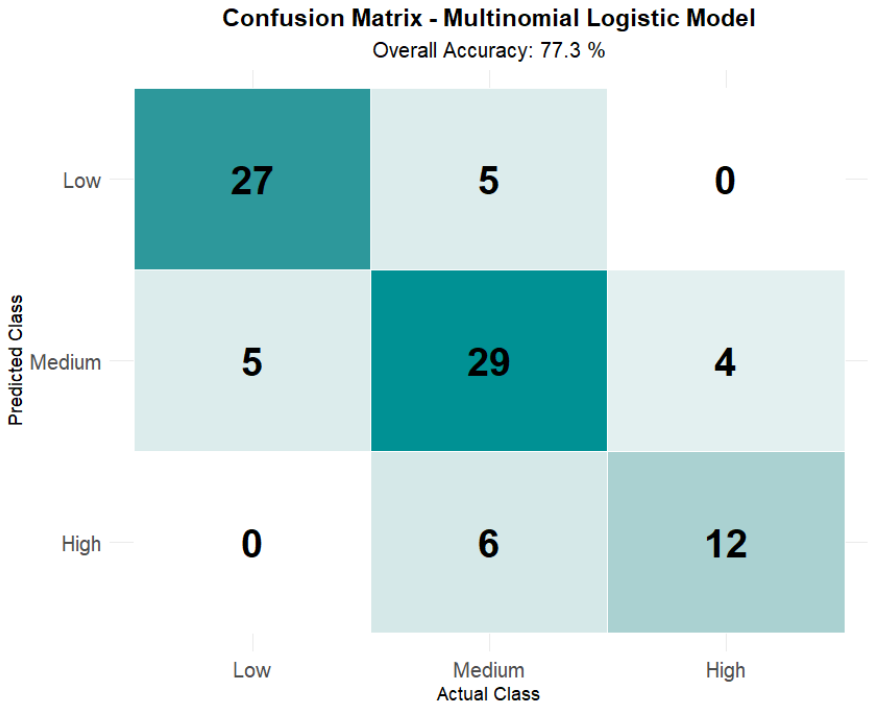

Figure 5.2: Confusion matrix for the multinomial logistic regression model

The model correctly classifies 77.3% of the cases (68 out of 88 teams). This percentage is satisfactory for a multinomial model with three categories, indicating practical utility in classifying teams by performance level. For the Low category, the model exhibits high sensitivity, with 84.4% (27 out of 32) of teams that were actually eliminated in the group stage correctly classified. The error pattern shows an upward bias towards the higher neighboring category (Medium), with 5 teams misclassified as Medium and none as High. The Medium category shows the highest absolute frequency of correct classifications, with 29 teams and a sensitivity of 72.5% (29 out of 40). This means the model is fairly reliable in identifying medium-level teams. The errors are distributed asymmetrically: 5 teams are misclassified as Low and 6 as High. For the High category, sensitivity is 75.0% (12 out of 16). This is a strong performance given the smaller sample size of this category (only 16 teams). The main error pattern is underestimation towards Medium: out of the 4 incorrect predictions, 4 teams were classified as Medium and none as Low.

CHAPTER 6

6. Machine Learning

In this chapter, we will investigate, through machine learning techniques, the ability of various classification algorithms to predict the progress of national teams in the UEFA European Championship (EURO) for the period 2012-2024. The main objective of the analysis is to examine whether statistical performance data of teams can serve as reliable predictive factors for their advancement beyond the group stage. For this purpose, a comprehensive data analysis methodology was followed. Initially, data preprocessing was performed where the "Final Rank" variable was transformed into a binary target variable named "Progress", which divides the teams into two categories: those eliminated in the group stage (Group Stage) and those advancing to the next phase (Advanced). For our analysis, models such as KNN, SVM, and Neural Networks are employed.

6.1 Theoretical Background

Machine Learning constitutes a distinct subfield of Artificial Intelligence (AI) concerned with the development and application of algorithms and models that enable systems to "learn" from data and improve their performance without the need for explicit programming instructions. It incorporates techniques from mathematics, statistics, and computer science, and is widely used for analyzing and processing big data, pattern recognition, and decision-making in environments characterized by uncertainty.

At the foundation of machine learning lies the premise that systematic representations of data can be utilized to construct computational models that replicate or even surpass human intelligence in specific tasks. This learning capability is typically implemented through various algorithmic approaches such as supervised learning models, unsupervised learning models, and reinforcement learning models.

In supervised learning, a model is trained on a training dataset containing inputs and their corresponding correct outputs, enabling it to predict outputs for new inputs. Unsupervised learning involves identifying hidden structures in data without the presence of labels or predefined outputs. Finally, reinforcement learning is based on learning through interaction with an environment, where the system receives rewards or penalties for its actions, allowing it to improve its strategy over time.

6.1.1 Supervised learning

Supervised learning constitutes one of the fundamental and most widespread branches of machine learning, with extensive applications in scientific and engineering fields. Its primary objective is the derivation of a function that maps given inputs to known

outputs, based on a training dataset. At the core of supervised learning lies the concept of learning from examples. We have a training dataset consisting of N pairs of the form (x_i, y_i), for $i = 1, \dots, N$. Here, x_i is the input or feature vector, which can be quantitative or qualitative, and y_i is the output or label corresponding to x_i. The goal is to "learn" a function $f(x)$ that accurately predicts the output y for new, unknown inputs x, generalizing beyond the training data. The learning process of this function involves adjusting the model's parameters to minimize a specific error or cost. Statistical learning theory introduces the concept of a loss function $L(Y, f(X))$, which quantifies the discrepancy between the predicted value $f(X)$ and the true value Y. The training of the model aims to minimize this expected error, known as the expected prediction error.

The nature of the output variable Y determines the type of supervised learning problem, leading to two main categories:

Classification: In classification problems, the output Y is qualitative or categorical and takes values from a finite set of classes $\mathcal{G}$. Examples include digit recognition ($\mathcal{G} = \{0,1,\dots,9\}$) or classifying a team as "qualified" or "eliminated". The objective is to develop a decision rule that correctly assigns new observations to these classes. Popular classification algorithms include Support Vector Machines, Neural Networks, k-Nearest Neighbors and Decision Trees.

Regression: When the output variable Y is quantitative (e.g. temperature, stock price), the problem is defined as regression. The aim is to estimate the function $f(x)$ that describes the dependence of the output on the inputs. Characteristic regression algorithms include Linear Regression, Logistic Regression and Random Forests.

The development of a supervised learning model follows a standard methodology, summarized in the following steps:

1. **Data Collection and Preprocessing**: The dataset is assembled, followed by cleaning, handling of missing values, and often transformation or normalization of features (feature scaling) for the effective application of algorithms.
2. **Algorithm Selection**: One or more algorithms are selected, considering the nature of the problem (classification or regression), the size, and the properties of the data.
3. **Model Evaluation**: The generalization ability of the model is estimated using an independent test set or techniques such as cross-validation. Performance metrics are calculated, such as accuracy, sensitivity, and specificity for classification, or the mean squared error (MSE) for regression.
4. **Optimization (Tuning)**: Based on the evaluation, the algorithm's hyperparameters are adjusted to improve performance. This process is often repeated until a satisfactory result is achieved. *(Hastie, T., Tibshirani, R., & Friedman, J. (2009), Bishop, C. M. (2006), Murphy, K. P. (2012), Fernández-Delgado (2014))*

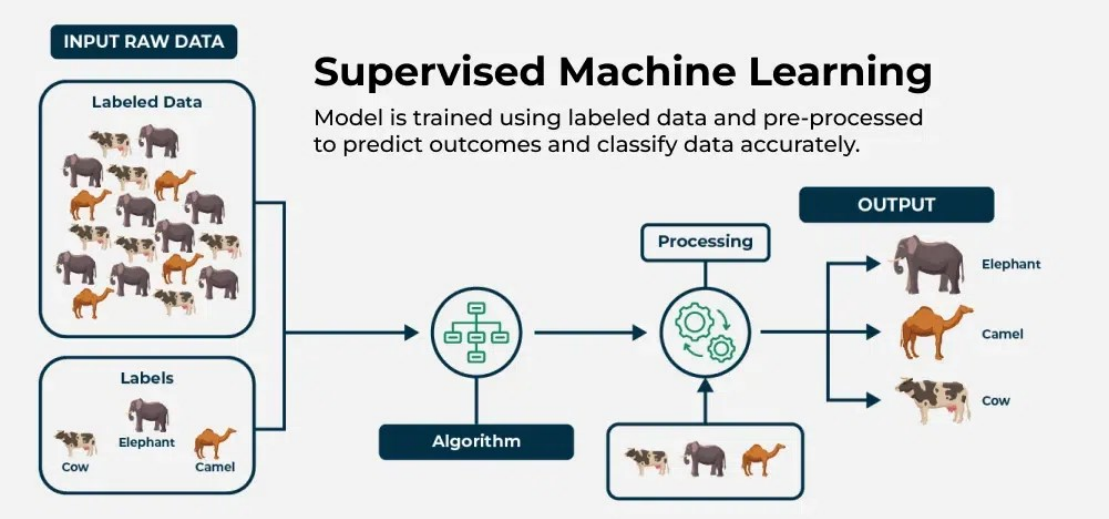

Figure 6.1: The Process of Classification. (Source: https://www.geeksforgeeks.org/machine-learning/supervised-machine-learning/)

6.1.2 Unsupervised learning

In contrast to supervised learning, where the goal is the prediction of a known output, unsupervised learning deals with cases where no labels or predefined outputs are available. In this context, we only have a set of input data x_i, without any corresponding target variable y_i. The aim is to discover hidden structures, patterns, or correlations within the data itself. Unsupervised learning is less objective than supervised learning, as there is no clear criterion for success, such as prediction accuracy. Instead, the objective is the discovery of interesting and interpretable representations of the data.

The main categories of problems and techniques in unsupervised learning include:

Clustering: This is the most fundamental technique, where the goal is the grouping of observations into "clusters", such that observations within the same group are more similar to each other compared to those in different groups.

- **K-Means Clustering**: One of the most popular clustering algorithms, which aims to partition the data into K predefined groups. Its mathematical foundation and its connection to the Expectation-Maximization (EM) algorithm for Gaussian mixture models are well-established.
- **Hierarchical Clustering**: This creates a tree-like structure (dendrogram) that groups the data at various levels of detail, without requiring the number of clusters to be predetermined.
- **DBSCAN**: A density-based clustering method that can identify clusters of arbitrary shape and recognize noise.

Dimensionality Reduction: These techniques are used to reduce the number of variables (features) under consideration, while preserving as much information from

the data as possible. They are particularly useful for visualizing high-dimensional data and addressing the "curse of dimensionality".

- **Principal Component Analysis (PCA)**: The most classic method, which finds new, uncorrelated variables (principal components) that linearly combine the original ones and capture the maximum possible variance of the data.
- **Non-negative Matrix Factorization**: A technique particularly popular in image and text data analysis, where the original values are non-negative.

The development of an unsupervised learning model follows a distinct methodology, summarized in the following steps:

1. **Data Collection**: An unlabeled dataset is gathered, consisting solely of input data x_i without any corresponding target variables or predefined categories.
2. **Algorithm Selection**: A suitable unsupervised algorithm is chosen based on the specific goal of the analysis. Common choices include clustering algorithms for grouping similar data points, dimensionality reduction techniques for reducing the feature space, or association rule learning algorithms for discovering relationships between variables.
3. **Model Training on Raw Data**: The entire unlabeled dataset is fed into the selected algorithm. The algorithm then autonomously explores the data, seeking inherent similarities, patterns, relationships, or hidden structures without any human guidance or external feedback.
4. **Data Grouping or Transformation**: Based on the patterns discovered during training, the algorithm organizes the data. This can take the form of grouping data points into distinct clusters, extracting a set of association rules, or transforming the data into a lower-dimensional representation.
5. **Interpretation and Utilization of Results**: The resulting groups, rules, or transformed features are analyzed to gain insights into the underlying structure of the data. These results can be used for various downstream tasks, such as data visualization, anomaly detection, or as a preprocessing step to create informative features for supervised learning models.

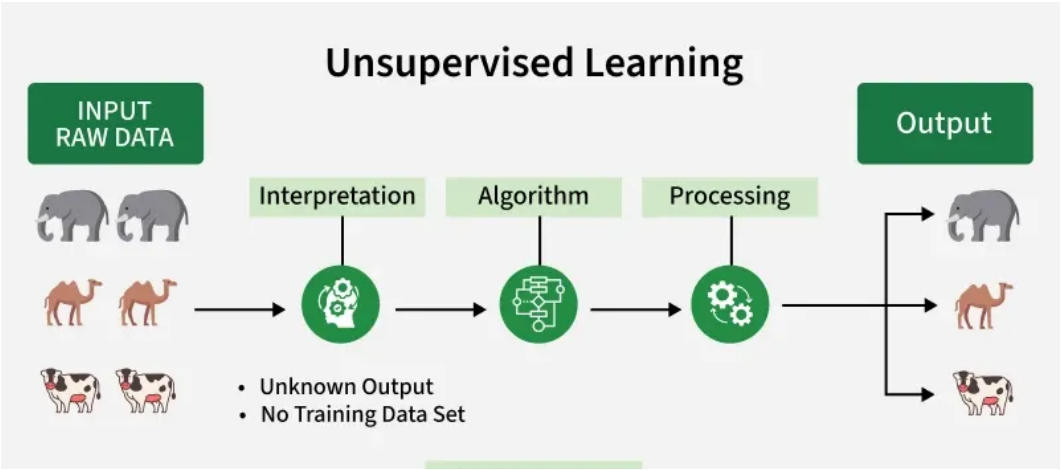

Figure 6.2: The Process of Clustering. (Source: https://www.geeksforgeeks.org/machine-learning/unsupervised-learning/)

Unsupervised learning constitutes a critical tool for exploratory data analysis, aiding in understanding the natural structure of the data, identifying outliers, and preprocessing for the application of supervised methods. *(Hastie, T., Tibshirani, R., & Friedman, J. (2009), Bishop, C. M. (2006), Murphy, K. P. (2012), Fernández-Delgado (2014))*

After having generally described the methods we will use and examined the logic behind them, we will now look in more detail at each step we will apply to bring our data into the most suitable form and to obtain the optimal result from each algorithm.

6.1.3 Feature Selection

Before applying machine learning algorithms to extract knowledge from our data, a crucial preparatory step is the selection of appropriate features (variables). This process, known as feature selection, aims to choose a subset of the most relevant features from the original dataset. This is done either to optimize a predictive model or to reduce the dimensionality of the data. The benefits are manifold: improving model accuracy, reducing the risk of overfitting, and speeding up the training process.

Feature selection techniques are divided into two main categories, depending on whether or not they utilize the information of the target variable.

- **Unsupervised Methods:** These methods are applied when a target variable is not available, i.e., on unlabeled data. The selection of features is based exclusively on the intrinsic characteristics of the data, such as their variance or their ability to preserve the data structure.
- **Supervised Methods:** In this case, the selection process is guided by the relationship of the features with the known target variable. The goal is to identify those independent variables that possess the greatest predictive power.

Supervised methods, in turn, are distinguished into three subcategories, depending on how they interact with the learning algorithm:

1. **Filter Methods:** These methods operate independently of any machine learning model. They evaluate and rank features based on statistical criteria, such as their correlation with the target variable. Typical examples include the Chi-Square test of independence and Fisher's Score. These are fast methods, suitable for datasets with many variables.
2. **Wrapper Methods:** In contrast to filter methods, here the evaluation of features is performed using a specific learning algorithm. The process is iterative: various subsets of features are tested, the model is trained with them, and its performance is evaluated. The search for the optimal subset is usually conducted using greedy strategies, such as the stepwise addition (Forward Selection) or the stepwise elimination of features (Backward Elimination). Although computationally more expensive, this category of methods takes into account the interactions between features.

3. **Embedded Methods:** These techniques attempt to combine the advantages of the two previous categories. Feature selection is integrated as part of the model's training process. During training, the algorithm learns which features contribute most to the accuracy of the predictions. Characteristic examples include regularization techniques such as Lasso (L1) and Ridge (L2) regression, which penalize or nullify the contribution of less important features. *(Rosidi, 2023)*

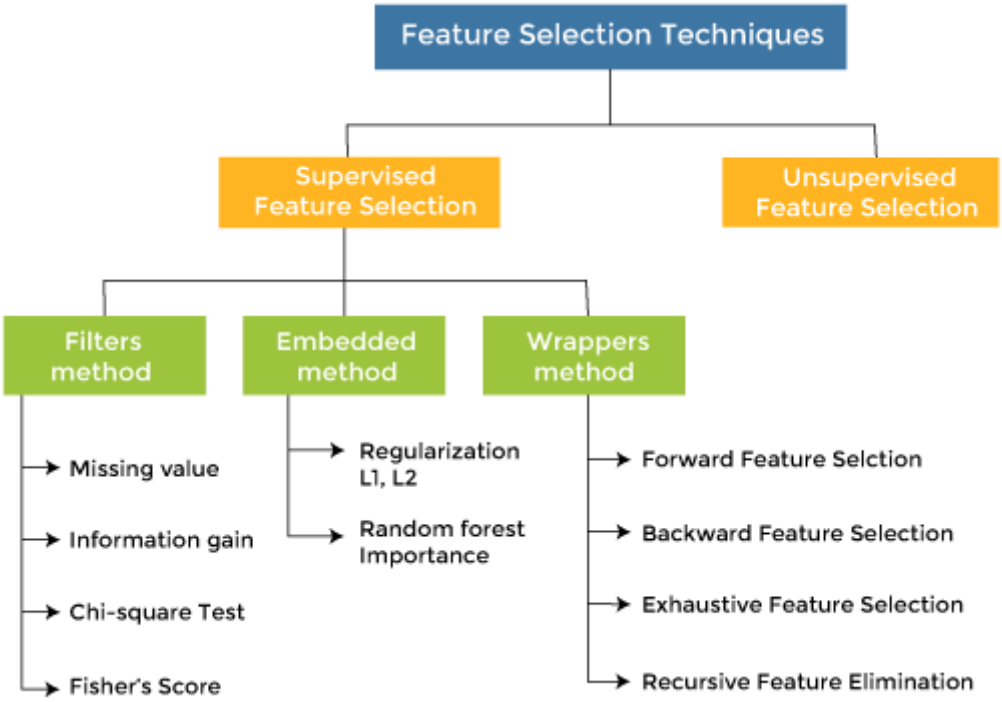

Figure 6.3: Categorization of Feature Selection Methods. (Source: https://www.linkedin.com/pulse/unleashing-power-feature-selection-machine-learning-ravi-singh/)

6.1.3.1 Boruta Method

The Boruta method is a feature selection algorithm that operates as a wrapper method and is based on Random Forests to estimate the importance of variables. Its primary goal is to identify all features that are truly relevant to a target variable. The algorithm begins by extending the original dataset, creating copies of the features known as "shadow features". The values of these shadow features are randomly shuffled to eliminate any correlation with the decision variable. A Random Forest model is then trained on this extended dataset. The importance of each feature (both real and shadow) is calculated based on the loss of classification accuracy caused by a random permutation of its values. This process yields an importance measure, the Z-score, which is the ratio of the average loss to its standard deviation. To determine which features are genuinely important, Boruta compares the importance (Z-score) of each real feature with the maximum importance observed among the shadow features. Real features with importance consistently higher than that of the shadow features are designated as "Confirmed". Conversely, those with lower importance are considered "Rejected". Finally, there is a category of "Tentative" features, about which the algorithm cannot decide with certainty. *(Szul, Tabor, Pancerz (2021))*

Thus, after selecting the most appropriate features, we will apply the methods that will be described right away.

6.1.4 Classification

Classification is a fundamental technique of supervised machine learning, where the goal is to predict a label or category for new data, based on training examples. Essentially, we aim to assign each observation to a predefined class, such as classifying emails as “spam” or “non-spam”.

The success of a classification model relies on learning patterns from a properly labeled dataset. During training, the algorithm analyzes the features of the data—such as the color, texture, or shape in an image—and learns their relationship with the corresponding class. The trained model can then predict the class for new, unseen data. Depending on the number and nature of the categories, we distinguish between three main types of classification problems:

1. **Binary Classification:** This is the simplest form, where data is separated into only two categories. A typical example is detecting whether a transaction is fraudulent or not.
2. **Multiclass Classification:** Here, the goal is to classify data into more than two categories, such as identifying an image as a cat, dog, or bird. Each data point belongs to only one class.
3. **Multi-Label Classification:** In this case, a single data point can belong to multiple categories simultaneously. For example, a movie could be tagged as both “action” and “comedy”.

To implement classification models, there is a variety of algorithms, which can be grouped into two main categories:

- **Linear Classifiers:** They create a linear decision boundary between classes. They are simple and computationally efficient. This category includes Logistic Regression.
- **Non-linear Classifiers:** They can create more complex, non-linear decision boundaries, allowing them to capture more intricate relationships in the data. Examples include Decision Trees, Random Forests, Support Vector Machines (SVM), k-Nearest Neighbors (KNN) and Neural Networks.

In our case, we will use the k-Nearest Neighbors, SVM, and Neural Network algorithms, and thus only these will be described theoretically in the following sections, starting with the first one.

6.1.5 k-Nearest Neighbors (k-NN)

The k-Nearest Neighbors (k-NN) method is one of the simplest algorithms in machine learning and is mainly used for classification problems. The algorithm is based on the simple logic that similar samples are located close to each other in the feature space.

To classify a new, unknown sample, the algorithm identifies the k closest samples from the training set and decides based on the majority of their categories.

k-NN belongs to the category of lazy learners. This means that, unlike other algorithms (such as Decision Trees), it does not build a model during the training phase. Instead, it stores all the training data in memory and "learning" essentially occurs at the time of classification. This results in the training phase being extremely fast, but the classification phase being computationally expensive, especially when the dataset is large.

The algorithm process is as follows:

1. **Definition of k:** The user determines the number of neighbors (k) to be considered.
2. **Distance Calculation:** The distance between the new sample and all training samples is calculated. The most common metric is the Euclidean distance.
3. **Finding Neighbors:** The k training samples with the smallest distance to the new sample are identified.
4. **Voting:** The new sample is classified into the category to which the majority of its k neighbors belong.

The choice of the **k** value is critical to the algorithm's performance. A small k value (e.g., k=1) makes the algorithm sensitive to local noise and can lead to overfitting. Conversely, a very large k value may include samples from other categories, leading to incorrect predictions and underfitting.

The k-Nearest Neighbors algorithm offers several advantages that contribute to its widespread use. Its primary strength is simplicity, as it is extremely easy to understand and implement compared to more complex machine learning models. Additionally, it features fast training since it requires no actual training phase; it simply stores the data for later use. The algorithm also demonstrates robustness, exhibiting good resistance to noisy training data.
However, k-NN has significant disadvantages. The most critical is slow classification, as classifying a new sample requires calculating the distance to all training samples, making it unsuitable for large datasets. It also suffers from sensitivity to k, since its performance heavily depends on choosing the right value for k. Another limitation is memory usage, as it requires storing the entire training set. Finally, it has scale sensitivity, being sensitive to different scales of features, which makes data normalization necessary before application. *(Jadhav, Channe (2016))*

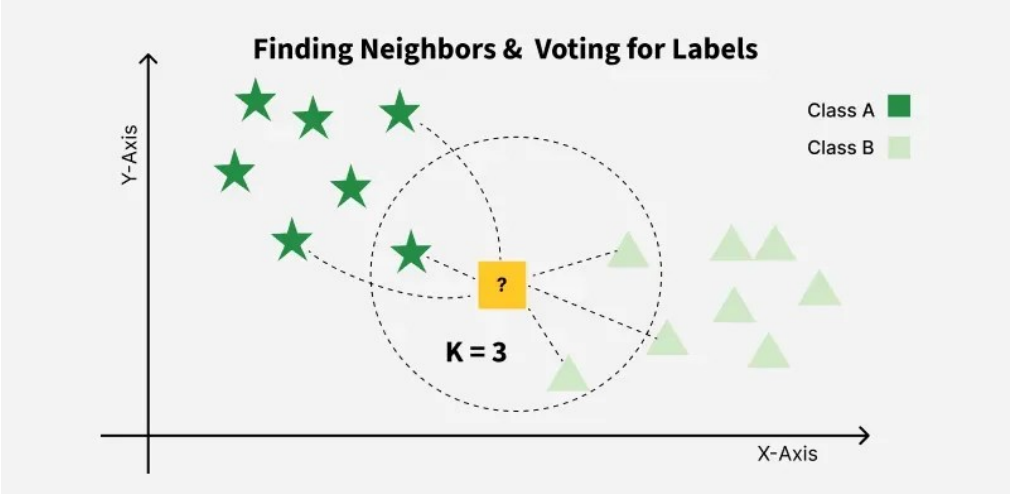

Figure 6.4: Description of the K-NN Method. (Source: https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/)

6.1.6 Support Vector Machines (SVM)

The Support Vector Machines (SVM) algorithm is a machine learning technique primarily used for classification and regression problems. It was first introduced in 1992 by Boser, Guyon, and Vapnik and is based on statistical learning theory. The main goal of SVM is to find the optimal hyperplane that separates data into different categories, maximizing the margin between the closest points of the categories, which are called support vectors.

Unlike other learning algorithms, SVM is based on the Structural Risk Minimization (SRM) principle, which minimizes an upper bound on the expected error, offering better generalization capability compared to the Empirical Risk Minimization (ERM) principle used by traditional neural networks. In its simplest form, SVM seeks to find a linear hyperplane in an N-dimensional space (where N is the number of features) that separates the data into two categories. The hyperplane can be expressed as:

$$\mathrm{w} \cdot \mathrm{x} + b = 0$$

where:

- **w** is the weight vector that determines the orientation of the hyperplane.
- **x** is the input vector.
- **b** is a constant (bias) that determines the position of the hyperplane in space.

The goal of SVM is to find the hyperplane that maximizes the margin between the two categories, i.e., the distance between the hyperplane and the closest points of the two categories. Maximizing the margin is desirable because it offers better generalization and reduces the probability of error on new data. The margin M is given by the equation:

$$M = \frac{2}{\| \mathrm{w} \|}$$

To achieve correct classification of all training data, the following conditions must be satisfied:

- If $y_i = +1$ (positive class): $w \cdot x_i + b \geq 1$
- If $y_i = -1$ (negative class): $w \cdot x_i + b \leq -1$

These can be combined into a single inequality: $y_i(w \cdot x_i + b) \geq 1$ for all i.

In cases where the data are not linearly separable, SVM uses a technique called the kernel trick. The data are transformed into a higher-dimensional space (feature space) where they become linearly separable. The kernel is a function that computes the inner product of the data in the new space, without requiring explicit computation of the transformation. This approach allows for finding non-linear decision boundaries. The most common kernel functions are:

- **Linear Kernel:** $K(x_i, x_j) = x_i \cdot x_j$
- **Polynomial Kernel:** $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$, where d is the degree of the polynomial.
- **RBF Kernel (Radial Basis Function):** $K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2)$, where γ is a parameter that controls the curvature of the decision surface.
- **Sigmoid Kernel (MLP Kernel):** $K(x_i, x_j) = \tanh(\rho(x_i \cdot x_j) + c)$.

In real-world applications, data often contain noise or are not perfectly separable. To address this problem, slack variables ξ_i are introduced, which allow some points to lie on the wrong side of the hyperplane or within the margin. This variant is called soft margin SVM, and the optimization function becomes:

$$\min \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to: $y_i(w \cdot x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

The parameter C controls the trade-off between maximizing the margin and minimizing the classification error. SVM optimization leads to a quadratic programming (QP) problem, which is typically solved through the dual formulation using Lagrange multipliers. The solution to this problem yields:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

where α_i are the Lagrange multipliers. The vectors x_i for which $\alpha_i > 0$ constitute the support vectors and determine the final hyperplane. The decision function for a new point x is given by:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \right)$$

SVM offers significant advantages compared to other learning techniques. It is relatively easy to train, does not get trapped in local optima (as often happens with neural networks), scales well to high-dimensional data, and provides explicit control over the trade-off between complexity and error. However, its performance largely depends on choosing the appropriate kernel function and tuning the parameters, which typically requires experimentation with cross-validation. SVM has been successfully applied to a variety of problems, such as handwritten character recognition, text classification, image analysis, and bioinformatics problems, demonstrating high accuracy and robustness in complex and high-dimensional data. *(Burges (1998))*

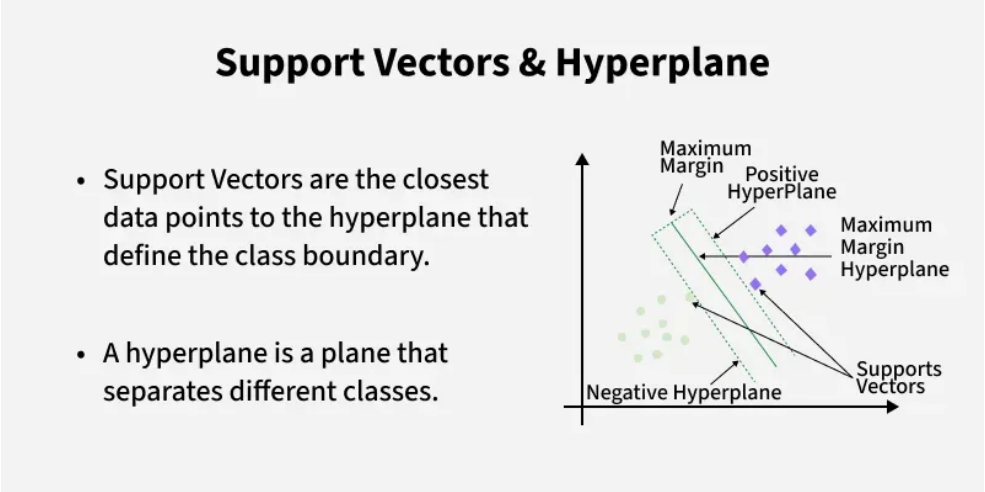

Figure 6.5: Description of SVM Method. (Source: https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/)

6.1.7 Neural Networks

Neural Networks (NNs) are a set of powerful mathematical tools for machine learning, inspired by the way the human brain processes information and learns. Their main strength lies in their ability to identify and learn highly complex, non-linear relationships directly from input-output data, without relying on predetermined equations that describe the system.

The fundamental unit of a neural network is the neuron, which simulates the function of biological neurons. These neurons are organized into distinct layers, creating the network's architecture:

- **Input Layer**: This is the entry point for data into the network. Each neuron in this layer corresponds to a feature of the input data. The number of neurons is determined by the data's dimension.
- **Hidden Layers**: These are located between the input and output layers. Neurons here receive signals from the previous layer, process them, and transmit them to the next layer. Hidden layers are what enable the network to

learn complex, non-linear representations and patterns, making it particularly effective. A network can have one or multiple hidden layers.

- **Output Layer**: This provides the final result or prediction of the network. The number of neurons here depends on the type of problem (e.g., one neuron for binary classification, or multiple neurons for multi-class classification).

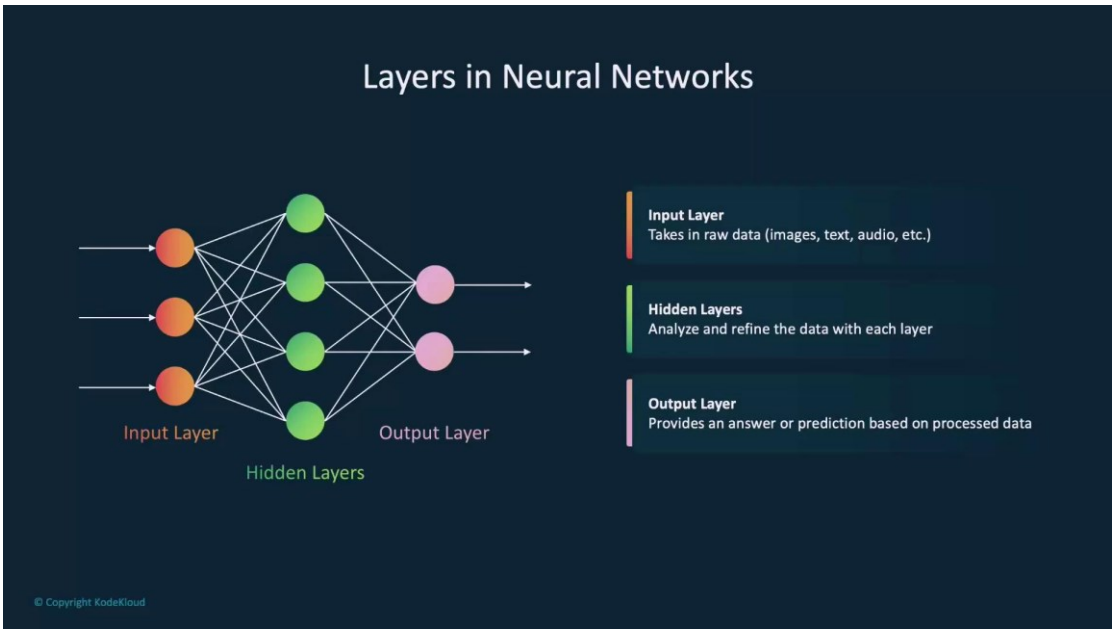

Figure 6.6: Description of Neural Network. (Source: https://notes.kodekloud.com/docs/PyTorch/Building-and-Training-Models/Introduction-to-Neural-Networks/page)

The computation process within a neuron involves two main stages:

1. **Linear Combination**: Each input (x_i) is multiplied by a corresponding weight (w_i). All these products are summed together, and a bias value (b) is added. The result is a linear sum $z = \sum_{i=1}^{n} w_i\, x_i + b$.
2. **Activation Function:** The linear sum z then passes through a non-linear activation function. This function determines whether the neuron will be "activated" or not, introducing non-linearity into the model. This is crucial because it allows the network to learn complex relationships. Some of the most popular functions are:
    - **Sigmoid:** It outputs values between 0 and 1, making it particularly useful for binary classification problems, where the output can be interpreted as a probability. Its formula is:
    $$\sigma(z) = \frac{1}{1 + e^{-z}}$$
    - **Tanh (Hyperbolic Tangent):** It outputs values between -1 and 1. It has the advantage of being zero-centered, which can facilitate learning in subsequent layers. Its formula is:

$$tahn(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- **ReLU (Rectified Linear Unit):** It is one of the most popular activation functions today, especially for hidden layers. It returns the input if it is positive, otherwise it returns 0. Its simplicity and the fact that it does not saturate for positive values make it computationally efficient and help address the vanishing gradient problem. It is defined as:

$$f(z) = \max(0, z)$$

- **Softmax**: This function is used almost exclusively in the output layer for classification problems with more than two categories. It converts a vector of numbers into a vector of probabilities, where their sum is 1, making it clear which class is most likely.

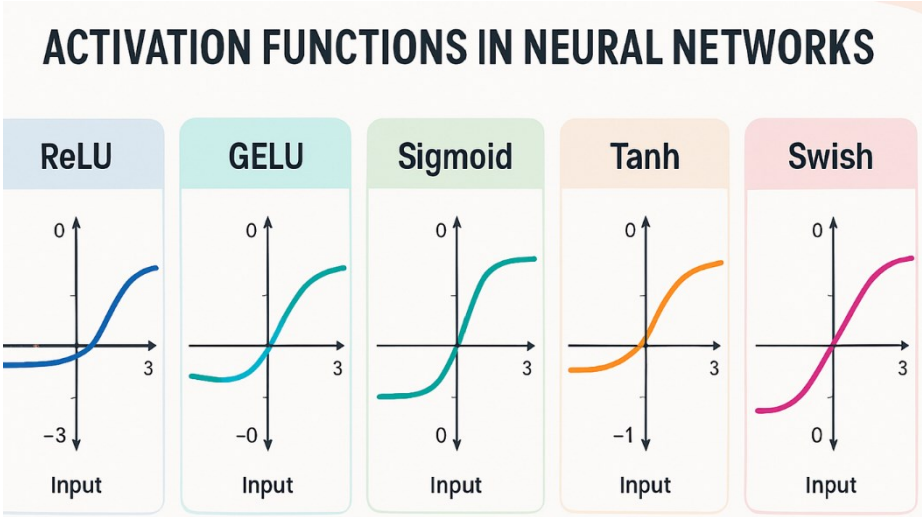

Figure 6.7: Activation Functions of Neural Networks. (Source: https://medium.com/@kiranvutukuri/43-activation-functions-in-neural-networks-eb5f84b0d496)

The training of a neural network is the learning process. The goal is for the network to learn to correctly map input examples to desired output examples. This is achieved through the iterative adjustment of weights and biases. The fundamental algorithm for this process is backpropagation. During backpropagation, the network calculates the error between its prediction and the actual value, and then propagates this error backwards, updating the weights (typically using algorithms such as gradient descent) in order to minimize future error.

There are various neural network architectures, each suitable for different types of problems:

- **Multi-Layer Perceptrons (MLPs):** These belong to the category of feedforward networks, where information moves in only one direction, from input to output, without loops. They are suitable for classification and regression problems.
- **Recurrent Neural Networks (RNNs):** These feature internal feedback, allowing them to process sequential data (e.g., time series, text, speech) where previous information influences the current state. Advanced forms, such

as LSTM and GRU, address the vanishing gradient problem, enabling the learning of long-range dependencies.

- **Convolutional Neural Networks (CNNs):** These are specifically designed for processing data with a topological structure, such as images (although they can also be applied to 1D or 3D data). They use convolutional layers for feature extraction and pooling layers for dimensionality reduction, simulating the function of the visual cortex. *(Chondrodima (2024))*

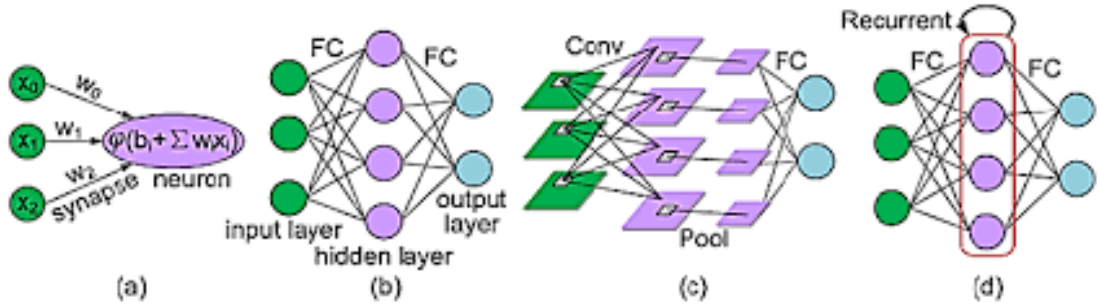

Figure 6.8: Neural network structures. (a) Single neuron. (b) MLP. (c) CNN. (d) RNN. (Source: https://www.researchgate.net/figure/Neural-network-structures-a-Single-neuron-b-MLP-c-CNN-d-RNN-modified-from_fig1_372836166)

6.1.8 Evaluation Metrics

Evaluation metrics are crucial for assessing the performance of a machine learning model, such as those we have described and will use ourselves. Depending on the type of problem (binary classification, multi-class classification, regression, etc.), certain metrics are more suitable than others. Below, we analyze the key metrics commonly used in classification problems.

- **Accuracy:** The percentage of correct predictions relative to the total number of predictions.
- **Sensitivity or Recall:** The percentage of actual positive instances that were correctly identified.
- **Specificity:** The percentage of actual negative instances that were correctly identified.
- **Precision:** The percentage of positive predictions that were actually positive.
- **F1-score:** A combination of precision and recall that provides a balanced view of performance.
- **Youden's Index:** A metric that gives equal weight to accuracy within positive and negative instances.
- **Cohen's Kappa (k):** Measures the agreement between predicted and actual classes, accounting for the possibility of chance agreement.
- **Matthews Correlation Coefficient (MCC):** Measures the correlation between actual and predicted values.

6.1.8.1 Accuracy

Accuracy is the most common evaluation metric and refers to the percentage of correct predictions made by the model relative to the total number of predictions. It is calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

or equivalently:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- **TP (True Positives):** Observations that are positive and were correctly predicted as positive.
- **TN (True Negatives):** Observations that are negative and were correctly predicted as negative.
- **FP (False Positives):** Observations that are negative but were incorrectly predicted as positive.
- **FN (False Negatives):** Observations that are positive but were incorrectly predicted as negative.

6.1.8.2 Sensitivity and Specificity

Sensitivity, often referred to as Recall or True Positive Rate, measures the proportion of actual positive instances that the model correctly identified. It is calculated as:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

A high sensitivity value indicates that the model identifies most of the actual positive instances, avoiding false negatives.

Specificity, or True Negative Rate, measures the proportion of actual negative instances that the model correctly identified. It is calculated as:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

A high specificity value indicates that the model identifies most of the actual negative instances, avoiding false positives. These two metrics are often used as a pair and reveal more about the model than accuracy alone, especially when the data is imbalanced.

6.1.8.3 Precision

Precision, or Positive Predictive Value, measures the proportion of positive predictions that were actually positive. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high precision value indicates that few of the positive predictions were actually negative (i.e., few false positives).

6.1.8.4 F1-Score

The F1-Score is a metric that combines precision and recall into a single measure, providing a balanced assessment of model performance. It is calculated as the harmonic mean of the two:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score is particularly useful when we want an overall picture of model performance, considering both its ability to avoid false positives (precision) and its ability to identify all actual positive instances (recall). It is preferred over accuracy in problems with class imbalance.

6.1.8.5 Youden's Index

Youden's Index is another metric derived from the confusion matrix and is defined as:

$$\text{Youden's Index} = \text{Sensitivity} + \text{Specificity} - 1$$

This index gives equal weight to the accuracy within positive (sensitivity) and negative (specificity) instances, regardless of their count, and is often used to find the optimal classification threshold.

6.1.8.6 Cohen's Kappa (k)

Cohen's Kappa (k) coefficient measures the degree of agreement between the predicted and actual classes, taking into account the probability of agreement that could occur by chance. It is calculated as:

$$\kappa = \frac{\text{Accuracy} - p_e}{1 - p_e}$$

where p_e is the probability of chance agreement. Values close to 1 indicate near-perfect agreement, while values near 0 or negative values indicate agreement no better than chance or worse.

6.1.8.7 Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient (MCC) is a more balanced metric that takes into account all four elements of the confusion matrix (TP, TN, FP, FN) and essentially

measures the correlation between the observed and predicted binary classes. It is calculated as:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC takes values from -1 (complete disagreement) to +1 (perfect agreement), with 0 indicating a random prediction. It is considered particularly reliable even in cases of severe class imbalance. *(Rainio, Teuho, Klén (2024))*

6.2 Applications to European cup teams data

In this section, the procedures described theoretically above will be implemented. Specifically, after determining the features with which we will work, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), and Neural Network models will be developed, aiming to predict the qualification of teams from the group stage based on the available features. For the sake of completeness and comparison, each machine learning model will be evaluated under two different variable selection scenarios: initially, it will be applied using the feature selection derived from the Boruta algorithm, and subsequently, it will be trained anew using exclusively those features that emerged as statistically significant in the corresponding logistic regression model.

6.2.1 Feature selection using the Boruta method

In this initial introductory part of the research, as mentioned, there are many ways and procedures to carry out, each with its own characteristics. We will use the Boruta method, which we described theoretically. Let us remind you that we have only taken the variables per match. Also, we will not use variables such as Goals, Right Foot, Left Foot, Goals Inside Area, Goals Outside Area because the qualification of the teams is determined largerly by the number of goals each team scores. For the same reason, the variable Assists was also excluded from the dataset, as it constitutes a direct derivative of goal-scoring actions. The only variable we kept that is related to goals is Goals Conceded. Applying the Boruta method, the features that have been selected are the following:

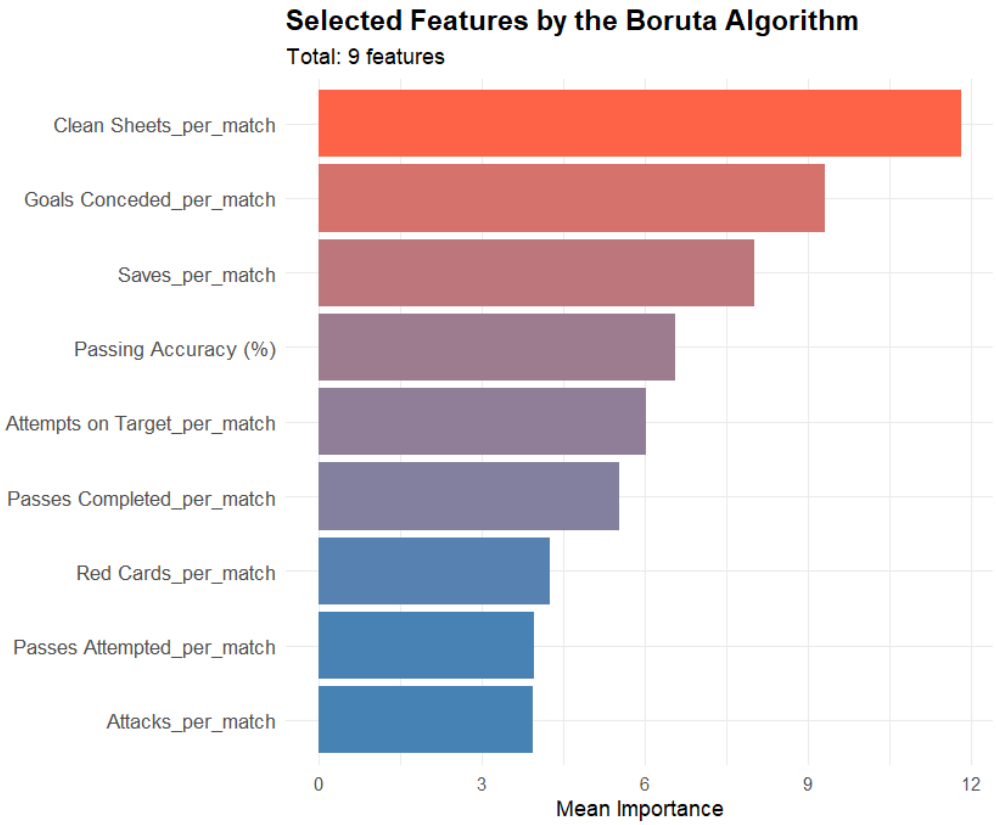

Figure 6.9: Selected features identified by the Boruta algorithm, ordered by mean importance.

Without the direct goal variables, the Boruta algorithm highlighted a set of features that describe the process leading to a victory rather than the outcome itself. Specifically, offensive threat and chance creation are captured through Attempts on Target and Attacks. Ball possession and game control are expressed by Passes Attempted, Passes Completed, and Passing Accuracy. Defensive solidity is described by Saves, Clean Sheets, and Goals Conceded (as a negative performance indicator). Finally, discipline and on-field behavior are influenced by Red Cards.

6.2.2 k-NN implementation for predicting qualification

Having determined the sets of input variables in the previous section, we now proceed to the implementation of the k-Nearest Neighbors (k-NN) algorithm. In this process, we considered it useful for validating the results to employ the 10-fold cross-validation technique. In this method, the dataset is randomly divided into 10 equal-sized subsets. Each time, 9 of these subsets are used as the training set and 1 as the test set. The process is repeated 10 times, so that each subset is used exactly once as a test set. 10-fold cross-validation is a particularly useful technique when we want to fully utilize a dataset and obtain a reliable estimate of the model's performance while limiting the risk of overfitting. It is suitable for datasets like ours, offering a balance between estimation accuracy and computational cost. *(Kaligeris 2025)*

In line with the methodology outlined in Section 6.2, the k-NN model will be trained and evaluated under two distinct feature selection scenarios: first using the nine features identified by the Boruta algorithm and subsequently using the subset of features retained from the logistic regression model. Before proceeding to the prediction of team qualification, we must first determine the optimal number of neighbors, k. The optimal *k* value will be selected based on the highest Area Under the Curve (AUC) score obtained during cross-validation. The AUC, which represents the area under the Receiver Operating Characteristic (ROC) curve, provides a quantitative measure of the model's overall ability to discriminate between the two classes (Group Stage vs. Advanced), with values closer to 1 indicating superior classification performance.

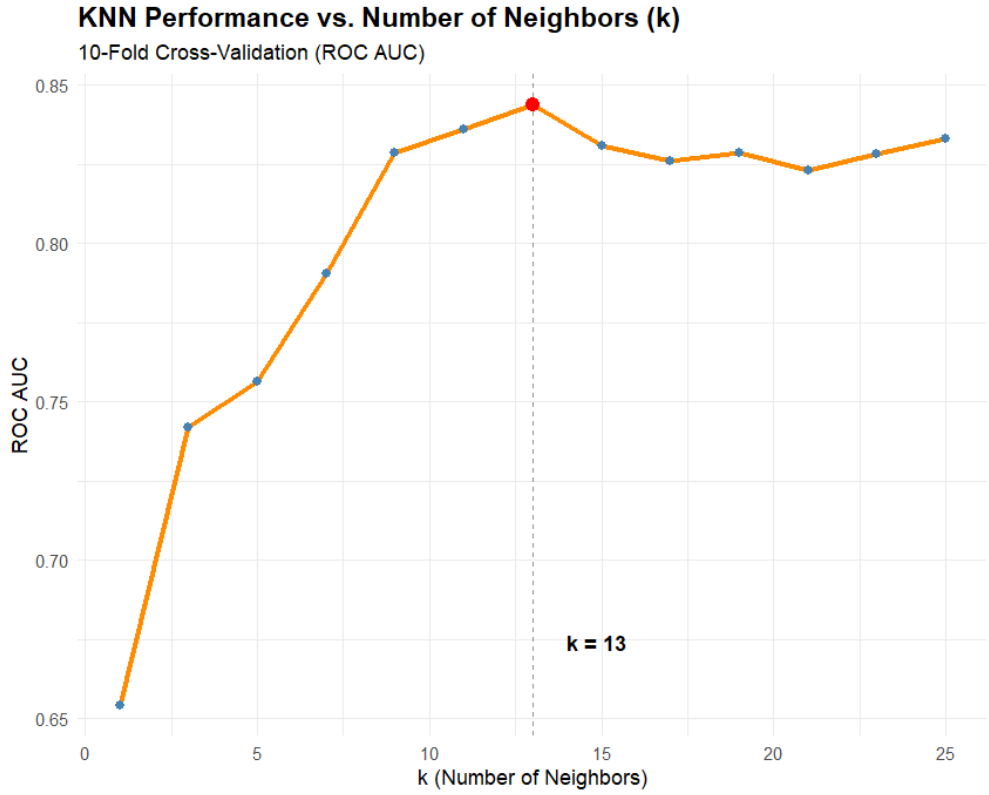

Figure 6.10: Finding the optimal k

After applying the k-NN algorithm with the optimal parameter k=13 (as determined by the optimization process), the model demonstrated satisfactory predictive performance. The evaluation of its performance is based on the confusion matrix and a series of classification metrics.

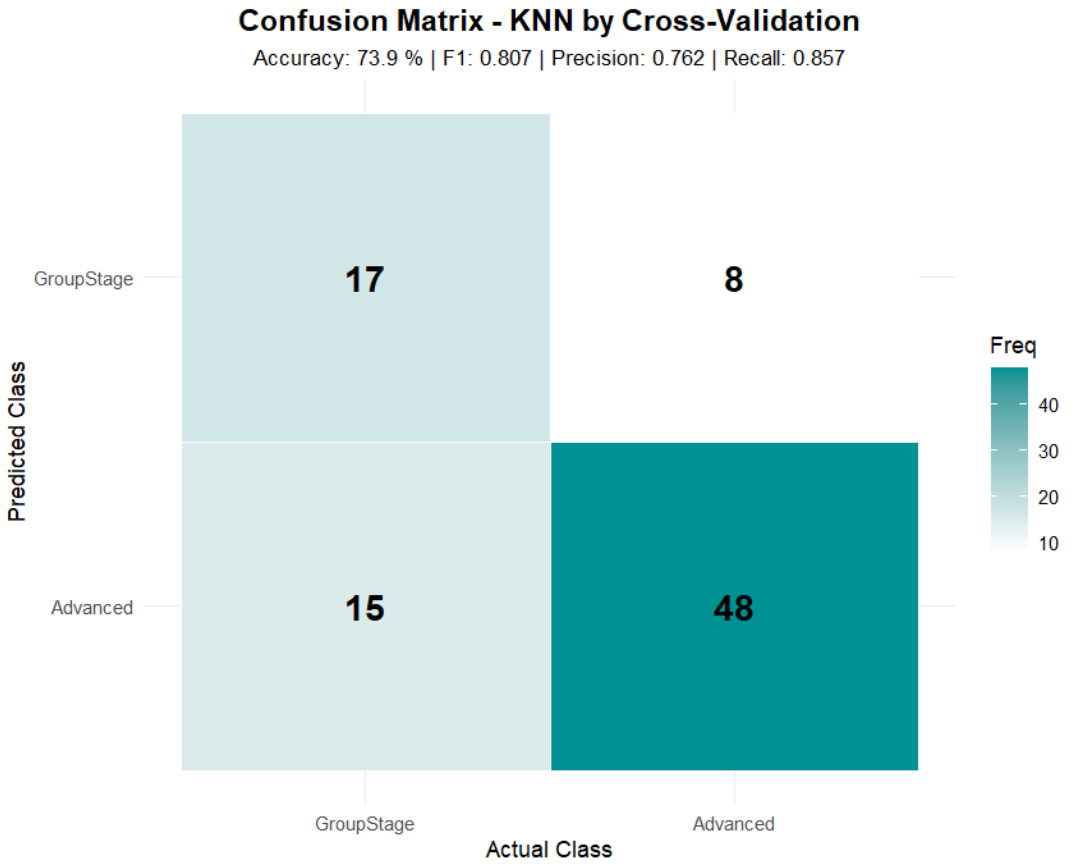

Figure 6.11: Confusion matrix of the k-NN model (k=13) for predicting qualification from the group stage

The confusion matrix presented above provides a detailed analysis of the classification performance of the KNN model using the features selected by the Boruta algorithm. Specifically, the model correctly predicted that 17 teams would not advance from the group stage (True Negatives), while successfully identifying 48 teams that advanced to the knockout phase (True Positives). However, classification errors were also observed: the model incorrectly predicted that 15 teams would advance when in reality they remained in the group stage (Type I Error), and conversely, it erroneously estimated that 8 teams would be eliminated when they actually progressed (Type II Error). Overall, the high number of True Positives indicates that the model exhibits satisfactory sensitivity in detecting advancing teams, although it demonstrates a tendency towards optimistic predictions due to the elevated number of False Positives. Below, a table is presented with the metrics for evaluating our model.

Metric	Value
Accuracy	0.739
Sensitivity/Recall	0.857
Specificity	0.531
Precision	0.762
F1-Score	0.807
Youden's Index	0.388
Cohen's Kappa (k)	0.407
Matthews Correlation Coefficient (MCC)	0.414
AUC	0.830

Table 6.1: Performance metrics for the k-NN model (Boruta Features)

The data presented in the table above yield useful insights into the predictive behavior of the model. The overall Accuracy stands at 73.9%, a value indicating satisfactory general performance. However, the true dynamics of the model are revealed by the individual metrics. The Recall is notably high (85.7%), signifying that the model is extremely effective at identifying teams that ultimately advance (low False Negative rate). Conversely, Specificity is markedly lower (53.1%). This asymmetry indicates that the model tends to be "optimistic," more frequently predicting advancement even for teams that are eventually eliminated (elevated False Positives).

The Precision value of 76.2% confirms that when the model predicts a team will advance, the prediction is quite reliable. The balance between Precision and Recall is reflected in the F1-Score (0.807), which is considered highly satisfactory for sports forecasting data. Finally, the aggregate metrics—Youden's Index (0.388), Cohen's Kappa (0.407), and MCC (0.414) —fall within a moderate range. These values underscore the model's difficulty in handling both classes with equal success, primarily due to its limited capacity to correctly identify teams that remain in the group stage. In conclusion, the k-NN model utilizing Boruta features serves as a reliable “detector” of advancing teams, albeit at the cost of a relatively high number of false alarms.

The Area Under the ROC Curve (AUC) was calculated at 0.83, a value that classifies the model as having good discriminative ability. However, it is essential to interpret this metric in conjunction with the individual performance indicators presented in the table. Although an AUC of 0.83 suggests that the model separates the two classes reasonably well in terms of predicted probabilities, the low Specificity (53.1%) and moderate MCC (0.414) reveal that, in practice, the classification is not equally effective for both categories. Specifically, the high AUC is largely driven by the excellent Recall (85.7%), meaning the model is adept at identifying advancing teams. Conversely, its weakness in correctly identifying eliminated teams (low Specificity) limits the overall practical utility of the model, as it results in a high number of False Positive predictions.

Subsequently, the k-Nearest Neighbors (k-NN) model will be trained and evaluated using the alternative set of variables derived from the logistic regression framework. As established in the preceding analysis, and after further experimentation and refinement concerning the selection of predictors, we ultimately decided to proceed with the exact same variables that were employed in the multinomial logistic model. These are: Attempts on Target per match, Blocked per match, Attacks per match, Goals Conceded per match, and Saves from Penalties per match. The aim of this section is to investigate whether this more parsimonious set of features can achieve comparable or improved predictive performance relative to the broader set selected by the Boruta algorithm.

As will be shown in the following section, this model yields very poor predictive performance. It is included purely for comparison purposes, since the other machine learning models—specifically SVM and neural networks—achieved highly satisfactory results, and we wish to contrast the outcomes. Before proceeding to the prediction of team qualification using this alternative feature set, the optimal number of neighbors, k, must first be determined; this value was found to be 25.

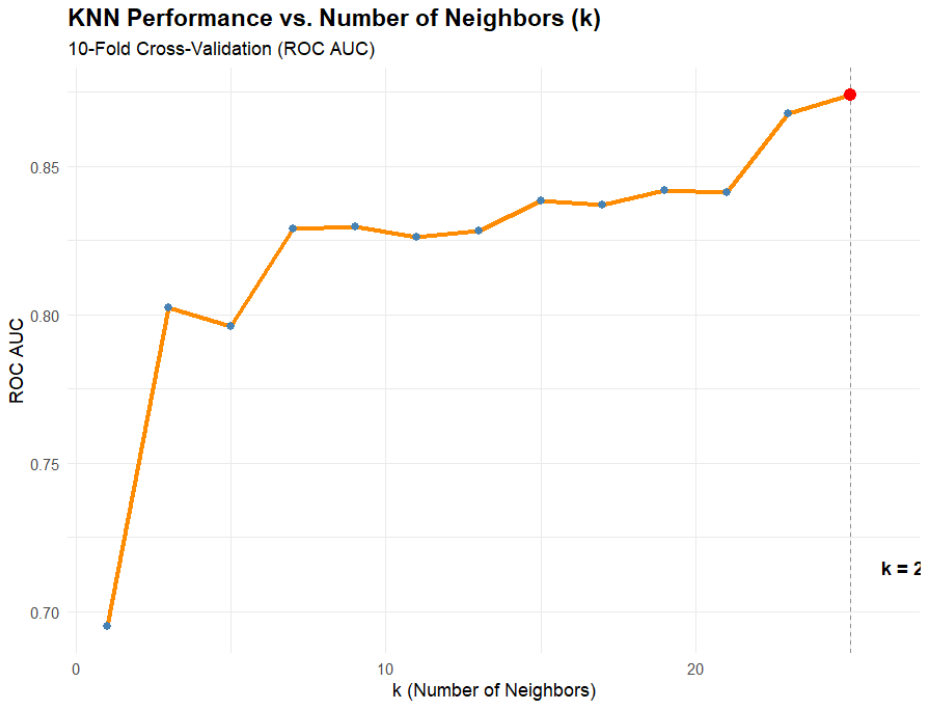

Figure 6.12: Finding the optimal k

After the optimal k was determined, we obtain the following results from the confusion matrix.

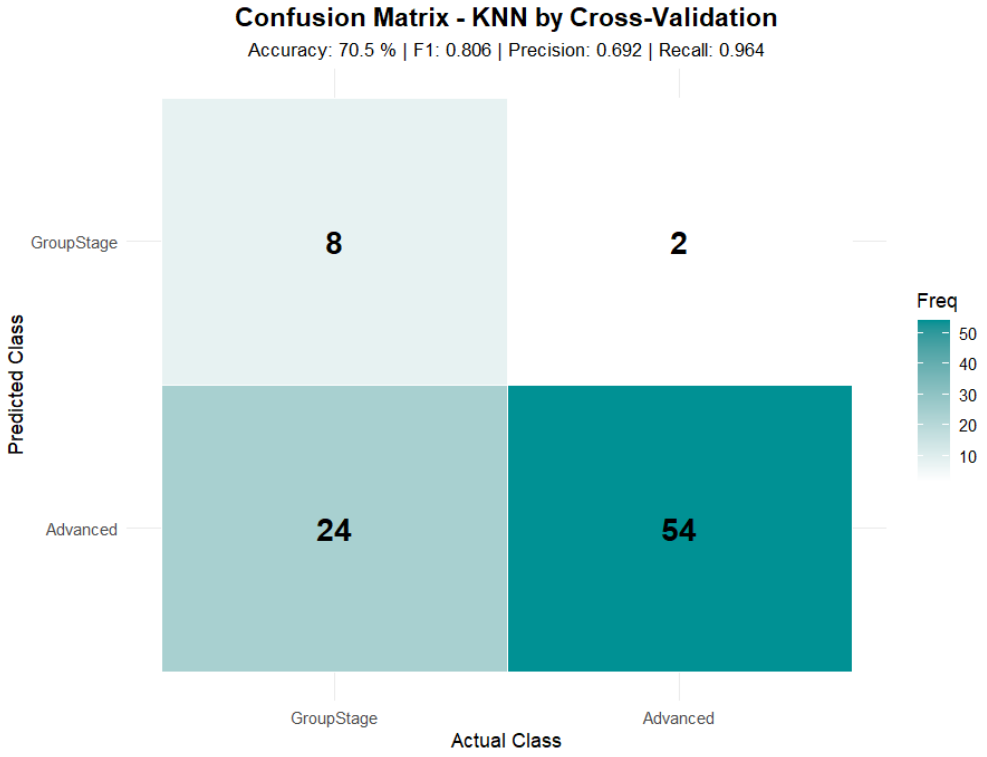

Figure 6.13: Confusion matrix for the k-NN model using the multinomial logistic regression feature set

The confusion matrix reveals the model's classification behavior using this restricted set of predictors. Specifically, the model correctly identified 8 teams that failed to advance from the group stage (True Negatives) and 54 teams that successfully reached the knockout phase (True Positives). However, it also produced 24 False Positives—incorrectly predicting advancement for teams that were eliminated—and only 2 False Negatives, where it failed to identify teams that actually advanced. This pattern indicates that while the model excels at capturing advancing teams (missing only 2 qualifications), it exhibits a notable tendency toward optimistic predictions, as reflected in the relatively high number of False Positives.

Metric	Value
Accuracy	0.705
Sensitivity/Recall	0.964
Specificity	0.250
Precision	0.692
F1-Score	0.806
Youden's Index	0.214
Cohen's Kappa (k)	0.251
Matthews Correlation Coefficient (MCC)	0.325

Table 6.2: Performance metrics for the k-NN model (Multinomial Logistic Regression Features)

The data presented in the table above yield useful insights into the predictive behavior of the model. The overall Accuracy stands at 70.5%, a value that might appear moderately acceptable at first glance. However, the true dynamics of the model are revealed by the individual metrics. The Recall is exceptionally high (96.4%), signifying that the model is almost perfect at identifying teams that ultimately advance—missing only a tiny fraction of qualifiers (extremely low False Negative rate). Conversely, Specificity is extremely low (25.0%). This stark asymmetry indicates that the model adopts a heavily "optimistic" stance, overwhelmingly predicting advancement even for teams that are eventually eliminated, resulting in a large number of False Positives.

The Precision value of 69.2% confirms that when the model predicts a team will advance, the prediction is correct roughly seven out of ten times—meaning that about one in three positive predictions is a false alarm. The balance between Precision and Recall is captured by the F1-Score (0.806), which remains superficially decent, but this figure is heavily inflated by the near-perfect Recall and masks the very poor performance on the negative class. Finally, the aggregate metrics—Youden's Index (0.214), Cohen's Kappa (0.251), and Matthews Correlation Coefficient (0.325)—fall within a low range, well below what would be considered acceptable for a balanced classifier. These values underscore the model's severe difficulty in handling both classes with equal success, primarily due to its minimal capacity to correctly identify teams that remain in the group stage.

In conclusion, the model functions as an extremely aggressive detector of advancing teams, successfully capturing nearly 97% of qualifiers, but does so at the cost of a very high false-alarm rate. Compared to the earlier Boruta-based model, this version pushes the "optimistic" bias even further, rendering it practically incapable of reliably distinguishing group-stage teams from those that advance.

As noted earlier, we anticipated encountering a model with poor classification performance, and indeed, we conclude that the first model (Boruta-based) is overall superior. Although it sacrifices a small degree of sensitivity, it gains substantially in the reliability of negative predictions and in overall probability calibration. It therefore constitutes a more reliable and balanced predictive mechanism, suitable for practical use.

6.2.3 SVM implementation for predicting qualification

Following the same methodological framework, we proceed to the implementation of the Support Vector Machine (SVM) algorithm for predicting a team's qualification from the group stage. SVM is a powerful classification technique that aims to find the optimal hyperplane which best separates the classes in the feature space. For non-linearly separable problems, SVM utilizes kernel functions to transform the data into a higher-dimensional space where linear separation becomes possible. Similarly to k-NN, we employ 10-fold cross-validation to ensure reliable performance estimation and to limit the risk of overfitting. Initially, we will use the set of variables obtained from Boruta, and subsequently we will employ the variables that were used in the multinomial logistic model.

Unlike k-NN, which only requires tuning of the k parameter, SVM involves selecting both the kernel function and its associated hyperparameters. In our implementation, we used the Radial Basis Function (RBF) kernel and, through 10-fold cross-validation, determined the optimal combination of parameters using the AUC metric as the criterion here as well. The optimal parameters obtained were C = 0.25 and sigma = 0.1093.

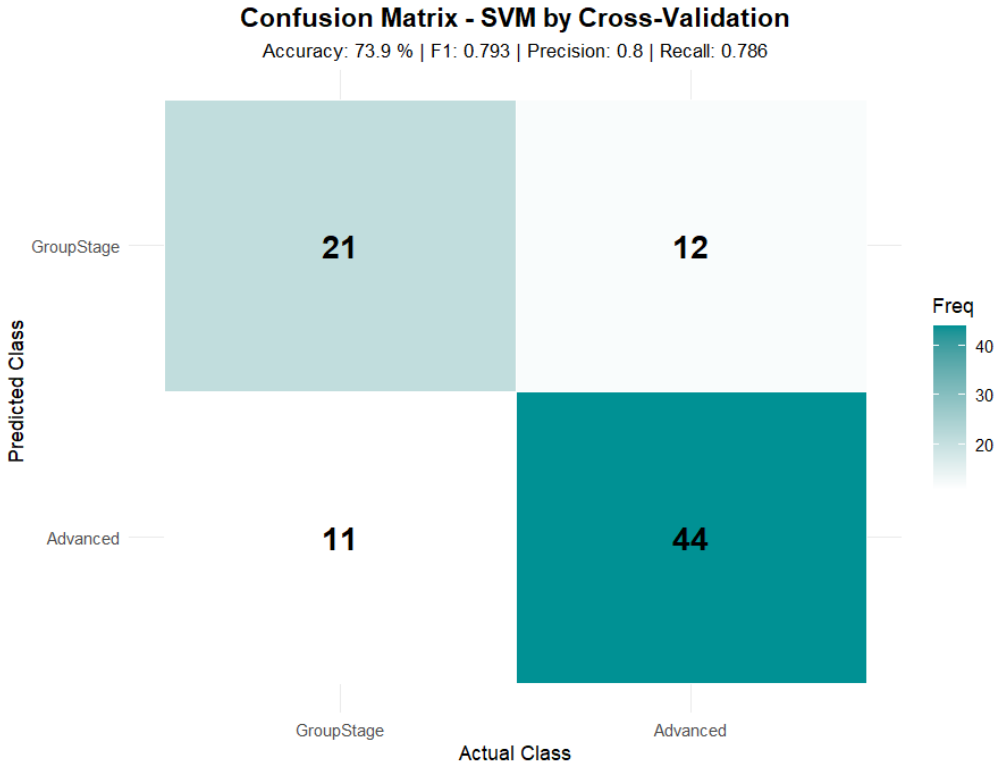

Figure 6.14: Confusion matrix of the SVM model for predicting qualification from the group stage

After applying the SVM algorithm with the optimal parameters (RBF kernel, C = 0.25 and sigma = 0.1093), the model demonstrated its predictive performance. The evaluation of its performance is based on the confusion matrix and a series of classification metrics.

The confusion matrix reveals the model's classification behavior using this set of predictors. Specifically, the model correctly identified 21 teams that failed to advance from the group stage (True Negatives) and 44 teams that successfully reached the knockout phase (True Positives). However, it also produced 11 False Positives—incorrectly predicting advancement for teams that were eliminated—and 12 False Negatives, where it failed to identify teams that actually advanced. This pattern indicates that while the model captures a substantial proportion of advancing teams (missing 12 qualifications), it still exhibits a certain tendency toward optimistic predictions, as reflected in the 11 False Positives.

Metric	Value
Accuracy	0.739
Sensitivity/Recall	0.786
Specificity	0.656
Precision	0.800
F1-Score	0.793
Youden’s Index	0.442
Cohen’s Kappa (k)	0.439
Matthews Correlation Coefficient (MCC)	0.439
AUC	0.856

Table 6.3: Performance metrics for the SVM model (Boruta Features)

The data presented in the table above yield useful insights into the predictive behavior of the model. The overall Accuracy stands at 73.9%, a value indicating satisfactory general performance. However, the true dynamics of the model are revealed by the individual metrics. The Recall stands at 78.6%, still high but notably more restrained, signifying that the model is effective at identifying advancing teams while missing a moderate share of them. More importantly, Specificity reaches 65.6%, a substantial improvement that demonstrates the model's markedly better capacity to correctly identify teams that fail to progress from the group stage. This far more balanced profile indicates that the SVM mitigates the excessive “optimism” observed in earlier models, producing fewer false alarms.

The Precision value of 80.0% confirms that when the model predicts a team will advance, the prediction is correct four out of five times—a meaningful gain in the reliability of positive predictions. The F1-Score (0.793), though marginally lower than before, remains highly satisfactory for sports forecasting data and reflects a healthier equilibrium between Precision and Recall. Finally, the aggregate metrics—Youden's Index (0.442), Cohen's Kappa (0.439), and MCC (0.439)—all register clear improvements and now fall within a moderate-to-solid range. These values underscore the model's enhanced ability to handle both classes with considerably greater parity. In conclusion, the SVM model utilizing Boruta features serves as a significantly more balanced and trustworthy classifier, effectively identifying advancing teams while maintaining a meaningful ability to correctly flag group-stage eliminations.

The Area Under the ROC Curve (AUC) reached 0.856, indicating that the model possesses very good discriminatory power. Nevertheless, a meaningful assessment requires examining this metric alongside the individual performance measures shown in the table. While an AUC of 0.856 implies that the model effectively distinguishes between the two classes based on predicted probabilities, the noticeably higher Specificity (65.6%) and the moderate-to-acceptable MCC (0.439) indicate that the classification has become more balanced across both categories in practice. More precisely, the elevated AUC is underpinned by both a strong Recall (78.6%), demonstrating the model's proficiency in detecting teams that advance, and an improved ability to correctly identify eliminated teams (higher Specificity). This substantially curtails false positive predictions and enhances the overall practical value of the model.

Let us now examine what happens to the model when we use the variables from the multinomial logistic model. In this case, C and sigma are 2 and 0.206, respectively.

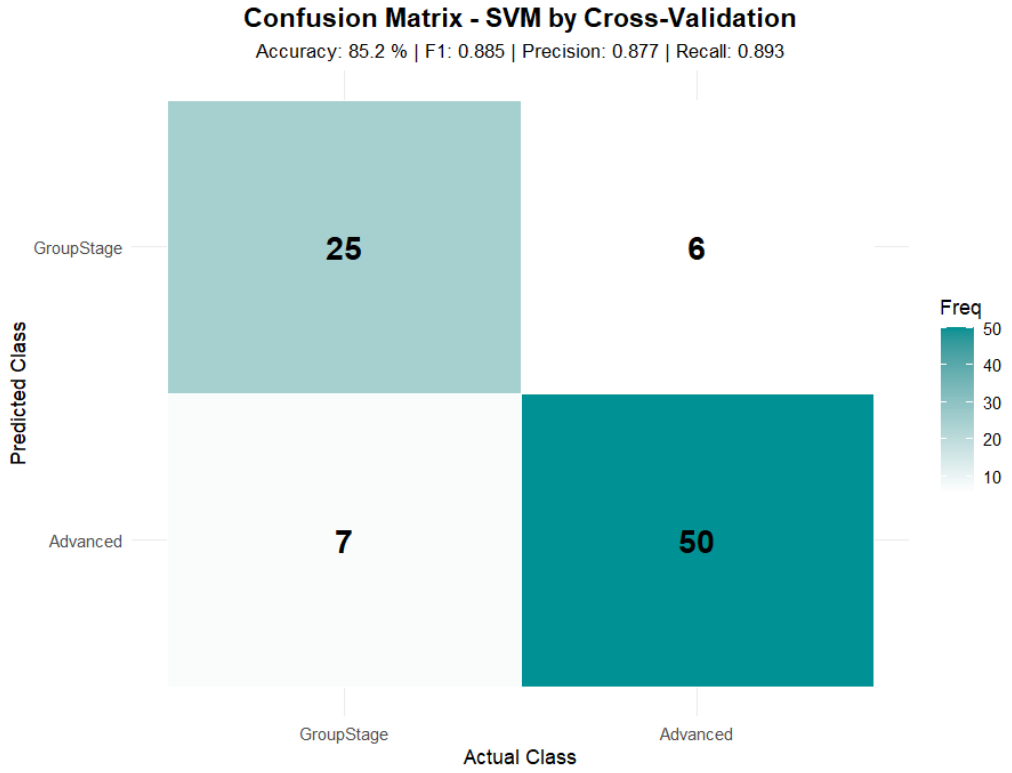

Figure 6.15: Confusion matrix of the SVM model for predicting qualification from the group stage (Multinomial Logistic Regression features)

The confusion matrix reveals the model's classification behavior using this set of predictors. Specifically, the model correctly identified 25 teams that failed to advance from the group stage (True Negatives) and 50 teams that successfully reached the knockout phase (True Positives). However, it also produced 7 False Positives—incorrectly predicting advancement for teams that were eliminated—and 6 False Negatives, where it failed to identify teams that actually advanced. This pattern indicates that the model captures an excellent proportion of advancing teams (missing only 6 qualifications) and, at the same time, maintains a solid ability to correctly identify teams that are eliminated, as reflected in the low number of False Positives (only 7).

Metric	Value
Accuracy	0.852
Sensitivity/Recall	0.893
Specificity	0.781
Precision	0.877
F1-Score	0.885
Youden's Index	0.674
Cohen's Kappa (k)	0.679
Matthews Correlation Coefficient (MCC)	0.679

Table 6.4: Performance metrics for the SVM model (Multinomial Logistic Regression features)

The data presented in the table above provide extremely encouraging evidence regarding the predictive behavior of the model. The overall Accuracy stands at 85.2%, a value indicating very good general performance. Beyond the overall picture, however, the model's substantial superiority is revealed through the individual metrics. Recall reaches 89.3%, meaning that the model identifies nearly nine out of ten teams that ultimately advance (very low False Negative rate). At the same time, Specificity stands at 78.1%, demonstrating a now remarkable ability to correctly identify teams that are eliminated in the group stage. This balance between Recall and Specificity reflects a classifier that avoids excessive optimism and handles both classes reliably.

The Precision value of 87.7% confirms that when the model predicts advancement, the prediction is correct nearly nine out of ten times—an exceptionally high reliability of positive predictions. The F1-Score (0.885) reflects an exemplary balance between Precision and Recall and constitutes a top-tier performance for sports forecasting data. Finally, the aggregate metrics confirm the image of a strong model: Youden's Index (0.674) approaches 0.7, indicating very good discriminative power; Cohen's Kappa (0.679) corresponds to substantial agreement beyond chance; and the Matthews Correlation Coefficient (MCC = 0.679) lies at a high level, demonstrating strong correlation between actual and predicted classes.

The Area Under the ROC Curve (AUC) was calculated at 0.896, a value that classifies the model as having excellent discriminative ability. However, it is essential to interpret this metric in conjunction with the individual performance indicators presented in the table. An AUC of 0.896 indicates that the model separates the two classes very successfully in terms of predicted probabilities. In contrast to previous models, this excellent value is not driven one-sidedly by high Recall, but is also substantially supported by Specificity (78.1%), which is now strong. This is also reflected in the very satisfactory MCC (0.679). Specifically, the model combines very good Recall (89.3%), meaning it effectively identifies nearly nine out of ten teams that advance, with a significantly enhanced capacity to correctly recognize teams that are eliminated (high Specificity), drastically reducing false positive errors. The result is a model with excellent overall discriminative power and genuinely improved practical utility.

In conclusion, the SVM model that employs the variables from the multinomial logistic model constitutes the most balanced and reliable classifier among those examined so far. It simultaneously achieves high detection of teams that advance and satisfactory

identification of teams that are eliminated, minimizing both false hopes (false positives) and missed qualifications (false negatives).

6.2.4 Neural Network implementation for predicting qualification

In this part, we apply a neural network to compare it with what we did above. In this specific project, we use 10-fold cross-validation to evaluate the model, selecting the same features as in the previous models. The neural network we constructed consists of two hidden layers with 32 and 16 neurons respectively, ReLU activation function in

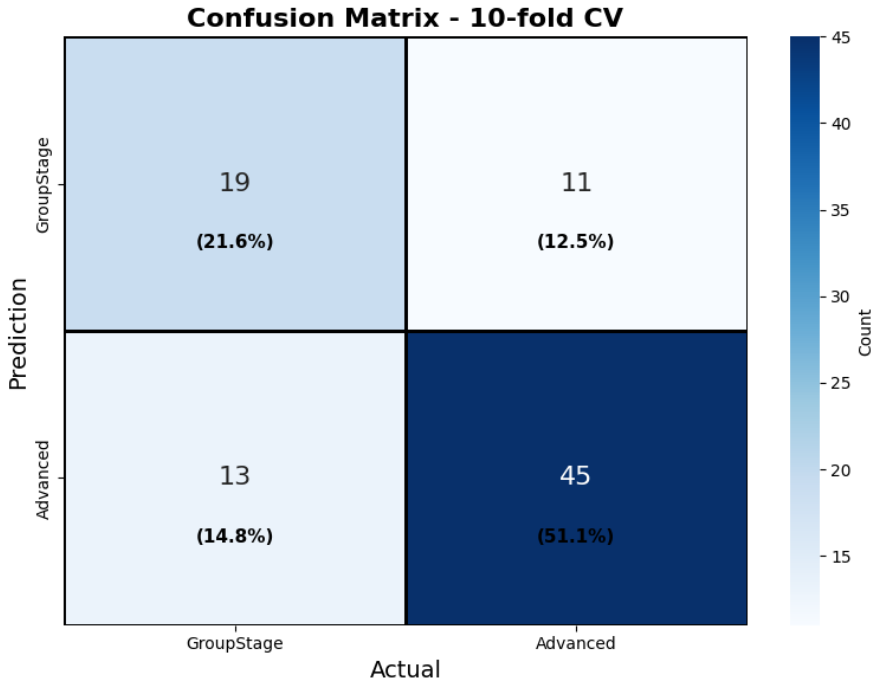

Figure 6.16: Confusion matrix of the Neural Network model for predicting qualification from the group stage (Boruta features)

the hidden layers, and sigmoid in the output layer for binary classification. To avoid overfitting, dropout of 0.5 was used after each hidden layer. At the beginning we will make a prediction with the variables we obtained from Boruta, thus we have this confusion matrix.

From the confusion matrix we see that the model correctly predicted 19 teams that were eliminated (True Negatives) and 45 that advanced (True Positives), while it made 11 errors by predicting advancement for teams that were ultimately eliminated (False Positives) and 13 errors by predicting elimination for teams that advanced (False Negatives). This means that the model tends to be slightly "pessimistic", seeing advancement even for teams that get eliminated, but at the same time it also misses a significant number of actual advancements.

Metric	Value
Accuracy	0.727
Sensitivity/Recall	0.804
Specificity	0.594

Precision	0.776
F1-Score	0.789
Youden's Index	0.397
Cohen's Kappa (k)	0.403
Matthews Correlation Coefficient (MCC)	0.403
AUC	0.791

Table 6.5: Performance metrics for the Neural Network (Boruta features)

Overall accuracy reaches 72.7%, meaning roughly 7 out of 10 predictions are correct – a moderate result that leaves room for improvement. Sensitivity (Recall) is 80.4%, so the model correctly identifies four out of five teams that advanced, while specificity is lower, at 59.4%, indicating that it struggles more to identify teams that will stay in the group stage. When the model predicts advancement (Advanced), the prediction is correct in 77.6% of cases (Precision), which gives relative reliability to its positive predictions. The harmonic mean of sensitivity and precision (F1-score) stands at 78.9%, confirming a balance between detecting advancements and the reliability of those predictions. Youden's index (0.397) and Matthews correlation coefficient (MCC = 0.403) show that the model distinguishes between the two classes slightly better than chance but not strongly, while Cohen's Kappa (0.403) indicates moderate agreement beyond chance.

The area under the ROC curve (AUC) reaches 0.791, which is considered good discriminatory ability: if we randomly select a team that advanced and one that was eliminated, the model has about a 79% probability of giving a higher score to the team that actually advanced. Overall, the model makes significant use of the features provided (shots on target, passes, attacks, etc.) and offers moderate to good performance, suitable for an initial level of evaluation; however, its tendency for optimistic predictions and the missed eliminations must be taken into account when interpreting the results.

In the image above, two graphs are presented, illustrating the evolution of accuracy and loss during the training of a machine learning model over the epochs using the 10-fold cross-validation method. In the left graph, the blue line represents the model's accuracy on the training set (train accuracy), while the red line represents the accuracy on the validation set (validation accuracy). We observe that accuracy gradually increases over the epochs, starting from approximately 0.50 and reaching values close

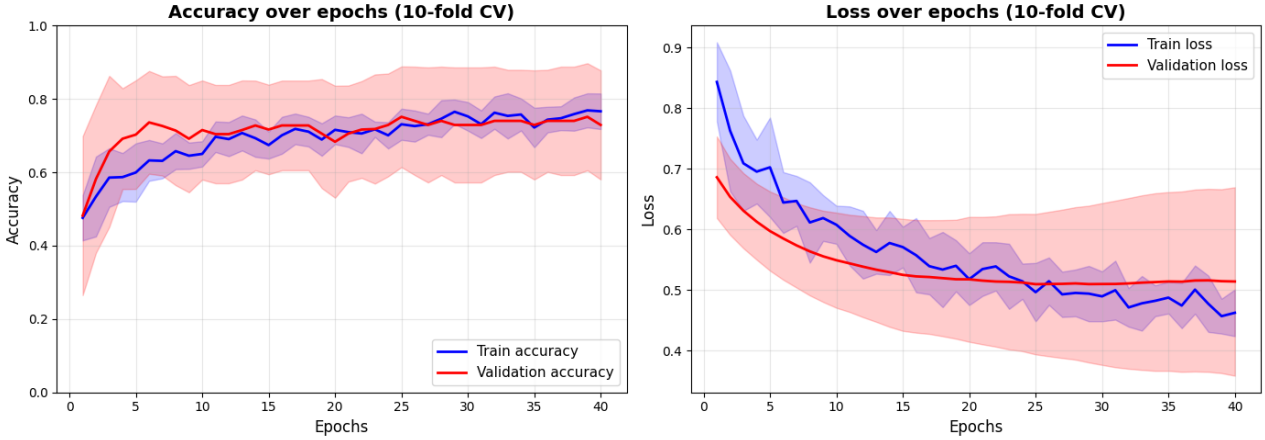

Figure 6.17: Neural network model Accuracy and Loss graphs

to 0.75–0.80. The two curves follow a similar trajectory and remain relatively close to each other, indicating that the model learns effectively from the data without showing strong signs of overfitting. The shaded areas around the curves represent the variability of the results across the different folds of the cross-validation process. The right graph illustrates the evolution of loss during training. The blue line corresponds to the training loss, while the red line represents the validation loss. The training loss steadily decreases over the epochs, dropping from values around 0.85 to below 0.50, which indicates that the model progressively improves its fit to the training data. The validation loss also decreases during the early epochs and then stabilizes around 0.50, remaining slightly higher than the training loss throughout the training process. Overall, the relatively small gap between the training and validation curves in both graphs suggests that the model demonstrates satisfactory generalization ability and does not suffer from significant overfitting.

Now let's do the same procedure with the variables we obtained from the multinomial logistic model, just as we did with the previous models, so that they can be compared with each other afterwards.

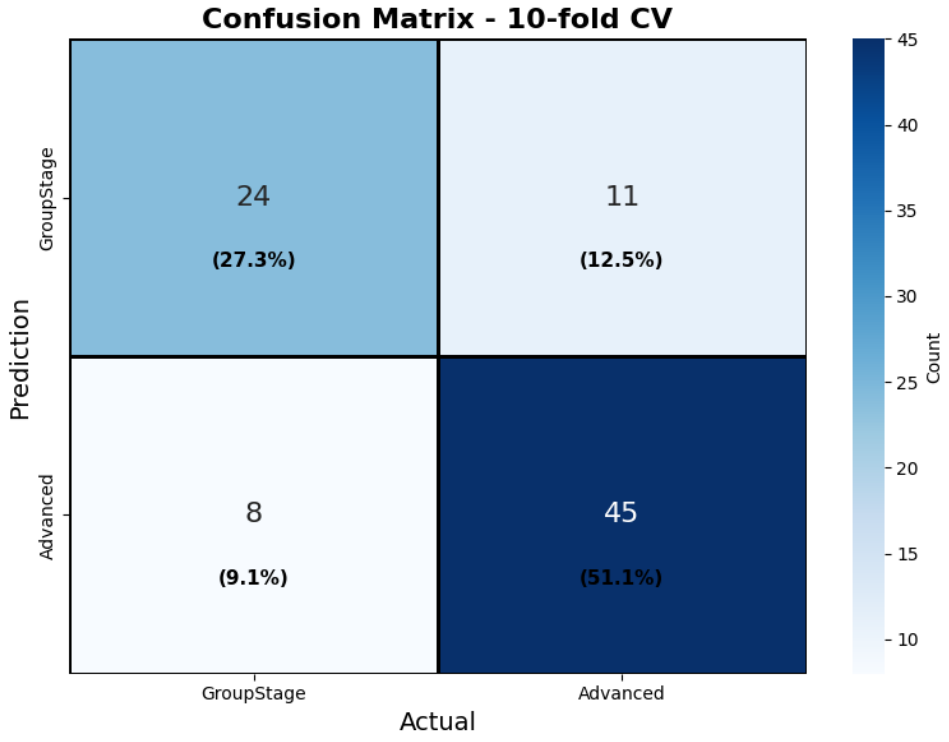

Figure 6.18: Confusion matrix of the Neural Network model for predicting qualification from the group stage (Multinomial Logistic Regression features)

From the confusion matrix we see that the model correctly predicted 24 teams that were eliminated (True Negatives) and 45 that advanced (True Positives), while it made 8 errors by predicting advancement for teams that were ultimately eliminated (False Positives) and 11 errors by predicting elimination for teams that advanced (False Negatives). Compared to the previous model, the errors decreased in both categories, and the balance between the two classes improved.

Metric	Value
Accuracy	0.784
Sensitivity/Recall	0.804
Specificity	0.750
Precision	0.849
F1-Score	0.826
Youden's Index	0.554
Cohen's Kappa (k)	0.542
Matthews Correlation Coefficient (MCC)	0.544
AUC	0.866

Table 6.6: Performance metrics for the Neural Network (Multinomial Logistic Regression features)

Overall accuracy reaches 78.4%, meaning that roughly 8 out of 10 predictions are correct – a fairly good result, clearly improved compared to previous models. Sensitivity (Recall) is 80.4%, so the model correctly identifies four out of five teams that advanced. Specificity stands at 75.0%, which means that three out of four teams eliminated in the group stage are correctly identified – a significant improvement over previous rates, indicating a much better balance between the two classes. When the model predicts advancement (Advanced), the prediction is correct in 84.9% of cases (Precision), indicating high reliability of positive predictions. The harmonic mean of sensitivity and precision (F1-score) reaches 82.6%, confirming an excellent balance between detecting advancements and the reliability of those predictions. Youden's Index (0.554) shows that the model has good overall discriminative power, while the Matthews correlation coefficient (MCC = 0.544) and Cohen's Kappa (0.543) suggest moderate to good agreement beyond chance, clearly outperforming a random classifier. The area under the ROC curve (AUC) reaches 0.866, which is considered very good discriminatory ability: if we randomly select a team that advanced and one that was eliminated, the model has approximately an 87% probability of assigning a higher score to the team that actually advanced. This represents a substantial improvement, making the model reliable for predictions on similar data. Overall, the model demonstrates very good performance, with high accuracy, a well-balanced trade-off between sensitivity and specificity, and excellent discriminatory ability as reflected by the AUC. The agreement metrics (Kappa, MCC) confirm that the model performs significantly better than chance, making it useful for practical applications in predicting advancement in football tournaments.

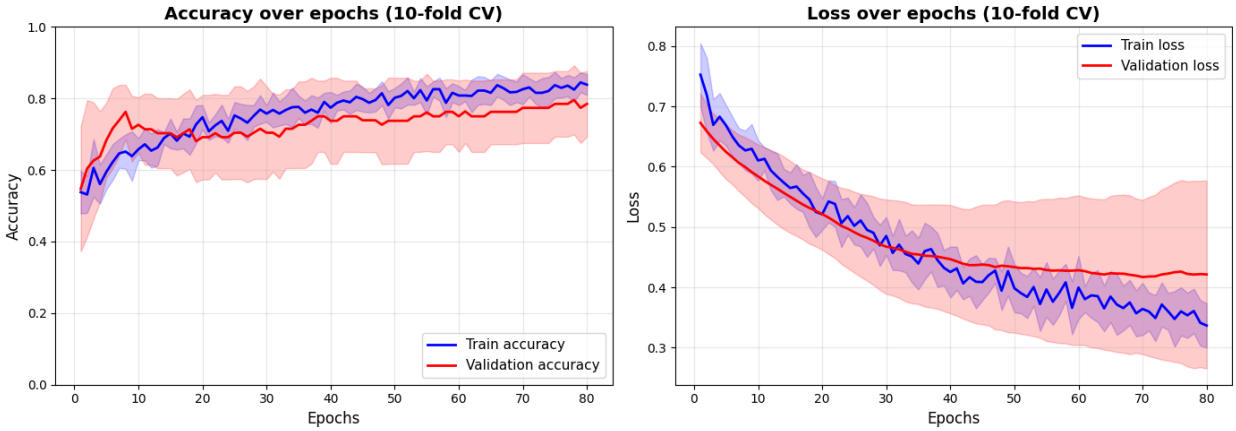

Figure 6.19: Neural network model Accuracy and Loss graphs

The image presents two graphs illustrating the evolution of accuracy and loss during the training of a machine learning model as a function of the number of epochs, using the 10-fold cross-validation method. In the left graph, the blue curve represents the model's accuracy on the training set (train accuracy), while the red curve shows the accuracy on the validation set (validation accuracy). We observe that accuracy increases gradually over the epochs, starting from approximately 0.50–0.55 and reaching close to 0.80–0.82. The two curves follow a similar trajectory and remain relatively close to each other, indicating that the model is learning effectively without strong signs of overfitting. The shaded areas around the curves represent the variability of the results across the different folds of the cross-validation. The right graph shows the evolution of the loss. The blue line represents the loss on the training set (train loss), while the red line represents the loss on the validation set (validation loss). We observe that the loss decreases steadily as the number of epochs increases, from values around 0.75–0.80 to approximately 0.38–0.42 for the training set, demonstrating that the model's performance improves over time. The validation loss follows a similar downward trend, although it remains slightly higher and appears to stabilize toward the end. Overall, the small gap between the training and validation curves in both accuracy and loss suggests that the model has good generalization ability and does not suffer from significant overfitting.

Comparing the two models, we observe a clear superiority of the model that was fed with the variables from the multinomial logistic model. The first model (Boruta variables) achieved moderate performance (Accuracy 71.6%, AUC 0.80, Sensitivity 75%, Specificity 65.6%), with a tendency for optimistic predictions and limited ability to distinguish eliminations. In contrast, the second model substantially improved all metrics (Accuracy 78.4%, AUC 0.86, Sensitivity 80.3%, Specificity 75%), significantly reducing false positives and false negatives, while achieving better balance and stronger discriminatory power (Youden's Index 0.55, MCC 0.54). The overall picture shows that feature selection via the multinomial logistic model leads to more reliable, generalizable, and balanced predictions regarding team qualification.

Comparison of model performance metrics						
Models	Accuracy	Sensitivity	Specificity	Precision	F1-Score	AUC
k-NN (Boruta)	73.9%	85.7%	53.1%	76.2%	80.7%	83%
k-NN (Multinomial)	70.5%	96.4%	25%	69.2%	80.6%	-
SVM (Boruta)	73.9%	78.6%	65.6%	80%	79.3%	85.6%
SVM (Multinomial)	85.2%	89.3%	78.1%	87.7%	88.5%	89.6%
MLP (Boruta)	72.7%	80.4%	59.4%	77.6%	78.9%	79.1%
MLP (Multinomial)	78.4%	80.4%	75%	84.9%	82.6%	86.6%

Table 6.7: Classification metrics of the k-NN, SVM, and MLP models

The table shows that the choice of independent variables significantly affects model performance. The set of variables derived from Multinomial Logistic Regression leads to substantially better results compared to those from the Boruta method, regardless of the algorithm used. Specifically, the SVM model using the Multinomial Logistic Regression variables achieves the highest overall performance: accuracy 85.2%, sensitivity 89.3%, specificity 78.1%, F1-score 88.5%, and AUC 89.6%. Next is the neural network (MLP) with the same variables, attaining an accuracy of 78.4% and specificity of 75%, demonstrating balanced behavior. In contrast, when the Boruta-selected variables are used, all three models underperform: k-NN (Boruta) and SVM (Boruta) achieve 73.9% accuracy, while MLP (Boruta) reaches only 72.7%. Notably, k-NN (Multinomial) shows very high sensitivity (96.4%) but extremely low specificity (25%), indicating a strong tendency to over-predict the positive class. In conclusion, the variables identified by Multinomial Logistic Regression are more informative for this problem, with SVM being the best performing classifier.

CHAPTER 7

7. Conclusions

This thesis attempted to study the main factors that influence the performance of national football teams in the European Championship, covering the tournaments of 2012, 2016, 2020 and 2024. The variables examined were mostly basic match statistics – shots, shots on target, attacks, passes, possession, goals conceded, saves, and cards. The aim was to present an analysis that blends common football sense with a rigorous statistical framework, highlighting not only the obvious drivers of success (goals, for instance) but also the deeper game-related variables that determine a team's ultimate achievement.

The data used in the analysis came from UEFA's official website, for the tournaments where complete statistics were available. Two levels of data were employed: a team-per-tournament aggregate level, and a per-match level where variables were expressed as averages per game. This made for a fairer comparison between teams that played a different number of matches – since sides that go further in the competition naturally record higher totals of actions. In addition, the study looked at two key classification problems: first, whether a team qualifies from the group stage or not, and second, whether a team belongs to a low, medium or high performance level based on the stage at which it exited the tournament.

As far as the descriptive results are concerned, higher-level teams stood out mainly in variables related to attacking output and quality of play. Specifically, teams that reached the later stages had higher values for goals, attacks, assists, shots, and shots on target; at the same time, they completed more passes and showed better passing accuracy. Greater possession and greater homogeneity among high-level teams also suggest that these sides do not rely on isolated attacking actions alone, but have better overall control of the game. In contrast, for variables such as crosses, fouls, and certain defensive actions, the differences between team categories were not always so pronounced. This indicates that simply producing a large volume of actions is not enough – it is the quality and efficiency of those actions that matter.

The evolution of characteristics over the tournaments also revealed interesting changes in the competitive profile of the European Championship. The number of shots and shots on target was found to have decreased compared to earlier tournaments, while attacks increased in the more recent editions. This finding suggests that teams may be creating more attacking phases, but that does not necessarily translate into more final attempts or greater efficiency. At the same time, passing accuracy seems to have improved relative to 2012, which may be linked to the evolution of modern football towards a more organised and technically demanding game. These observations support the view that football is not a static sport – its pace, tactics, and the way teams develop their play change from one tournament to the next.

Next, normality tests and correlation analysis confirmed that most quantitative variables do not follow a normal distribution, which led to the use of appropriate non-parametric methods where necessary. The correlation analysis provided useful information about which variables are most strongly linked to winning and to on-pitch success. As expected, variables directly related to scoring goals – such as goals, goals from inside the box, and assists – showed a strong positive correlation with wins. At the same time, the relationship with clean sheets was also significant, highlighting the importance of defensive solidity. Conversely, goals conceded and a high number of saves had a negative relationship with wins, suggesting that teams that are frequently on the back foot and face many scoring opportunities have a lower chance of success.

The results of the difference tests between teams of different levels were particularly important as well. These tests largely confirmed what the descriptive analysis had already shown. For goals and assists, a clear hierarchy emerged: high-level teams had the highest values, medium-level teams followed, and low-level teams had the lowest. A similar pattern appeared for shots, especially shots on target, where high-level teams clearly stood out from the rest. Correspondingly, for goals conceded, the hierarchy was reversed – high-level teams conceded fewer goals, medium-level conceded more, and low-level conceded the most. Overall, then, success at the EURO is linked to two main pillars: the ability to create and convert quality chances, and defensive stability.

Regarding the generalised linear models, binary logistic regression was used to predict whether a team would qualify from the group stage or not. A key point of this approach was that variables directly related to goals scored were deliberately excluded, so that the model would not simply conclude that teams scoring more goals go through. Instead, we used variables that describe the pathway to creating and preventing goals, such as shots on target, completed passes, blocked shots, attacks, goals conceded, and penalty saves. The final model achieved a very satisfactory accuracy, correctly classifying 86.4% of the cases – showing that these match indicators have strong predictive power for group-stage qualification.

In the same vein, multinomial logistic regression was used to classify teams into low, medium and high performance levels. The results showed that the selected variables were jointly statistically significant and that the model had strong explanatory power, as reflected in the McFadden's Pseudo R^2 value. The overall classification accuracy reached 77.3%, which is a respectable figure for a three-category problem. Interpreting the coefficients revealed that more shots on target and more attacks increase the probability of a team belonging to a higher performance category, while more goals conceded and more blocked shots decrease that probability. This result is particularly interesting because it shows that mere attacking activity is not enough when attempts do not hit the target or when the team cannot defend effectively.

As for machine learning methods, k-NN, SVM and neural networks were applied, using 10-fold cross-validation to evaluate the models. The analysis was run with two sets of variables: one selected by the Boruta method and one based on the variables from the multinomial logistic model. The results showed that these algorithms can be used with football data, but their effectiveness depends heavily on the choice of variables. k-NN, although it correctly identified quite a few teams that eventually qualified, tended to make over-optimistic predictions – that is, it predicted qualification for several teams that were ultimately eliminated. Neural networks performed better when using the

variables from the multinomial logistic model, achieving an accuracy of 79.6% and an AUC of 0.86, which indicates that they can work satisfactorily, though they do not come out as the best approach in the present analysis.

The most balanced and reliable machine learning model was the SVM using the variables derived from the multinomial logistic model. This model achieved an accuracy of 85.2%, high sensitivity, satisfactory specificity, and very good discriminatory power with an AUC of 0.896. Its main advantage was that it not only identified teams that go through, but could also recognise, to a satisfactory degree, teams that remain in the group stage. In other words, it largely avoided the over-optimistic prediction problem observed in other models. This suggests that combining statistical modelling with machine learning can be particularly useful – variables coming from an interpretable statistical model can feed a machine learning algorithm and lead to more reliable predictions.

Overall, this analysis leads to the conclusion that a team's success at the European Championship does not depend solely on the sheer quantity of attacking actions, but mainly on their efficiency and quality. Shots on target, attacking activity, accuracy in ball handling, and creating chances all matter, but they are not enough on their own. Defensive solidity – and especially avoiding goals – appears to be an equally decisive factor. Therefore, the teams that succeed in major tournaments are not necessarily those that attack the most, but those that combine effective attack, quality finishing, and strong defensive organisation.

Finally, it should be noted that this study is limited to those tournaments for which complete data were available, i.e. from 2012 to 2024. This means the sample size is relatively small, as is often the case in studies of national team football tournaments. Nevertheless, the results are quite consistent across different methodological approaches and offer a meaningful insight into the factors that influence qualification and performance at the EURO. Future research could incorporate more tournaments, data from other national or international competitions, and more complex variables such as expected goals (xG), tactical indicators, or contextual match factors. However, based on the data available for this thesis, the main conclusion remains clear: in modern football, it is not enough for a team to create – it must create with quality, execute efficiently, and defend consistently. In the end, at the European Championship, as in football more generally, success does not always belong to the team that does the most, but to the one that does the right things at the most critical moments.

THE END

References

Greek

Androulakis, E. (2024). Course notes "*Statistical Software 2*". Department of Statistics and Insurance Science, University of Piraeus. (Course material for the Undergraduate Program).

Chondrodima, E. (2024). Course notes "*Statistical Methods for Data Mining*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics

.

Chondrorrizos, V. (2024*). Statistical models for the performance analysis of teams in the FIFA World Cup*. https://dione.lib.unipi.gr/xmlui/handle/unipi/16873

Evangelaras, C. (2024). Course notes "*Data analysis using statistical packages: Notes for IBM SPSS Statistics*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics.

Iliopoulos, G. (2025). Course notes "*Generalized Linear Models*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics.

Kaligheris, E. (2025). Course notes "*Statistical Machine Learning*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics.

Kamitsis, A. (2023). *Indicators for the evaluation of player and team performance in basketball matches and the factors that affect them.* https://dione.lib.unipi.gr/xmlui/handle/unipi/15593

Karagrigoriou, A. (2025). Course notes "*Statistical Machine Learning*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics.

Koutras, M. (2024). Course notes "*Regression analysis and analysis of variance*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics.

Politis, K. (2025). Course notes "*Generalized Linear Models*". Department of Statistics and Insurance Science, University of Piraeus, Master's Program in Applied Statistics.

Spyridakis, A. E. (2022). *Statistical analysis of the factors affecting team performance in European football competitions*. https://dione.lib.unipi.gr/xmlui/handle/unipi/14180

Foreign

Abdi, H., & Williams, L. J. (2010). Tukey's honestly significant difference (HSD) test. In N. Salkind (Ed.), *Encyclopedia of research design* (pp. 1–5). Sage Publications.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 121–167. https://doi.org/10.1023/A:1009715923555

Daoud, J. I. (2017). Multicollinearity and regression analysis. *Journal of Physics: Conference Series*, 949, Article 012009. https://doi.org/10.1088/1742-6596/949/1/012009

Dinno, A. (2015). Nonparametric pairwise multiple comparisons in independent groups using Dunn's test. *The Stata Journal*, 15(1), 292–300. https://doi.org/10.1177/1536867X1501500117

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(90), 3133–3181.

Groll, A., Hvattum, L. M., Ley, C., & Schauberger, G. (2021). Hybrid machine learning forecasts for the UEFA EURO 2020. *AStA Advances in Statistical Analysis*, 105(3), 413–437. https://doi.org/10.1007/s10182-021-00405-5

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

Jadhav, S. D., & Channe, H. P. (2016). Comparative study of K-NN, Naive Bayes and decision tree classification techniques. *International Journal of Science and Research (IJSR)*, 5(1), 1842–1845. https://dx.doi.org/10.21275/NOV153131

Lapre, M. A., & Amato, J. G. (2025). The impact of imbalanced groups in UEFA Euro 1980–2024 and comparison with the FIFA World Cup. *Journal of Sports Analytics*, 11(1), 45–61. https://doi.org/10.3233/JSA-240789

Maneiro, R., Casal, C. A., Losada, J. L., & Ardá, T. (2019). Offensive transitions in high-performance football: Differences between UEFA EURO 2008 and UEFA EURO 2016. *International Journal of Performance Analysis in Sport*, 19(3), 411–428. https://doi.org/10.1080/24748668.2019.1618541

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14, Article 6086. https://doi.org/10.1038/s41598-024-56706-x

Renner, V., Gorgen, K., & Schmidt, M. (2024). Success factors in national team football: An analysis of the UEFA EURO 2020. *Frontiers in Sports and Active Living*, 6, Article 1289452. https://doi.org/10.3389/fspor.2024.1289452

Rosidi, N. (2023). Feature selection techniques in machine learning. *StrataScratch*. Retrieved from https://www.stratascratch.com/blog/feature-selection-techniques-in-machine-learning/

Smith, T. J., & McKenna, C. M. (2013). A comparison of logistic regression pseudo R^2 indices. *Multiple Linear Regression Viewpoints*, 39(2), 17–26.

Szul, T., Tabor, S., & Pancerz, K. (2021). Application of the BORUTA algorithm to input data selection for a model based on rough set theory (RST) to prediction energy consumption for building heating. *Energies*, 14(10), 2779. https://doi.org/10.3390/en14102779

Vapnik, V., Golowich, S. E., & Smola, A. J. (1997). Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems 9 (NIPS 1996)* (pp. 281–287). MIT Press.

Appendix

Code in R

CHAPTER 3

```
library(readxl)
library(dplyr)
library(ggplot2)
library(tidyr)

# Load data
data <- read_excel("UEFA_EURO_12-24.xlsx")

# 1. Create Phase variable with new categories
data_analysis <- data %>%
  mutate(
    Phase = case_when(
      `Final Rank` == "Group Stage" ~ "Low",
      `Final Rank` %in% c("Round of 16", "Quarter-Finals") ~ "Medium",
      `Final Rank` %in% c("Semi-Finals", "Finalist", "Winner") ~ "High",
      TRUE ~ NA_character_
    )
  ) %>%
  filter(!is.na(Phase))

# 2. Order phases for correct display
phase_order <- c("Low", "Medium", "High")
data_analysis$Phase <- factor(data_analysis$Phase, levels = phase_order)

# 3. List of all variables for boxplots
all_variables <- c("Goals_per_match", "Total Attempts_per_match",
                   "Attempts on Target_per_match", "Attempts off Target_per_match",
                   "Passes Attempted_per_match", "Passes Completed_per_match",
                   "Crosses Attempted_per_match", "Crosses Completed_per_match",
                   "Attacks_per_match", "Free-kicks Taken_per_match",
                   "Tackles_per_match", "Fouls Commited_per_match",
                   "Fouls Suffered_per_match", "Yellow Cards_per_match",
                   "Corners Taken_per_match", "Assists_per_match",
                   "Offsides_per_match", "Saves_per_match")

# 4. Reshape for multiple boxplots
boxplot_data <- data_analysis %>%
  select(Phase, all_of(all_variables)) %>%
  pivot_longer(-Phase, names_to = "Variable", values_to = "Value")

# 5. Improved variable names for the plot
variable_labels <- c(
  "Goals_per_match" = "Goals per match",
```

```
  "Total Attempts_per_match" = "Total Attempts per match",
  "Attempts on Target_per_match" = "Attempts on Target per match",
  "Attempts off Target_per_match" = "Attempts off Target per match",
  "Passes Attempted_per_match" = "Passes Attempted per match",
  "Passes Completed_per_match" = "Passes Completed per match",
  "Crosses Attempted_per_match" = "Crosses Attempted per match",
  "Crosses Completed_per_match" = "Crosses Completed per match",
  "Attacks_per_match" = "Attacks per match",
  "Free-kicks Taken_per_match" = "Free-kicks Taken per match",
  "Tackles_per_match" = "Tackles per match",
  "Fouls Commited_per_match" = "Fouls Commited per match",
  "Fouls Suffered_per_match" = "Fouls Suffered per match",
  "Yellow Cards_per_match" = "Yellow Cards per match",
  "Corners Taken_per_match" = "Corners Taken per match",
  "Assists_per_match" = "Assists per match",
  "Offsides_per_match" = "Offsides per match",
  "Saves_per_match" = "Saves per match"
)

boxplot_data <- boxplot_data %>%
  mutate(Variable_Label = factor(Variable,
                                 levels = all_variables,
                                 labels = variable_labels))

# 6. Colors for each phase category
phase_colors <- c(
  "Low" = "#3498DB",      # Blue for Low
  "Medium" = "#E74C3C",   # Red for Medium
  "High" = "#9B59B6"      # Purple for High
)

# 7. BOXPLOT 1: Offensive Variables
offensive_vars <- c("Goals_per_match", "Total Attempts_per_match",
                    "Attempts on Target_per_match", "Attempts off Target_per_match",
                    "Attacks_per_match", "Assists_per_match")

p1 <- ggplot(boxplot_data %>% filter(Variable %in% offensive_vars),
             aes(x = Phase, y = Value, fill = Phase)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 1) +
  facet_wrap(~ Variable_Label, scales = "free_y", ncol = 3) +
  labs(title = "Offensive Variables by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Value per Match", x = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom") +
  scale_fill_manual(values = phase_colors)
```

```
print(p1)

# 8. BOXPLOT 2: Passing Variables
playing_style_vars <- c("Passes Attempted_per_match", "Passes
Completed_per_match",
                        "Crosses Attempted_per_match", "Crosses Completed_per_match")

p2 <- ggplot(boxplot_data %>% filter(Variable %in% playing_style_vars),
             aes(x = Phase, y = Value, fill = Phase)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 1) +
  facet_wrap(~ Variable_Label, scales = "free_y", ncol = 2) +
  labs(title = "Playing Style by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Value per match", x = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom") +
  scale_fill_manual(values = phase_colors)

print(p2)

# 9. BOXPLOT 3: Defensive Variables
defensive_vars <- c("Tackles_per_match", "Fouls Commited_per_match",
                    "Fouls Suffered_per_match", "Yellow Cards_per_match")

p3 <- ggplot(boxplot_data %>% filter(Variable %in% defensive_vars),
             aes(x = Phase, y = Value, fill = Phase)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 1) +
  facet_wrap(~ Variable_Label, scales = "free_y", ncol = 2) +
  labs(title = "Defensive Variables by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Value per match", x = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom") +
  scale_fill_manual(values = phase_colors)

print(p3)

# 10. BOXPLOT 4: Other Variables
others_var <- c("Free-kicks Taken_per_match", "Corners Taken_per_match",
                "Offsides_per_match", "Saves_per_match")

p4 <- ggplot(boxplot_data %>% filter(Variable %in% others_var),
             aes(x = Phase, y = Value, fill = Phase)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
```

```
  geom_jitter(width = 0.2, alpha = 0.5, size = 1) +
  facet_wrap(~ Variable_Label, scales = "free_y", ncol = 2) +
  labs(title = "Other Variables by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Value per match", x = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom") +
  scale_fill_manual(values = phase_colors)

print(p4)

# 11. Comparison table by phase category
detailed_comparison <- data_analysis %>%
  group_by(Phase) %>%
  summarise(
    Teams = n(),
    # Offensive variables
    Goals = round(mean(Goals_per_match, na.rm = TRUE), 2),
    Total_Attempts = round(mean(`Total Attempts_per_match`, na.rm = TRUE), 1),
    Attempts_On_Target = round(mean(`Attempts on Target_per_match`, na.rm =
TRUE), 1),
    Attempts_Off_Target = round(mean(`Attempts off Target_per_match`, na.rm =
TRUE), 1),
    Attacks = round(mean(Attacks_per_match, na.rm = TRUE), 1),
    Assists = round(mean(Assists_per_match, na.rm = TRUE), 1),

    # Passing variables
    Passes_Attempted = round(mean(`Passes Attempted_per_match`, na.rm = TRUE),
1),
    Passes_Completed = round(mean(`Passes Completed_per_match`, na.rm = TRUE),
1),
    Crosses_Attempted = round(mean(`Crosses Attempted_per_match`, na.rm =
TRUE), 1),
    Crosses_Completed = round(mean(`Crosses Completed_per_match`, na.rm =
TRUE), 1),

    # Defensive variables
    Tackles = round(mean(Tackles_per_match, na.rm = TRUE), 1),
    Fouls_Commited = round(mean(`Fouls Commited_per_match`, na.rm = TRUE),
1),
    Fouls_Suffered = round(mean(`Fouls Suffered_per_match`, na.rm = TRUE), 1),
    Yellow_Cards = round(mean(`Yellow Cards_per_match`, na.rm = TRUE), 1),

    # Other variables
    Free_kicks_Taken = round(mean(`Free-kicks Taken_per_match`, na.rm = TRUE),
1),
    Corners_Taken = round(mean(`Corners Taken_per_match`, na.rm = TRUE), 1),
    Offsides = round(mean(Offsides_per_match, na.rm = TRUE), 1),
```

```
    Saves = round(mean(Saves_per_match, na.rm = TRUE), 1),

    # Percentage variables
    Passing_Accuracy = round(mean(`Passing Accuracy (%)`, na.rm = TRUE), 1),
    Crossing_Accuracy = round(mean(`Crossing Accuracy (%)`, na.rm = TRUE), 1),
    Possession = round(mean(`Possession (%)`, na.rm = TRUE), 1)
  )

# Display table
print(detailed_comparison)

# 12. Density plots for percentage variables
p_passing <- ggplot(data_analysis, aes(x = `Passing Accuracy (%)`, fill = Phase)) +
  geom_density(alpha = 0.6, adjust = 1.5) +
  labs(title = "Passing Accuracy Distribution by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       x = "Passing Accuracy (%)", y = "Density") +
  theme_minimal() +
  scale_fill_manual(values = phase_colors) +
  theme(legend.position = "top")

p_crossing <- ggplot(data_analysis, aes(x = `Crossing Accuracy (%)`, fill = Phase)) +
  geom_density(alpha = 0.6, adjust = 1.5) +
  labs(title = "Crossing Accuracy Distribution by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       x = "Crossing Accuracy (%)", y = "Density") +
  theme_minimal() +
  scale_fill_manual(values = phase_colors) +
  theme(legend.position = "top")

p_possession <- ggplot(data_analysis, aes(x = `Possession (%)`, fill = Phase)) +
  geom_density(alpha = 0.6, adjust = 1.5) +
  labs(title = "Possession Distribution by Performance Level",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       x = "Possession (%)", y = "Density") +
  theme_minimal() +
  scale_fill_manual(values = phase_colors) +
  theme(legend.position = "top")

print(p_passing)
print(p_crossing)
print(p_possession)
```

```
library(readxl)
library(dplyr)
library(ggplot2)
```

```
library(tidyr)
library(scales)

# Load data
data <- read_excel("UEFA_EURO_12-24.xlsx")

# Colors for each goal type
goal_type_colors <- c(
  "Right_Foot" = "#E74C3C",    # Red for right foot
  "Left_Foot" = "#3498DB",     # Blue for left foot
  "Head" = "#2ECC71",          # Green for head
  "Other" = "#9B59B6"          # Purple for other
)

# Calculate goal percentage by type per year
goals_by_type <- data %>%
  group_by(Year) %>%
  summarise(
    Total_Goals = sum(Goals, na.rm = TRUE),
    Right_Foot_Goals = sum(`Right foot`, na.rm = TRUE),
    Left_Foot_Goals = sum(`Left foot`, na.rm = TRUE),
    Head_Goals = sum(Head, na.rm = TRUE),
    Other_Goals = sum(Other, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  mutate(
    Right_Foot_Pct = (Right_Foot_Goals / Total_Goals) * 100,
    Left_Foot_Pct = (Left_Foot_Goals / Total_Goals) * 100,
    Head_Pct = (Head_Goals / Total_Goals) * 100,
    Other_Pct = (Other_Goals / Total_Goals) * 100
  ) %>%
  select(Year, Right_Foot_Pct, Left_Foot_Pct, Head_Pct, Other_Pct) %>%
  pivot_longer(
    cols = c(Right_Foot_Pct, Left_Foot_Pct, Head_Pct, Other_Pct),
    names_to = "Goal_Type",
    values_to = "Percentage"
  ) %>%
  mutate(
    Goal_Type = case_when(
      Goal_Type == "Right_Foot_Pct" ~ "Right_Foot",
      Goal_Type == "Left_Foot_Pct" ~ "Left_Foot",
      Goal_Type == "Head_Pct" ~ "Head",
      Goal_Type == "Other_Pct" ~ "Other",
      TRUE ~ Goal_Type
    )
  )

# Grouped (side-by-side) barplot with value labels
ggplot(goals_by_type, aes(x = factor(Year), y = Percentage, fill = Goal_Type)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.8) +
```

```
  scale_fill_manual(
    values = goal_type_colors,
    labels = c("Right_Foot" = "Right Foot",
               "Left_Foot" = "Left Foot",
               "Head" = "Head",
               "Other" = "Other")
  ) +
  labs(
    title = "Goal Distribution by Type Across EURO Tournaments (2012-2024)",
    subtitle = "Percentage of Total Goals by Scoring Method",
    y = "Percentage of Total Goals (%)",
    x = "Tournament Year",
    fill = "Goal Type"
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 11),
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 12, hjust = 0.5)
  ) +
  # Add value labels on bars
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_dodge(width = 0.7),
            vjust = -0.5, size = 4, fontface = "bold")

# Colors for inside/outside area
area_colors <- c(
  "Inside_Area" = "#E74C3C",    # Red for inside area
  "Outside_Area" = "#3498DB"    # Blue for outside area
)

# Calculate goal percentage by area per year
goals_by_area <- data %>%
  group_by(Year) %>%
  summarise(
    Total_Goals = sum(Goals, na.rm = TRUE),
    Inside_Area_Goals = sum(`Goals inside area`, na.rm = TRUE),
    Outside_Area_Goals = sum(`Goals outside area`, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  mutate(
    Inside_Area_Pct = (Inside_Area_Goals / Total_Goals) * 100,
    Outside_Area_Pct = (Outside_Area_Goals / Total_Goals) * 100
  ) %>%
  select(Year, Inside_Area_Pct, Outside_Area_Pct) %>%
  pivot_longer(
    cols = c(Inside_Area_Pct, Outside_Area_Pct),
    names_to = "Goal_Area",
```

```
    values_to = "Percentage"
  ) %>%
  mutate(
    Goal_Area = case_when(
      Goal_Area == "Inside_Area_Pct" ~ "Inside_Area",
      Goal_Area == "Outside_Area_Pct" ~ "Outside_Area",
      TRUE ~ Goal_Area
    )
  )

# Grouped barplot for goals inside/outside area
ggplot(goals_by_area, aes(x = factor(Year), y = Percentage, fill = Goal_Area)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.8, width = 0.7) +
  scale_fill_manual(
    values = area_colors,
    labels = c("Inside_Area" = "Inside Area",
               "Outside_Area" = "Outside Area")
  ) +
  labs(
    title = "Goal Distribution by Area Across EURO Tournaments (2012-2024)",
    subtitle = "Percentage of Total Goals by Scoring Area",
    y = "Percentage of Total Goals (%)",
    x = "Tournament Year",
    fill = "Goal Area"
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 11),
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 12, hjust = 0.5),
    legend.text = element_text(size = 11)
  ) +
  # Add value labels on bars
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_dodge(width = 0.7),
            vjust = -0.5, size = 4, fontface = "bold")

# Colors for attempt types
attempts_colors <- c(
  "On_Target" = "#2ECC71",    # Green for on target
  "Off_Target" = "#E74C3C",   # Red for off target
  "Blocked" = "#3498DB"       # Blue for blocked
)

# Calculate attempt percentage per year
attempts_analysis <- data %>%
  group_by(Year) %>%
  summarise(
```

```
  Total_Attempts = sum(`Total Attempts`, na.rm = TRUE),
  On_Target = sum(`Attempts on Target`, na.rm = TRUE),
  Off_Target = sum(`Attempts off Target`, na.rm = TRUE),
  Blocked = sum(Blocked, na.rm = TRUE),
  .groups = 'drop'
 ) %>%
 mutate(
  On_Target_Pct = (On_Target / Total_Attempts) * 100,
  Off_Target_Pct = (Off_Target / Total_Attempts) * 100,
  Blocked_Pct = (Blocked / Total_Attempts) * 100
 ) %>%
 select(Year, On_Target_Pct, Off_Target_Pct, Blocked_Pct) %>%
 pivot_longer(
  cols = c(On_Target_Pct, Off_Target_Pct, Blocked_Pct),
  names_to = "Attempt_Type",
  values_to = "Percentage"
 ) %>%
 mutate(
  Attempt_Type = case_when(
   Attempt_Type == "On_Target_Pct" ~ "On_Target",
   Attempt_Type == "Off_Target_Pct" ~ "Off_Target",
   Attempt_Type == "Blocked_Pct" ~ "Blocked",
   TRUE ~ Attempt_Type
  )
 )

# Grouped barplot for attempts
ggplot(attempts_analysis, aes(x = factor(Year), y = Percentage, fill = Attempt_Type))
+
 geom_bar(stat = "identity", position = "dodge", alpha = 0.8, width = 0.7) +
 scale_fill_manual(
  values = attempts_colors,
  labels = c("On_Target" = "On Target",
             "Off_Target" = "Off Target",
             "Blocked" = "Blocked")
 ) +
 labs(
  title = "Shot Distribution by Type Across EURO Tournaments (2012-2024)",
  subtitle = "Percentage of Total Attempts by Shot Type",
  y = "Percentage of Total Attempts (%)",
  x = "Tournament Year",
  fill = "Shot Type"
 ) +
 theme_minimal() +
 theme(
  legend.position = "bottom",
  axis.text.x = element_text(size = 12),
  axis.text.y = element_text(size = 11),
  plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
  plot.subtitle = element_text(size = 12, hjust = 0.5),
```

```
    legend.text = element_text(size = 11)
  ) +
  # Add value labels on bars
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_dodge(width = 0.7),
            vjust = -0.5, size = 4, fontface = "bold")

# Colors for tackle types
tackles_colors <- c(
  "Won" = "#2ECC71",    # Green for won
  "Lost" = "#E74C3C"    # Red for lost
)

# Calculate tackle percentage per year
tackles_analysis <- data %>%
  group_by(Year) %>%
  summarise(
    Total_Tackles = sum(Tackles, na.rm = TRUE),
    Won = sum(`Tackles Won`, na.rm = TRUE),
    Lost = sum(`Tackles Lost`, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  mutate(
    Won_Pct = (Won / Total_Tackles) * 100,
    Lost_Pct = (Lost / Total_Tackles) * 100
  ) %>%
  select(Year, Won_Pct, Lost_Pct) %>%
  pivot_longer(
    cols = c(Won_Pct, Lost_Pct),
    names_to = "Tackle_Type",
    values_to = "Percentage"
  ) %>%
  mutate(
    Tackle_Type = case_when(
      Tackle_Type == "Won_Pct" ~ "Won",
      Tackle_Type == "Lost_Pct" ~ "Lost",
      TRUE ~ Tackle_Type
    )
  )

# Grouped barplot for tackles
ggplot(tackles_analysis, aes(x = factor(Year), y = Percentage, fill = Tackle_Type)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.8, width = 0.7) +
  scale_fill_manual(
    values = tackles_colors,
    labels = c("Won" = "Won",
               "Lost" = "Lost")
  ) +
  labs(
    title = "Tackle Success Rate Across EURO Tournaments (2012-2024)",
```

```
    subtitle = "Percentage of Successful vs Unsuccessful Tackles",
    y = "Percentage of Total Tackles (%)",
    x = "Tournament Year",
    fill = "Tackle Result"
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 11),
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 12, hjust = 0.5),
    legend.text = element_text(size = 11)
  ) +
  # Add value labels on bars
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_dodge(width = 0.7),
            vjust = -0.5, size = 4, fontface = "bold")
```

```
library(readxl)
library(dplyr)
library(ggplot2)
library(tidyr)

# Load data
data <- read_excel("UEFA_EURO_12-24.xlsx")

# Create Phase categories (Low, Medium, High)
euro_analysis <- data %>%
  mutate(
    Phase = case_when(
      `Final Rank` == "Group Stage" ~ "Low",
      `Final Rank` %in% c("Round of 16", "Quarter-Finals") ~ "Medium",
      `Final Rank` %in% c("Semi-Finals", "Finalist", "Winner") ~ "High",
      TRUE ~ NA_character_
    )
  ) %>%
  filter(!is.na(Phase))

# Order phases for correct display
phase_order <- c("Low", "Medium", "High")
euro_analysis$Phase <- factor(euro_analysis$Phase, levels = phase_order)

# Colors for the new phase categories
phase_colors <- c(
  "Low" = "#3498DB",      # Blue for Low
  "Medium" = "#E74C3C",   # Red for Medium
  "High" = "#9B59B6"      # Purple for High
)
```

```
# 1. Evolution of offensive variables
trend_comparison <- euro_analysis %>%
  group_by(Year, Phase) %>%
  summarise(
    Goals = mean(Goals_per_match, na.rm = TRUE),
    Shots = mean(`Total Attempts_per_match`, na.rm = TRUE),
    Shots_On_Target = mean(`Attempts on Target_per_match`, na.rm = TRUE),
    Shots_Off_Target = mean(`Attempts off Target_per_match`, na.rm = TRUE),
    Attacks = mean(Attacks_per_match, na.rm = TRUE),
    Assists = mean(Assists_per_match, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  pivot_longer(cols = c(Goals, Shots, Shots_On_Target, Shots_Off_Target, Attacks,
Assists),
               names_to = "Metric", values_to = "Value")

ggplot(trend_comparison, aes(x = Year, y = Value, color = Phase)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 2) +
  labs(title = "Evolution of Offensive Variables by Team Performance Level (2012-
2024)",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Mean", x = "Year") +
  theme_minimal() +
  scale_color_manual(values = phase_colors) +
  theme(legend.position = "bottom")

# 2. Evolution of passing variables
trend_comparison2 <- euro_analysis %>%
  group_by(Year, Phase) %>%
  summarise(
    Passes_Attempted = mean(`Passes Attempted_per_match`, na.rm = TRUE),
    Passes_Completed = mean(`Passes Completed_per_match`, na.rm = TRUE),
    Crosses_Attempted = mean(`Crosses Attempted_per_match`, na.rm = TRUE),
    Crosses_Completed = mean(`Crosses Completed_per_match`, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  pivot_longer(cols = c(Passes_Attempted, Passes_Completed, Crosses_Attempted,
Crosses_Completed),
               names_to = "Metric", values_to = "Value")

ggplot(trend_comparison2, aes(x = Year, y = Value, color = Phase)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 2) +
  labs(title = "Evolution of Passing Game Variables by Team Performance Level
(2012-2024)",
```

```
      subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
      y = "Mean", x = "Year") +
  theme_minimal() +
  scale_color_manual(values = phase_colors) +
  theme(legend.position = "bottom")

# 3. Evolution of defensive variables
trend_comparison3 <- euro_analysis %>%
  group_by(Year, Phase) %>%
  summarise(
    Fouls_Commited = mean(`Fouls Commited_per_match`, na.rm = TRUE),
    Fouls_Suffered = mean(`Fouls Suffered_per_match`, na.rm = TRUE),
    Yellow_Cards = mean(`Yellow Cards_per_match`, na.rm = TRUE),
    Tackles = mean(Tackles_per_match, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  pivot_longer(cols = c(Fouls_Commited, Fouls_Suffered, Yellow_Cards, Tackles),
               names_to = "Metric", values_to = "Value")

ggplot(trend_comparison3, aes(x = Year, y = Value, color = Phase)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 2) +
  labs(title = "Evolution of Defensive Variables by Team Performance Level (2012-
2024)",
      subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
      y = "Mean", x = "Year") +
  theme_minimal() +
  scale_color_manual(values = phase_colors) +
  theme(legend.position = "bottom")

# 4. Evolution of other variables
trend_comparison4 <- euro_analysis %>%
  group_by(Year, Phase) %>%
  summarise(
    Free_kicks_Taken = mean(`Free-kicks Taken_per_match`, na.rm = TRUE),
    Corners_Taken = mean(`Corners Taken_per_match`, na.rm = TRUE),
    Offsides = mean(Offsides_per_match, na.rm = TRUE),
    Saves = mean(Saves_per_match, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  pivot_longer(cols = c(Free_kicks_Taken, Corners_Taken, Offsides, Saves),
               names_to = "Metric", values_to = "Value")

ggplot(trend_comparison4, aes(x = Year, y = Value, color = Phase)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 2) +
```

```
  labs(title = "Evolution of Other Variables by Team Performance Level (2012-
2024)",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Mean", x = "Year") +
  theme_minimal() +
  scale_color_manual(values = phase_colors) +
  theme(legend.position = "bottom")

# 5. Evolution of percentage variables
trend_comparison5 <- euro_analysis %>%
  group_by(Year, Phase) %>%
  summarise(
    Passing_Accuracy = mean(`Passing Accuracy (%)`, na.rm = TRUE),
    Crossing_Accuracy = mean(`Crossing Accuracy (%)`, na.rm = TRUE),
    Possession = mean(`Possession (%)`, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  pivot_longer(cols = c(Passing_Accuracy, Crossing_Accuracy, Possession),
               names_to = "Metric", values_to = "Value")

ggplot(trend_comparison5, aes(x = Year, y = Value, color = Phase)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 2) +
  labs(title = "Evolution of Percentage Variables by Team Performance Level (2012-
2024)",
       subtitle = "Low (Group Stage) | Medium (Round of 16 + Quarter-Finals) | High
(Semi-Finals + Finalist + Winner)",
       y = "Mean (%)", x = "Year") +
  theme_minimal() +
  scale_color_manual(values = phase_colors) +
  theme(legend.position = "bottom")
```

```
# Load required libraries
library(ggplot2)
library(dplyr)
library(readxl)

# Read the Excel file
data <- read_excel("UEFA_EURO_12-24.xlsx")

# Create scatter plots with a single color for all observations

# 1. Passes Completed vs Goals per match
ggplot(data, aes(x = `Passes Completed_per_match`, y = `Goals_per_match`)) +
  geom_point(size = 3, alpha = 0.7, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Passes Completed vs Goals",
```

```
      x = "Passes Completed per match",
      y = "Goals per match") +
  theme_minimal()

# 2. Attempts on Target vs Goals per match
ggplot(data, aes(x = `Attempts on Target_per_match`, y = `Goals_per_match`)) +
  geom_point(size = 3, alpha = 0.7, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Attempts on Target vs Goals",
      x = "Attempts on Target per match",
      y = "Goals per match") +
  theme_minimal()

# 3. Possession (%) vs Passing Accuracy (%)
ggplot(data, aes(x = `Possession (%)`, y = `Passing Accuracy (%)`)) +
  geom_point(size = 3, alpha = 0.7, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Possession (%) vs Passing Accuracy (%)",
      x = "Possession (%)",
      y = "Passing Accuracy (%)") +
  theme_minimal()

# 4. Attacks per match vs Total Attempts per match
ggplot(data, aes(x = `Attacks_per_match`, y = `Total Attempts_per_match`)) +
  geom_point(size = 3, alpha = 0.7, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Attacks vs Total Attempts",
      x = "Attacks per match",
      y = "Total Attempts per match") +
  theme_minimal()

# 6. Assists per match vs Goals per match
ggplot(data, aes(x = `Assists_per_match`, y = `Goals_per_match`)) +
  geom_point(size = 3, alpha = 0.7, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Assists vs Goals",
      x = "Assists per match",
      y = "Goals per match") +
  theme_minimal()

# 7. Corners Taken per match vs Total Attempts per match
ggplot(data, aes(x = `Corners Taken_per_match`, y = `Total Attempts_per_match`)) +
  geom_point(size = 3, alpha = 0.7, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Corners Taken vs Total Attempts",
      x = "Corners Taken per match",
      y = "Total Attempts per match") +
  theme_minimal()
```

```
# Load required packages
library(readxl)
library(dplyr)
library(nortest)

# Read the Excel file
euro_data <- read_excel("UEFA_EURO_12-24.xlsx")

# --------------------------------
# 1. Lilliefors normality test
#    on per-match and (%) variables
# --------------------------------
numeric_data <- euro_data %>%
  select(-Teams, -`Final Rank`, -Year)

selected_vars <- numeric_data %>%
  select(matches("_per_match$|\\(%\\)"))

selected_var_names <- names(selected_vars)

normality_results <- data.frame(
  Variable = character(),
  Lilliefors_Statistic = numeric(),
  Lilliefors_p_value = numeric(),
  Normal_Distribution = character(),
  stringsAsFactors = FALSE
)

for (var in selected_var_names) {
  clean_data <- na.omit(selected_vars[[var]])
  if (length(clean_data) >= 4) {
    lillie_test <- lillie.test(clean_data)
    normality_results <- rbind(normality_results, data.frame(
      Variable = var,
      Lilliefors_Statistic = lillie_test$statistic,
      Lilliefors_p_value = lillie_test$p.value,
      Normal_Distribution = ifelse(lillie_test$p.value > 0.05, "YES", "NO"),
      stringsAsFactors = FALSE
    ))
  }
}

normality_results <- normality_results %>% arrange(Lilliefors_p_value)
print("Normality test results (per-match and % variables):")
print(normality_results)

# --------------------------------
# 2. Poisson goodness-of-fit for
#    Goals_per_match (simulated p-value)
# --------------------------------
```

```
goals_data <- euro_data$Goals_per_match

# Estimate lambda
lambda_est <- mean(goals_data)

# Observed frequencies for all goal counts 0..max
max_goals <- max(goals_data)
goal_values <- 0:max_goals

observed <- sapply(goal_values, function(x) sum(goals_data == x))
names(observed) <- goal_values

# Expected probabilities (Poisson)
expected_probs <- dpois(goal_values, lambda = lambda_est)
# Normalize (in case of truncation)
expected_probs <- expected_probs / sum(expected_probs)
set.seed(123)
# Chi-squared test with simulated p-value (Monte Carlo)
chi_test <- chisq.test(observed, p = expected_probs,
                       simulate.p.value = TRUE, B = 10000)

print(chi_test)
```

CHAPTER 4

```
# Load required libraries
library(readxl)
library(dplyr)
library(ggplot2)
library(reshape2)
library(Hmisc)

# Read the Excel file
df <- read_excel("UEFA_EURO_12-24.xlsx")

# Identify all per_match columns
all_per_match_cols <- grep("per_match", colnames(df), value = TRUE, ignore.case =
TRUE)

# Exclude variables already analyzed elsewhere
exclude_vars <- c(
  "Fouls Suffered_per_match",
  "Free-kicks Taken_per_match",
  "Corners Taken_per_match",
  "Attacks_per_match",
  "Passes Attempted_per_match",
  "Passes Completed_per_match",
  "Yellow Cards_per_match",
  "Crosses Completed_per_match",
  "Tackles_per_match",
```

```
  "Tackles Lost_per_match",
  "Right foot_per_match",
  "Left foot_per_match",
  "Head_per_match",
  "Other_per_match"
)

remaining_per_match_cols <- setdiff(all_per_match_cols, exclude_vars)
df_remaining <- df %>% select(all_of(remaining_per_match_cols))

# Spearman correlation with p-values
cor_result <- rcorr(as.matrix(df_remaining), type = "spearman")
correlation_matrix <- cor_result$r
p_value_matrix <- cor_result$P

# Clean variable names for plot labels
clean_names <- function(name) {
  name <- gsub("_per_match", "", name)
  name <- gsub("_", " ", name)
  name <- gsub("Total Attempts", "Total Attempts", name)
  name <- gsub("Attempts on Target", "Attempts On Target", name)
  name <- gsub("Attempts off Target", "Attempts Off Target", name)
  name <- gsub("Free kicks Taken", "Free Kicks", name)
  name <- gsub("Corners Taken", "Corners", name)
  name <- gsub("Goals Conceded", "Goals Conceded", name)
  name <- gsub("Clean Sheets", "Clean Sheets", name)
  name <- gsub("Fouls Commited", "Fouls Committed", name)
  name <- gsub("Fouls Suffered", "Fouls Suffered", name)
  name <- gsub("Yellow Cards", "Yellow Cards", name)
  name <- gsub("Red Cards", "Red Cards", name)
  name <- gsub("Own Goals Conceded", "Own Goals", name)
  name <- gsub("Passing Accuracy", "Passing Accuracy %", name)
  name <- gsub("Crossing Accuracy", "Crossing Accuracy %", name)
  name <- gsub("Possession", "Possession %", name)

  # Specific per-match variables
  name <- gsub("Goals per match", "Goals", name)
  name <- gsub("Goals inside area per match", "Goals Inside Area", name)
  name <- gsub("Goals outside area per match", "Goals Outside Area", name)
  name <- gsub("Penalties scored per match", "Penalties Scored", name)
  name <- gsub("Right foot per match", "Right Foot Goals", name)
  name <- gsub("Left foot per match", "Left Foot Goals", name)
  name <- gsub("Head per match", "Head Goals", name)
  name <- gsub("Other per match", "Other Goals", name)
  name <- gsub("Blocked per match", "Blocked Shots", name)
  name <- gsub("Saves per match", "Saves", name)
  name <- gsub("Saves from Penalties per match", "Penalty Saves", name)

  return(name)
}
```

```
clean_colnames <- sapply(colnames(df_remaining), clean_names)
colnames(correlation_matrix) <- clean_colnames
rownames(correlation_matrix) <- clean_colnames
colnames(p_value_matrix) <- clean_colnames
rownames(p_value_matrix) <- clean_colnames

# Melt matrices for ggplot
cor_melted <- melt(correlation_matrix)
p_melted <- melt(p_value_matrix)

combined <- data.frame(
  Var1 = cor_melted$Var1,
  Var2 = cor_melted$Var2,
  Correlation = cor_melted$value,
  P_value = p_melted$value
)

# Flag significant correlations (p < 0.05)
combined <- combined %>%
  mutate(
    Significance = ifelse(P_value < 0.05, "*", ""),
    Significant = P_value < 0.05
  )

# Heatmap highlighting significant correlations with asterisks
ggplot(combined, aes(x = Var1, y = Var2, fill = Correlation)) +
  geom_tile(color = "white") +
  # Dim non-significant tiles
  geom_tile(data = subset(combined, !Significant),
            aes(fill = Correlation), alpha = 0.3) +
  # Add asterisks on significant tiles
  geom_text(data = subset(combined, Significant),
            aes(label = Significance),
            color = "black", size = 6, fontface = "bold") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       midpoint = 0, limit = c(-1, 1),
                       name = "Spearman\nCorrelation") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1, size = 8),
    axis.text.y = element_text(size = 8),
    axis.title = element_blank(),
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    legend.position = "right",
    plot.subtitle = element_text(hjust = 0.5, size = 10)
  ) +
  ggtitle("Spearman Correlation Heatmap (per match)") +
  labs(subtitle = "* p < 0.05") +
  coord_fixed()
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(ggplot2)
library(reshape2)
library(Hmisc)

# Read the Excel file
df <- read_excel("UEFA_EURO_12-24.xlsx")

# Select the per-match variables of interest
vars <- c(
  "Fouls Suffered_per_match",
  "Free-kicks Taken_per_match",
  "Corners Taken_per_match",
  "Attacks_per_match",
  "Passes Attempted_per_match",
  "Passes Completed_per_match",
  "Yellow Cards_per_match",
  "Crosses Completed_per_match",
  "Crossing Accuracy (%)",
  "Tackles_per_match",
  "Possession (%)",
  "Tackles Lost_per_match"
)

# Keep only these variables
df_per_match <- df %>%
  select(all_of(vars))

# Pearson correlation with p-values
cor_result <- rcorr(as.matrix(df_per_match), type = "pearson")
cor_matrix <- cor_result$r
p_value_matrix <- cor_result$P

# Melt matrices for ggplot
cor_melted <- melt(cor_matrix)
p_melted <- melt(p_value_matrix)

combined <- data.frame(
  Var1 = cor_melted$Var1,
  Var2 = cor_melted$Var2,
  Correlation = cor_melted$value,
  P_value = p_melted$value
)

# Flag significant correlations (p < 0.05)
combined <- combined %>%
  mutate(
```

```
    Significance = ifelse(P_value < 0.05, "*", ""),
    Significant = P_value < 0.05
  )

# Heatmap highlighting significant correlations with asterisks
ggplot(combined, aes(x = Var1, y = Var2, fill = Correlation)) +
  geom_tile(color = "white") +
  geom_tile(data = subset(combined, !Significant),
            aes(fill = Correlation), alpha = 0.3) +
  geom_text(data = subset(combined, Significant),
            aes(label = Significance),
            color = "black", size = 6, fontface = "bold") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       midpoint = 0, limit = c(-1, 1),
                       name = "Pearson\nCorrelation") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1),
    axis.title = element_blank(),
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 10)
  ) +
  ggtitle("Pearson Correlation Heatmap (per match)") +
  labs(subtitle = "* p < 0.05") +
  coord_fixed()
```

```
# Load required libraries
library(readxl)
library(dplyr)

# Read the Excel file
data <- read_excel("UEFA_EURO_12-24.xlsx")

# Define the quantitative variables for analysis
quantitative_vars <- c(
  "Goals_per_match", "Right foot_per_match", "Left foot_per_match",
  "Head_per_match", "Other_per_match", "Goals inside area_per_match",
  "Goals outside area_per_match", "Penalties scored_per_match",
  "Total Attempts_per_match", "Attempts on Target_per_match",
  "Attempts off Target_per_match", "Blocked_per_match",
  "Passing Accuracy (%)", "Passes Attempted_per_match",
  "Passes Completed_per_match", "Possession (%)",
  "Crossing Accuracy (%)", "Crosses Attempted_per_match",
  "Crosses Completed_per_match", "Free-kicks Taken_per_match",
  "Attacks_per_match", "Assists_per_match", "Corners Taken_per_match",
  "Offsides_per_match", "Tackles_per_match", "Tackles Won_per_match",
  "Tackles Lost_per_match", "Saves_per_match",
  "Goals Conceded_per_match", "Own Goals Conceded_per_match",
  "Saves from Penalties_per_match", "Clean Sheets_per_match",
```

```
  "Fouls Commited_per_match", "Fouls Suffered_per_match",
  "Yellow Cards_per_match", "Red Cards_per_match"
)

# Function to compute Spearman correlations and p-values
create_spearman_cor_table_with_pvalues <- function(data, quant_vars, cat_vars) {
  cor_values_list <- list()
  pvalues_list <- list()

  for (cat_var in cat_vars) {
    # Convert categorical variable to numeric
    if (cat_var == "Final Rank") {
      rank_levels <- c("Winner", "Finalist", "Semi-Finals", "Quarter-Finals",
                       "Round of 16", "Group Stage")
      cat_numeric <- as.numeric(factor(data[[cat_var]], levels = rank_levels))
    } else {
      cat_numeric <- as.numeric(data[[cat_var]])
    }

    cor_values <- numeric(length(quant_vars))
    pvalues <- numeric(length(quant_vars))

    for (i in seq_along(quant_vars)) {
      quant_var <- quant_vars[i]
      if (quant_var %in% names(data)) {
        cor_test <- cor.test(data[[quant_var]], cat_numeric,
                             method = "spearman", use = "complete.obs",
                             exact = FALSE)
        cor_values[i] <- round(cor_test$estimate, 3)
        pvalues[i] <- round(cor_test$p.value, 4)
      } else {
        cor_values[i] <- NA
        pvalues[i] <- NA
      }
    }

    cor_values_list[[cat_var]] <- cor_values
    pvalues_list[[paste0(cat_var, "_pvalue")]] <- pvalues
  }

  cor_table <- do.call(cbind, cor_values_list)
  pvalue_table <- do.call(cbind, pvalues_list)
  rownames(cor_table) <- quant_vars
  rownames(pvalue_table) <- quant_vars

  return(list(correlations = cor_table, pvalues = pvalue_table))
}

# Compute correlations for Final Rank and Year
categorical_vars <- c("Final Rank", "Year")
```

```
results <- create_spearman_cor_table_with_pvalues(data, quantitative_vars,
categorical_vars)

# Build summary table
final_table <- data.frame(
  Variable = quantitative_vars,
  Correlation_FinalRank = results$correlations[, "Final Rank"],
  Pvalue_FinalRank = results$pvalues[, "Final Rank_pvalue"],
  Significant_FinalRank = ifelse(results$pvalues[, "Final Rank_pvalue"] < 0.05, "*",
""),
  Correlation_Year = results$correlations[, "Year"],
  Pvalue_Year = results$pvalues[, "Year_pvalue"],
  Significant_Year = ifelse(results$pvalues[, "Year_pvalue"] < 0.05, "*", "")
)

# Print the table
print(final_table)
```

```
# Load required libraries
library(readxl)
library(dplyr)

# Read the Excel file
euro_data <- read_excel("UEFA_EURO_12-24.xlsx")

# Define per-match variables
per_match_vars <- c(
  "Goals_per_match", "Right foot_per_match", "Left foot_per_match",
  "Head_per_match", "Other_per_match", "Goals inside area_per_match",
  "Goals outside area_per_match", "Penalties scored_per_match",
  "Total Attempts_per_match", "Attempts on Target_per_match",
  "Attempts off Target_per_match", "Blocked_per_match",
  "Passing Accuracy (%)", "Passes Attempted_per_match",
  "Passes Completed_per_match", "Possession (%)",
  "Crossing Accuracy (%)", "Crosses Attempted_per_match",
  "Crosses Completed_per_match", "Free-kicks Taken_per_match",
  "Attacks_per_match", "Assists_per_match", "Corners Taken_per_match",
  "Offsides_per_match", "Tackles_per_match", "Tackles Won_per_match",
  "Tackles Lost_per_match", "Saves_per_match",
  "Goals Conceded_per_match", "Own Goals Conceded_per_match",
  "Saves from Penalties_per_match", "Clean Sheets_per_match",
  "Fouls Commited_per_match", "Fouls Suffered_per_match",
  "Yellow Cards_per_match", "Red Cards_per_match"
)

# Create per-match result metrics
euro_data <- euro_data %>%
  mutate(
    Won_per_match = Won / `Matches played`,
```

```
    Drawn_per_match = Drawn / `Matches played`,
    Lost_per_match = Lost / `Matches played`
  )

results_vars <- c("Won_per_match", "Drawn_per_match", "Lost_per_match")

# Compute Spearman correlation matrix
correlation_matrix <- cor(euro_data[per_match_vars],
                          euro_data[results_vars],
                          method = "spearman",
                          use = "complete.obs")

# Compute p-values for each pair
p_value_matrix <- matrix(NA, nrow = nrow(correlation_matrix), ncol =
ncol(correlation_matrix))
rownames(p_value_matrix) <- rownames(correlation_matrix)
colnames(p_value_matrix) <- colnames(correlation_matrix)

for (i in 1:nrow(correlation_matrix)) {
  for (j in 1:ncol(correlation_matrix)) {
    cor_test <- cor.test(euro_data[[rownames(correlation_matrix)[i]]],
                         euro_data[[colnames(correlation_matrix)[j]]],
                         method = "spearman",
                         use = "complete.obs",
                         exact = FALSE)
    p_value_matrix[i, j] <- cor_test$p.value
  }
}

# Top correlations with Wins (statistically significant at 5% level)
won_results <- data.frame(
  Variable = rownames(correlation_matrix),
  Correlation = correlation_matrix[, "Won_per_match"],
  P_Value = p_value_matrix[, "Won_per_match"],
  Significant = p_value_matrix[, "Won_per_match"] < 0.05
) %>%
  arrange(desc(abs(Correlation))) %>%
  mutate(
    Correlation = round(Correlation, 3),
    P_Value = round(P_Value, 4),
    Significance = ifelse(Significant, "*", "")
  )
print("Top correlations with Wins:")
print(won_results)

# Top correlations with Draws
drawn_results <- data.frame(
  Variable = rownames(correlation_matrix),
  Correlation = correlation_matrix[, "Drawn_per_match"],
  P_Value = p_value_matrix[, "Drawn_per_match"],
```

```
  Significant = p_value_matrix[, "Drawn_per_match"] < 0.05
) %>%
  arrange(desc(abs(Correlation))) %>%
  mutate(
    Correlation = round(Correlation, 3),
    P_Value = round(P_Value, 4),
    Significance = ifelse(Significant, "*", "")
  )
print("Top correlations with Draws:")
print(drawn_results)

# Top correlations with Losses
lost_results <- data.frame(
  Variable = rownames(correlation_matrix),
  Correlation = correlation_matrix[, "Lost_per_match"],
  P_Value = p_value_matrix[, "Lost_per_match"],
  Significant = p_value_matrix[, "Lost_per_match"] < 0.05
) %>%
  arrange(desc(abs(Correlation))) %>%
  mutate(
    Correlation = round(Correlation, 3),
    P_Value = round(P_Value, 4),
    Significance = ifelse(Significant, "*", "")
  )
print("Top correlations with Losses:")
print(lost_results)
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(tidyr)
library(nortest)
library(car)

# Read data and create Phase
data <- read_excel("UEFA_EURO_12-24.xlsx")

data <- data %>%
  mutate(
    Phase = case_when(
      `Final Rank` == "Group Stage" ~ "Low",
      `Final Rank` %in% c("Round of 16", "Quarter-Finals") ~ "Medium",
      `Final Rank` %in% c("Semi-Finals", "Finalist", "Winner") ~ "High",
      TRUE ~ NA_character_
    )
  )

# Variables to analyze (exact match to desired tables)
vars <- c(
```

```
  "Fouls Suffered_per_match",
  "Free-kicks Taken_per_match",
  "Corners Taken_per_match",
  "Attacks_per_match",
  "Passes Attempted_per_match",
  "Passes Completed_per_match",
  "Possession (%)"
)

# Clean variable names for display
clean_names <- c(
  "Fouls Suffered",
  "Free-kicks Taken",
  "Corners Taken",
  "Attacks",
  "Passes Attempted",
  "Passes Completed",
  "Possession (%)"
)

# 1. Lilliefors Test for Normality (p-values per Phase)
normality <- data %>%
  select(all_of(vars), Phase) %>%
  pivot_longer(cols = -Phase, names_to = "Variable", values_to = "Value") %>%
  group_by(Phase, Variable) %>%
  summarise(p_value = lillie.test(Value)$p.value, .groups = "drop") %>%
  pivot_wider(names_from = Phase, values_from = p_value) %>%
  select(Variable, Medium, Low, High) %>%
  mutate(Variable = factor(Variable, levels = vars, labels = clean_names)) %>%
  arrange(Variable)

cat("Lilliefors Test for Normality\n")
print(as.data.frame(normality), row.names = FALSE)

# 2. Levene's Test for Homogeneity of Variances
levene_results <- data.frame(Variable = character(),
                             Levene_F = numeric(),
                             p_value = numeric(),
                             stringsAsFactors = FALSE)

for (i in seq_along(vars)) {
  var <- vars[i]
  f <- as.formula(paste0("`", var, "` ~ Phase"))
  lev <- leveneTest(f, data = data)
  levene_results <- rbind(levene_results,
                          data.frame(Variable = clean_names[i],
                                     Levene_F = round(lev$`F value`[1], 4),
                                     p_value = round(lev$`Pr(>F)`[1], 4)))
}
```

```
cat("\nLevene's Test for Homogeneity of Variances\n")
print(levene_results, row.names = FALSE)

# 3. ANOVA
anova_results <- data.frame(Variable = character(),
                            F_value = numeric(),
                            p_value = numeric(),
                            stringsAsFactors = FALSE)

models <- list()

for (i in seq_along(vars)) {
  var <- vars[i]
  f <- as.formula(paste0("`", var, "` ~ Phase"))
  mod <- aov(f, data = data)
  models[[clean_names[i]]] <- mod
  sm <- summary(mod)
  anova_results <- rbind(anova_results,
                         data.frame(Variable = clean_names[i],
                                    F_value = round(sm[[1]]$`F value`[1], 4),
                                    p_value = round(sm[[1]]$`Pr(>F)`[1], 4)))
}

cat("\nANOVA Test\n")
print(anova_results, row.names = FALSE)

# 4. Post-hoc Tukey HSD for significant ANOVAs
sig_vars <- anova_results$Variable[anova_results$p_value < 0.05]

cat("\nPOST-HOC ANALYSIS (Tukey HSD)\n")
for (var in sig_vars) {
  cat("\n", var, "\n", sep = "")
  tuk <- TukeyHSD(models[[var]])
  tuk_df <- as.data.frame(tuk$Phase)
  tuk_df$Comparison <- rownames(tuk_df)
  tuk_df <- tuk_df[, c("Comparison", "diff", "p adj")]
  colnames(tuk_df)[2:3] <- c("Diff", "p-value")
  # Keep only the order shown: Low-High, Medium-High, Medium-Low (already in
output)
  print(tuk_df, row.names = FALSE)
}
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(FSA)

# Read and prepare data
data <- read_excel("UEFA_EURO_12-24.xlsx")
```

```
data <- data %>%
  mutate(Phase = case_when(
    `Final Rank` == "Group Stage" ~ "Low",
    `Final Rank` %in% c("Round of 16", "Quarter-Finals") ~ "Medium",
    `Final Rank` %in% c("Semi-Finals", "Finalist", "Winner") ~ "High",
    TRUE ~ NA_character_
  )) %>%
  mutate(Phase = factor(Phase, levels = c("Low", "Medium", "High")))

# Variables to analyse
variables <- c("Goals_per_match", "Total Attempts_per_match",
               "Attempts off Target_per_match", "Attempts on Target_per_match",
               "Assists_per_match", "Offsides_per_match",
               "Passing Accuracy (%)", "Goals Conceded_per_match")

# Clean names for output
clean_names <- c("Goals", "Total Attempts", "Attempts off Target",
                 "Attempts on Target", "Assists", "Offsides",
                 "Passing Accuracy (%)", "Goals Conceded")

# Kruskal-Wallis tests
kw_results <- data.frame(Variable = character(),
                         H_Value = numeric(),
                         p_value = character(),
                         stringsAsFactors = FALSE)

for (i in seq_along(variables)) {
  var <- variables[i]
  f <- as.formula(paste0("`", var, "` ~ Phase"))
  kw <- kruskal.test(f, data = data)

  p_val <- kw$p.value
  p_text <- ifelse(p_val < 0.001, "<0.001", round(p_val, 4))

  kw_results <- rbind(kw_results,
                      data.frame(Variable = clean_names[i],
                                 H_Value = round(kw$statistic, 4),
                                 p_value = p_text,
                                 stringsAsFactors = FALSE))
}

cat("Kruskal-Wallis Test\n")
print(kw_results, row.names = FALSE)

# Dunn's test with Holm correction for significant Kruskal-Wallis (p < 0.05)
for (i in seq_along(variables)) {
  var <- variables[i]
  f <- as.formula(paste0("`", var, "` ~ Phase"))
  kw <- kruskal.test(f, data = data)
```

```
  if (kw$p.value < 0.05) {
    dunn <- dunnTest(f, data = data, method = "holm")
    dunn_df <- dunn$res

    # Format comparisons
    dunn_df$Comparison <- gsub(" - ", "-", dunn_df$Comparison)
    # Keep desired columns and order
    out <- dunn_df[, c("Comparison", "Z", "P.adj")]
    colnames(out) <- c("Categories", "Z Value", "p-value")

    # Format p-values
    out$`p-value` <- ifelse(out$`p-value` < 0.001, "<0.001", round(out$`p-value`, 4))
    out$`Z Value` <- round(out$`Z Value`, 4)

    cat("\n", clean_names[i], "\n", sep = "")
    cat("Dunn's test with the Holm correction\n")
    print(out, row.names = FALSE)
  }
}
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(tidyr)
library(nortest)
library(car)

# Read data and prepare Year factor
data <- read_excel("UEFA_EURO_12-24.xlsx")
data$Year <- factor(data$Year, levels = c("2012", "2016", "2020", "2024"))

# Variables of interest (same seven as before)
vars <- c("Fouls Suffered_per_match",
          "Free-kicks Taken_per_match",
          "Corners Taken_per_match",
          "Attacks_per_match",
          "Passes Attempted_per_match",
          "Passes Completed_per_match",
          "Possession (%)")

clean_names <- c("Fouls Suffered",
                 "Free-Kicks Taken",
                 "Corners Taken",
                 "Attacks",
                 "Passes Attempted",
                 "Passes Completed",
                 "Possession (%)")
```

```
# 1. Lilliefors Test for Normality (p-values per Year)
normality <- data %>%
  select(all_of(vars), Year) %>%
  pivot_longer(cols = -Year, names_to = "Variable", values_to = "Value") %>%
  group_by(Year, Variable) %>%
  summarise(p_value = lillie.test(Value)$p.value, .groups = "drop") %>%
  pivot_wider(names_from = Year, values_from = p_value) %>%
  mutate(Variable = factor(Variable, levels = vars, labels = clean_names)) %>%
  arrange(Variable)

cat("Lilliefors Test for Normality\n")
print(as.data.frame(normality), row.names = FALSE)

# 2. Levene's Test for Homogeneity of Variances
levene_results <- data.frame(Variable = character(),
                             Levene_F = numeric(),
                             p_value = numeric(),
                             stringsAsFactors = FALSE)

for (i in seq_along(vars)) {
  var <- vars[i]
  f <- as.formula(paste0("`", var, "` ~ Year"))
  lev <- leveneTest(f, data = data)
  levene_results <- rbind(levene_results,
                          data.frame(Variable = clean_names[i],
                                     Levene_F = round(lev$`F value`[1], 4),
                                     p_value = round(lev$`Pr(>F)`[1], 4)))
}

cat("\nLevene's Test for Homogeneity of Variances\n")
print(levene_results, row.names = FALSE)

# 3. ANOVA
anova_results <- data.frame(Variable = character(),
                            F_value = numeric(),
                            p_value = numeric(),
                            stringsAsFactors = FALSE)

models <- list()

for (i in seq_along(vars)) {
  var <- vars[i]
  f <- as.formula(paste0("`", var, "` ~ Year"))
  mod <- aov(f, data = data)
  models[[clean_names[i]]] <- mod
  sm <- summary(mod)
  anova_results <- rbind(anova_results,
                         data.frame(Variable = clean_names[i],
                                    F_value = round(sm[[1]]$`F value`[1], 4),
                                    p_value = round(sm[[1]]$`Pr(>F)`[1], 4)))
```

```
}

cat("\nAnova Test\n")
print(anova_results, row.names = FALSE)

# 4. Post-hoc Tukey HSD for significant ANOVAs
sig_vars <- anova_results$Variable[anova_results$p_value < 0.05]

cat("\nPOST-HOC ANALYSIS (Tukey HSD)\n")
for (var in sig_vars) {
  cat("\n", var, "\n", sep = "")
  tuk <- TukeyHSD(models[[var]])
  tuk_df <- as.data.frame(tuk$Year)
  tuk_df$Comparison <- rownames(tuk_df)
  tuk_df <- tuk_df[, c("Comparison", "diff", "p adj")]
  colnames(tuk_df)[2:3] <- c("Diff", "p-value")
  # Round for clarity
  tuk_df$Diff <- round(tuk_df$Diff, 4)
  tuk_df$`p-value` <- round(tuk_df$`p-value`, 4)
  print(tuk_df, row.names = FALSE)
}
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(FSA)

# Read data
data <- read_excel("UEFA_EURO_12-24.xlsx")
data$Year <- factor(data$Year, levels = c("2012", "2016", "2020", "2024"))

# Variables to analyse
variables <- c("Goals_per_match", "Total Attempts_per_match",
               "Attempts off Target_per_match", "Attempts on Target_per_match",
               "Assists_per_match", "Offsides_per_match",
               "Passing Accuracy (%)", "Goals Conceded_per_match")

# Clean names
clean_names <- c("Goals", "Total Attempts", "Attempts off Target",
                 "Attempts on Target", "Assists", "Offsides",
                 "Passing Accuracy (%)", "Goals Conceded")

# Kruskal-Wallis tests
kw_results <- data.frame(Variable = character(),
                         H_Value = numeric(),
                         p_value = character(),
                         stringsAsFactors = FALSE)

for (i in seq_along(variables)) {
```

```
  var <- variables[i]
  f <- as.formula(paste0("`", var, "` ~ Year"))
  kw <- kruskal.test(f, data = data)

  p_val <- kw$p.value
  p_text <- ifelse(p_val < 0.001, "<0.001", round(p_val, 4))

  kw_results <- rbind(kw_results,
                      data.frame(Variable = clean_names[i],
                                 H_Value = round(kw$statistic, 4),
                                 p_value = p_text,
                                 stringsAsFactors = FALSE))
}

cat("Kruskal-Wallis Test\n")
print(kw_results, row.names = FALSE)

# Dunn's test with Holm correction for significant KW results
for (i in seq_along(variables)) {
  var <- variables[i]
  f <- as.formula(paste0("`", var, "` ~ Year"))
  kw <- kruskal.test(f, data = data)

  if (kw$p.value < 0.05) {
    dunn <- dunnTest(f, data = data, method = "holm")
    dunn_df <- dunn$res

    # Format comparison string: remove spaces around hyphen, e.g., "2012 - 2016" ->
"2012-2016"
    dunn_df$Comparison <- gsub(" - ", "-", dunn_df$Comparison)

    out <- dunn_df[, c("Comparison", "Z", "P.adj")]
    colnames(out) <- c("Categories", "Z Value", "p-value")

    # Format p-values
    out$`p-value` <- ifelse(out$`p-value` < 0.001, "<0.001", round(out$`p-value`, 4))
    out$`Z Value` <- round(out$`Z Value`, 4)

    cat("\n", clean_names[i], "\n", sep = "")
    cat("Dunn's test with the Holm correction\n")
    print(out, row.names = FALSE)
  }
}
```

CHAPTER 5

```
# Load required libraries
library(readxl)
library(dplyr)
library(car)
```

```
library(ResourceSelection)
library(ggplot2)

# Read and prepare data
euro_data <- read_excel("UEFA_EURO_12-24.xlsx") %>%
  mutate(
    Progress = factor(
      case_when(
        `Final Rank` == "Group Stage" ~ "GroupStage",
        TRUE ~ "Advanced"
      ),
      levels = c("GroupStage", "Advanced")
    )
  ) %>%
  filter(!is.na(Progress))

# Keep only per_match variables and the outcome
per_match_cols <- grep("_per_match$", names(euro_data), value = TRUE)
analysis_data <- euro_data[, c("Progress", per_match_cols)]
analysis_data <- na.omit(analysis_data)

# Fit the final logistic model
m4 <- glm(Progress ~ `Attempts on Target_per_match` + `Passes
Completed_per_match` +
            Blocked_per_match + Attacks_per_match + `Goals Conceded_per_match` +
            `Saves from Penalties_per_match`,
          data = analysis_data, family = binomial)

# Logistic regression coefficients
coef_table <- data.frame(
  Variables = rownames(coef(summary(m4))),
  Estimate = round(coef(summary(m4))[, 1], 3),
  Std_Error = round(coef(summary(m4))[, 2], 3),
  Z_value = round(coef(summary(m4))[, 3], 3),
  P_value = round(coef(summary(m4))[, 4], 3)
)
rownames(coef_table) <- NULL

cat("Logistic Regression\n")
print(coef_table, row.names = FALSE)

# Model deviance and AIC
cat(sprintf("\nNull deviance: %.3f  on %d  degrees of freedom", m4$null.deviance,
m4$df.null))
cat(sprintf("\nResidual deviance: %.3f  on %d  degrees of freedom", m4$deviance,
m4$df.residual))
cat(sprintf("\nAIC: %.3f\n", AIC(m4)))

# VIF
vif_table <- data.frame(Variables = names(vif(m4)), VIF = round(vif(m4), 3))
```

```
rownames(vif_table) <- NULL
cat("\nVariance Inflation Factor (VIF) Analysis\n")
print(vif_table, row.names = FALSE)

# Analysis of deviance
anova_tab <- anova(m4, test = "Chisq")
anova_df <- data.frame(
  Variables = rownames(anova_tab),
  Df = anova_tab$Df,
  Deviance = round(anova_tab$Deviance, 3),
  Resid_Df = anova_tab$`Resid. Df`,
  Resid_Dev = round(anova_tab$`Resid. Dev`, 3),
  P_value = round(anova_tab$`Pr(>Chi)`, 3)
)
anova_df$Variables[1] <- "NULL"
cat("\nAnalysis of Deviance Table\n")
print(anova_df, row.names = FALSE)

# Hosmer-Lemeshow test
analysis_data$Progress_num <- as.numeric(analysis_data$Progress) - 1
hl <- hoslem.test(analysis_data$Progress_num, fitted(m4), g = 8)
hl_table <- data.frame(
  Fit_Measure = c("X-squared (H-L)", "Df", "P-value"),
  Value = c(round(hl$statistic, 3), hl$parameter, round(hl$p.value, 3))
)
cat("\nHosmer-Lemeshow Test\n")
print(hl_table, row.names = FALSE)

# Predictions and confusion matrix
analysis_data$pred_prob <- predict(m4, type = "response")
analysis_data$pred_class <- ifelse(analysis_data$pred_prob > 0.5, "Advanced",
"GroupStage")
analysis_data$pred_class <- factor(analysis_data$pred_class, levels =
c("GroupStage", "Advanced"))

conf_matrix <- table(Predicted = analysis_data$pred_class, Actual =
analysis_data$Progress)

# Accuracy, sensitivity, specificity
accuracy <- mean(analysis_data$pred_class == analysis_data$Progress)
sensitivity <- conf_matrix["Advanced", "Advanced"] / sum(conf_matrix[,
"Advanced"])
specificity <- conf_matrix["GroupStage", "GroupStage"] / sum(conf_matrix[,
"GroupStage"])

cat(sprintf("Accuracy: %.2f %%\n", accuracy * 100))
cat(sprintf("Sensitivity: %.2f %%\n", sensitivity * 100))
cat(sprintf("Specificity: %.2f %%\n", specificity * 100))

# Data frame for heatmap
```

```
conf_df <- as.data.frame(conf_matrix)
conf_df$Predicted <- factor(conf_df$Predicted, levels =
rev(levels(conf_df$Predicted)))

# Confusion matrix heatmap
ggplot(conf_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 8, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "#009194") +
  labs(title = "Confusion Matrix - Logistic Model",
       subtitle = paste("Overall Accuracy:", round(accuracy * 100, 1), "%"),
       x = "Actual Class", y = "Predicted Class") +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12),
        axis.text = element_text(size = 12))

# For Multinomial
mlt<- glm(Progress ~ `Attempts on Target_per_match`  +
            Blocked_per_match + Attacks_per_match + `Goals Conceded_per_match` +
            `Saves from Penalties_per_match`,
          data = analysis_data, family = binomial)

vif_table <- data.frame(Variables = names(vif(mlt)), VIF = round(vif(mlt), 3))
rownames(vif_table) <- NULL
cat("\nVariance Inflation Factor (VIF) Analysis\n")
print(vif_table, row.names = FALSE)
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(nnet)
library(car)
library(DescTools)
library(ggplot2)

# Read and prepare data
data <- read_excel("UEFA_EURO_12-24.xlsx")

analysis_data <- data %>%
  mutate(
    Phase = case_when(
      `Final Rank` == "Group Stage" ~ "Low",
      `Final Rank` %in% c("Round of 16", "Quarter-Finals") ~ "Medium",
      `Final Rank` %in% c("Semi-Finals", "Finalist", "Winner") ~ "High",
      TRUE ~ NA_character_
    )
  ) %>%
```

```
  filter(!is.na(Phase)) %>%
  mutate(Phase = factor(Phase, levels = c("Low", "Medium", "High"))) %>%
  select(
    Phase,
    `Penalties scored_per_match`,
    `Total Attempts_per_match`,
    `Attempts on Target_per_match`,
    `Attempts off Target_per_match`,
    `Blocked_per_match`,
    `Passes Attempted_per_match`,
    `Passes Completed_per_match`,
    `Crosses Attempted_per_match`,
    `Crosses Completed_per_match`,
    `Free-kicks Taken_per_match`,
    `Attacks_per_match`,
    `Assists_per_match`,
    `Corners Taken_per_match`,
    `Offsides_per_match`,
    `Tackles_per_match`,
    `Tackles Won_per_match`,
    `Tackles Lost_per_match`,
    `Saves_per_match`,
    `Goals Conceded_per_match`,
    `Own Goals Conceded_per_match`,
    `Saves from Penalties_per_match`,
    `Clean Sheets_per_match`,
    `Fouls Commited_per_match`,
    `Fouls Suffered_per_match`,
    `Yellow Cards_per_match`,
    `Red Cards_per_match`
  ) %>%
  na.omit()

# Fit final multinomial model
fit <- multinom(
  Phase ~ `Attempts on Target_per_match` + Blocked_per_match +
Attacks_per_match +
    `Goals Conceded_per_match` + `Saves from Penalties_per_match`,
  data = analysis_data,
  maxit = 1000
)

# 2. Analysis of Deviance (Type III)
anova_tab <- Anova(fit, type = "III")
# Extract LR Chisq, Df, Pr(>Chisq) from the Anova output
var_names <- rownames(anova_tab)
# Remove intercept row if present
main_effects <- var_names[var_names != "(Intercept)"]
deviance_table <- data.frame(
  Variables = main_effects,
```

```
  LR_Chisq = round(anova_tab[main_effects, "LR Chisq"], 3),
  Df = anova_tab[main_effects, "Df"],
  P_value = round(anova_tab[main_effects, "Pr(>Chisq)"], 3)
)
rownames(deviance_table) <- NULL
cat("\nAnalysis of Deviance Table\n")
print(deviance_table, row.names = FALSE)

# 3. McFadden's Pseudo R²
mcfadden <- PseudoR2(fit, which = "McFadden")
cat("\nFit Measure\n")
cat(sprintf("McFadden's Pseudo R²: %.3f\n", mcfadden))

# 4. Model coefficients table
coefs <- summary(fit)$coefficients  # matrix: Medium and High rows, coefficients
columns
coef_names <- colnames(coefs)
# Intercept is named "(Intercept)"
# Create data frame with Parameter, Medium Coef, Medium SE, High Coef, High SE
se <- summary(fit)$standard.errors
coef_table <- data.frame(
  Parameter = coef_names,
  `Medium Coef` = round(coefs["Medium", ], 3),
  `Medium SE` = round(se["Medium", ], 3),
  `High Coef` = round(coefs["High", ], 3),
  `High SE` = round(se["High", ], 3)
)
rownames(coef_table) <- NULL
cat("\nMultinomial Logistic Model\n")
cat("Parameter | Medium (Coef, SE) | High (Coef, SE)\n")
print(coef_table, row.names = FALSE)

# 5. Residual Deviance and AIC
cat(sprintf("\nResidual Deviance: %.3f\n", fit$deviance))
cat(sprintf("AIC: %.3f\n", AIC(fit)))

# 6. Confusion matrix and heatmap
analysis_data$pred_phase <- predict(fit, type = "class")
cm <- table(Predicted = analysis_data$pred_phase, Actual = analysis_data$Phase)
accuracy <- mean(analysis_data$pred_phase == analysis_data$Phase)

# Data frame for ggplot with reversed predicted levels (for correct diagonal)
conf_df <- as.data.frame(cm)
conf_df$Predicted <- factor(conf_df$Predicted, levels =
rev(levels(conf_df$Predicted)))

ggplot(conf_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 8, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "#009194") +
```

```
  labs(title = "Confusion Matrix - Multinomial Logistic Model",
       subtitle = paste("Overall Accuracy:", round(accuracy * 100, 1), "%"),
       x = "Actual Class", y = "Predicted Class") +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12),
        axis.text = element_text(size = 12))
```

CHAPTER 6

```
# Load required libraries
library(readxl)
library(dplyr)
library(Boruta)
library(ggplot2)

# Read and prepare data
data <- read_excel("UEFA_EURO_12-24.xlsx") %>%
  mutate(
    Progress = factor(
      case_when(
        `Final Rank` == "Group Stage" ~ "GroupStage",
        `Final Rank` %in% c("Round of 16", "Quarter-Finals",
                            "Semi-Finals", "Finalist", "Winner") ~ "Advanced",
        TRUE ~ NA_character_
      ),
      levels = c("GroupStage", "Advanced")
    )
  ) %>%
  filter(!is.na(Progress))

# Select per_match and percentage columns
per_match_cols <- grep("_per_match$", colnames(data), value = TRUE)
percentage_cols <- grep("%", colnames(data), value = TRUE)
selected_features <- unique(c(per_match_cols, percentage_cols))

# Exclude goal-related variables
exclude_vars <- c("Goals_per_match", "Right foot_per_match", "Left
foot_per_match",
                  "Goals inside area_per_match", "Goals outside area_per_match",
                  "Assists_per_match", "Head_per_match")
selected_features <- selected_features[!selected_features %in% exclude_vars]

# Boruta feature selection
features <- data[, selected_features]
labels1 <- data$Progress
set.seed(123)
boruta_res1 <- Boruta(labels1 ~ ., data = na.omit(features), doTrace = 0)
```

```
# Extract confirmed features and their importance scores
boruta_stats <- attStats(boruta_res1)
boruta_confirmed <- boruta_stats %>%
  filter(decision == "Confirmed") %>%
  arrange(desc(meanImp))
boruta_confirmed$Feature <- rownames(boruta_confirmed)

# Importance plot of selected features
ggplot(boruta_confirmed, aes(x = reorder(Feature, meanImp), y = meanImp, fill =
meanImp)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Selected Features by the Boruta Algorithm",
    subtitle = paste("Total:", nrow(boruta_confirmed), "features"),
    x = "",
    y = "Mean Importance"
  ) +
  theme_minimal() +
  theme(
    axis.text.y = element_text(size = 11),
    plot.title = element_text(face = "bold", size = 16)
  ) +
  scale_fill_gradient(low = "steelblue", high = "tomato") +
  guides(fill = "none")
```

```
# Load required libraries
library(readxl)
library(dplyr)
library(caret)
library(pROC)
library(ggplot2)

# Read data and create Progress target
euro_analysis <- read_excel("UEFA_EURO_12-24.xlsx") %>%
  mutate(
    Progress = factor(
      case_when(
        `Final Rank` == "Group Stage" ~ "GroupStage",
        TRUE ~ "Advanced"
      ),
      levels = c("GroupStage", "Advanced")
    )
  ) %>%
  filter(!is.na(Progress))

# Features selected by Boruta (previously determined)
features <- c(
  "Attempts on Target_per_match",        #"Attempts on Target_per_match"
```

```
  "Passes Attempted_per_match",          #"Attacks_per_match"
  "Passes Completed_per_match",          #"Goals Conceded_per_match"
  "Saves_per_match",                    #"Blocked_per_match"
  "Clean Sheets_per_match",              #"Saves from penalties_per_match"
  "Red Cards_per_match",
  "Passing Accuracy (%)",
  "Attacks_per_match",
  "Goals Conceded_per_match"
)

# Prepare data matrix and outcome
X <- euro_analysis[, features, drop = FALSE]
y <- euro_analysis$Progress
complete <- complete.cases(X)
X <- X[complete, , drop = FALSE]
y <- y[complete]

# 10-fold cross-validation setup
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final",
  verboseIter = FALSE
)

# Train KNN
set.seed(123)
knn_model <- train(
  x = X,
  y = y,
  method = "knn",
  trControl = ctrl,
  tuneGrid = expand.grid(k = seq(1, 25, by = 2)),
  metric = "ROC",
  preProcess = c("center", "scale")
)

# Best k and cross-validated predictions
best_k <- knn_model$bestTune$k
cv_preds <- knn_model$pred %>% filter(k == best_k)

# Performance metrics from CV
cm <- confusionMatrix(
  data = factor(cv_preds$pred, levels = levels(y)),
  reference = factor(cv_preds$obs, levels = levels(y))
)
TP <- cm$table[2, 2]
TN <- cm$table[1, 1]
```

```
FP <- cm$table[2, 1]
FN <- cm$table[1, 2]

accuracy  <- (TP + TN) / sum(cm$table)
precision <- TP / (TP + FP)
recall    <- TP / (TP + FN)           # Sensitivity
specificity <- TN / (TN + FP)
f1        <- 2 * (precision * recall) / (precision + recall)
youden    <- recall + specificity - 1
p0        <- (TP + TN) / sum(cm$table)
pe        <- (TP+FP)/sum(cm$table)*(TP+FN)/sum(cm$table) +
(TN+FN)/sum(cm$table)*(TN+FP)/sum(cm$table)
kappa     <- (p0 - pe) / (1 - pe)
mcc       <- (TP*TN - FP*FN) / sqrt( (TP+FP)*(TP+FN)*(TN+FP)*(TN+FN) )

# Print metrics
cat("Best k:", best_k, "\n")
cat(sprintf("Accuracy : %.3f\n", accuracy))
cat(sprintf("Precision: %.3f\n", precision))
cat(sprintf("Recall   : %.3f\n", recall))
cat(sprintf("Specificity: %.3f\n", specificity))
cat(sprintf("F1 Score : %.3f\n", f1))
cat(sprintf("Youden's Index: %.3f\n", youden))
cat(sprintf("Cohen's Kappa: %.3f\n", kappa))
cat(sprintf("MCC: %.3f\n", mcc))

# === Plot 1: ROC AUC vs k ===
plot_k <- knn_model$results
best_point <- plot_k[plot_k$k == best_k, ]

ggplot(plot_k, aes(x = k, y = ROC)) +
  geom_line(color = "darkorange", linewidth = 1.2) +
  geom_point(color = "steelblue", size = 2) +
  geom_vline(xintercept = best_k, linetype = "dashed", color = "gray40", alpha = 0.7)
+
  geom_point(data = best_point, aes(x = k, y = ROC), color = "red", size = 3) +
  annotate("text", x = best_k + 1, y = min(plot_k$ROC) + 0.02,
           label = paste("k =", best_k), color = "black", fontface = "bold", hjust = 0) +
  labs(
    title = "KNN Performance vs. Number of Neighbors (k)",
    subtitle = "10-Fold Cross-Validation (ROC AUC)",
    x = "k (Number of Neighbors)",
    y = "ROC AUC"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 11)
  )
```

```
# === Plot 3: Confusion Matrix Heatmap ===
cm_df <- as.data.frame(cm$table)
cm_df$Prediction <- factor(cm_df$Prediction, levels =
rev(levels(cm_df$Prediction)))

ggplot(cm_df, aes(Reference, Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 6, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "#009194") +
  labs(title = "Confusion Matrix - KNN by Cross-Validation",
       subtitle = sprintf("Accuracy: %.1f%% | F1: %.3f | Precision: %.3f | Recall: %.3f",
                          accuracy * 100, f1, precision, recall),
       x = "Actual Class",
       y = "Predicted Class") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 10))
```

```
# Load required libraries
library(caret)
library(e1071)
library(ggplot2)
library(dplyr)
library(readxl)
library(pROC)

# Read and prepare data
euro_data <- read_excel("UEFA_EURO_12-24.xlsx") %>%
  mutate(
    Progress = factor(
      case_when(
        `Final Rank` == "Group Stage" ~ "GroupStage",
        TRUE ~ "Advanced"
      ),
      levels = c("GroupStage", "Advanced")
    )
  ) %>%
  filter(!is.na(Progress))

# Boruta features
boruta_features <- c(
  "Attempts on Target_per_match",        #"Attempts on Target_per_match"
  "Passes Attempted_per_match",          #"Attacks_per_match"
  "Passes Completed_per_match",          #"Goals Conceded_per_match"
  "Saves_per_match",                   #"Blocked_per_match"
  "Clean Sheets_per_match",              #"Saves from penalties_per_match"
  "Red Cards_per_match",
  "Passing Accuracy (%)",
  "Attacks_per_match",
```

```
  "Goals Conceded_per_match"
)

# Prepare data matrix and outcome
X <- euro_data[, boruta_features, drop = FALSE]
y <- euro_data$Progress
complete <- complete.cases(X)
X <- X[complete, , drop = FALSE]
y <- y[complete]
svm_data <- data.frame(X, Progress = y)

# 10-fold CV with internal scaling
set.seed(123)
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final",
  verboseIter = FALSE
)

# Train SVM (RBF kernel)
svm_model <- train(
  Progress ~ .,
  data = svm_data,
  method = "svmRadial",
  trControl = ctrl,
  tuneLength = 10,
  metric = "ROC",
  preProcess = c("center", "scale")
)

# Best parameters and CV ROC
best_params <- svm_model$bestTune
best_roc <- max(svm_model$results$ROC, na.rm = TRUE)

# Filter CV predictions for the best tune
best_preds <- svm_model$pred %>%
  filter(C == best_params$C & sigma == best_params$sigma)

# Confusion matrix from CV
cm <- confusionMatrix(
  data = factor(best_preds$pred, levels = levels(y)),
  reference = factor(best_preds$obs, levels = levels(y))
)
TP <- cm$table[2, 2]
TN <- cm$table[1, 1]
FP <- cm$table[2, 1]
FN <- cm$table[1, 2]
```

```
# Performance metrics
accuracy    <- (TP + TN) / sum(cm$table)
sensitivity <- TP / (TP + FN)
specificity <- TN / (TN + FP)
precision   <- TP / (TP + FP)
f1          <- 2 * (precision * sensitivity) / (precision + sensitivity)
youden      <- sensitivity + specificity - 1
p0          <- (TP + TN) / sum(cm$table)
pe          <- ((TP+FP) * (TP+FN) + (TN+FN) * (TN+FP)) / sum(cm$table)^2
kappa       <- (p0 - pe) / (1 - pe)
mcc_num     <- TP * TN - FP * FN
mcc_den     <- sqrt((TP+FP) * (TP+FN) * (TN+FP) * (TN+FN))
mcc         <- mcc_num / mcc_den

# Print all metrics
cat("Best C:", best_params$C, "\n")
cat("Best Sigma:", round(best_params$sigma, 4), "\n")
cat(sprintf("Best ROC AUC: %.4f\n", best_roc))
cat(sprintf("Accuracy : %.3f (%.1f%%)\n", accuracy, accuracy*100))
cat(sprintf("Sensitivity (Advanced): %.3f\n", sensitivity))
cat(sprintf("Specificity (GroupStage): %.3f\n", specificity))
cat(sprintf("Precision (Advanced): %.3f\n", precision))
cat(sprintf("F1 Score : %.3f\n", f1))
cat(sprintf("Youden's Index: %.3f\n", youden))
cat(sprintf("Cohen's Kappa: %.3f\n", kappa))
cat(sprintf("MCC: %.3f\n", mcc))

# ============================================================
# Confusion Matrix Heatmap
# ============================================================
cm_df <- as.data.frame(cm$table)
cm_df$Prediction <- factor(cm_df$Prediction, levels =
rev(levels(cm_df$Prediction)))

ggplot(cm_df, aes(Reference, Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 6, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "#009194") +
  labs(
    title = "Confusion Matrix - SVM by Cross-Validation",
    subtitle = paste("Accuracy:", round(accuracy * 100, 1),
                     "% | F1:", round(f1, 3),
                     "| Precision:", round(precision, 3),
                     "| Recall:", round(sensitivity, 3)),
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal() +
  theme(
```

```
  plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
  plot.subtitle = element_text(hjust = 0.5, size = 10)
 )
```

Code in Python

CHAPTER 6

```
import pandas as pd
import numpy as np
import random
import os

# Reproducibility: set seeds
np.random.seed(42)
random.seed(42)
import tensorflow as tf
tf.random.set_seed(42)
# For full determinism on CPU (if using GPU there may be small deviations)
os.environ['TF_DETERMINISTIC_OPS'] = '1'
tf.config.threading.set_inter_op_parallelism_threads(1)
tf.config.threading.set_intra_op_parallelism_threads(1)

from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import (confusion_matrix, accuracy_score, precision_score,
                             recall_score, f1_score, roc_auc_score, roc_curve,
                             cohen_kappa_score, matthews_corrcoef)
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input, Dropout
from tensorflow.keras.optimizers import Adam
import warnings
warnings.filterwarnings('ignore')

# ------------------------------
# 1. Data preparation
# ------------------------------
df = pd.read_excel("UEFA_EURO_12-24.xlsx", sheet_name="Sheet1")

def create_progress(rank):
    if rank == "Group Stage":
        return "GroupStage"
    elif rank in ["Round of 16", "Quarter-Finals", "Semi-Finals", "Finalist", "Winner"]:
        return "Advanced"
    return None

df["Progress"] = df["Final Rank"].apply(create_progress)
```

```
df = df.dropna(subset=["Progress"]).copy()

selected_features = [
  "Attempts on Target_per_match",        #"Attempts on Target_per_match"
 "Passes Attempted_per_match",          #"Attacks_per_match"
 "Passes Completed_per_match",          #"Goals Conceded_per_match"
 "Saves_per_match",                   #"Blocked_per_match"
 "Clean Sheets_per_match",              #"Saves from Penalties_per_match"
 "Red Cards_per_match",
 "Passing Accuracy (%)",
 "Attacks_per_match",
 "Goals Conceded_per_match"
]

X = df[selected_features].values
y = df["Progress"].values

# Encoding: originally Advanced=0, GroupStage=1 → invert so Advanced=1
(positive class)
le = LabelEncoder()
y_encoded = le.fit_transform(y)   # Advanced=0, GroupStage=1
y_encoded = 1 - y_encoded         # Advanced=1, GroupStage=0

# Print class order for confirmation
print("Class order (after inversion):")
print(f"0 = GroupStage")
print(f"1 = Advanced")
print()

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# ------------------------------
# 2. Model creation function
# ------------------------------
def create_model(input_dim, learning_rate, dropout_rate, neurons_layer1,
neurons_layer2):
    model = Sequential([
        Input(shape=(input_dim,)),
        Dense(neurons_layer1, activation='relu'),
        Dropout(dropout_rate),
        Dense(neurons_layer2, activation='relu'),
        Dropout(dropout_rate),
        Dense(1, activation='sigmoid')
    ])
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer,
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
# ------------------------------
# 3. Model parameters
# ------------------------------
best_params = {
    'neurons_layer2': 16,
    'neurons_layer1': 32,
    'learning_rate': 0.001,
    'epochs': 40,                # 80
    'dropout_rate': 0.5,
    'batch_size': 4
}

# ------------------------------
# 4. 10-fold Cross Validation
# ------------------------------
cv_final = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

y_true_all = []
y_pred_all = []
y_prob_all = []

# Lists to store histories
train_acc_per_fold = []
val_acc_per_fold = []
train_loss_per_fold = []
val_loss_per_fold = []

fold = 0
for train_idx, val_idx in cv_final.split(X_scaled, y_encoded):
    fold += 1
    print(f"Fold {fold}/10")

    X_train, X_val = X_scaled[train_idx], X_scaled[val_idx]
    y_train, y_val = y_encoded[train_idx], y_encoded[val_idx]

    model = create_model(
        input_dim=X_scaled.shape[1],
        learning_rate=best_params['learning_rate'],
        dropout_rate=best_params['dropout_rate'],
        neurons_layer1=best_params['neurons_layer1'],
        neurons_layer2=best_params['neurons_layer2']
    )

    history = model.fit(
        X_train, y_train,
        epochs=best_params['epochs'],
        batch_size=best_params['batch_size'],
        verbose=0,
        validation_data=(X_val, y_val)
```

```
    )

    # Store metrics
    train_acc_per_fold.append(history.history['accuracy'])
    val_acc_per_fold.append(history.history['val_accuracy'])
    train_loss_per_fold.append(history.history['loss'])
    val_loss_per_fold.append(history.history['val_loss'])

    # Predictions (probability and class)
    y_prob = model.predict(X_val, verbose=0).flatten()
    y_pred = (y_prob >= 0.5).astype(int)

    y_true_all.extend(y_val)
    y_pred_all.extend(y_pred)
    y_prob_all.extend(y_prob)

# ------------------------------
# 5. Average curves
# ------------------------------
train_acc_per_fold = np.array(train_acc_per_fold)
val_acc_per_fold = np.array(val_acc_per_fold)
train_loss_per_fold = np.array(train_loss_per_fold)
val_loss_per_fold = np.array(val_loss_per_fold)

mean_train_acc = np.mean(train_acc_per_fold, axis=0)
std_train_acc = np.std(train_acc_per_fold, axis=0)
mean_val_acc = np.mean(val_acc_per_fold, axis=0)
std_val_acc = np.std(val_acc_per_fold, axis=0)

mean_train_loss = np.mean(train_loss_per_fold, axis=0)
std_train_loss = np.std(train_loss_per_fold, axis=0)
mean_val_loss = np.mean(val_loss_per_fold, axis=0)
std_val_loss = np.std(val_loss_per_fold, axis=0)

epochs_range = range(1, best_params['epochs'] + 1)

# ------------------------------
# 6. Accuracy / Loss plots
# ------------------------------
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

axes[0].plot(epochs_range, mean_train_acc, 'b-', label='Train accuracy', linewidth=2)
axes[0].fill_between(epochs_range,
                     mean_train_acc - std_train_acc,
                     mean_train_acc + std_train_acc,
                     alpha=0.2, color='b')
axes[0].plot(epochs_range, mean_val_acc, 'r-', label='Validation accuracy',
linewidth=2)
axes[0].fill_between(epochs_range,
                     mean_val_acc - std_val_acc,
```

```
                mean_val_acc + std_val_acc,
                alpha=0.2, color='r')
axes[0].set_title('Accuracy over epochs (10-fold CV)', fontsize=14, fontweight='bold')
axes[0].set_xlabel('Epochs', fontsize=12)
axes[0].set_ylabel('Accuracy', fontsize=12)
axes[0].legend(loc='lower right', fontsize=11)
axes[0].grid(True, alpha=0.3)
axes[0].set_ylim([0, 1])

axes[1].plot(epochs_range, mean_train_loss, 'b-', label='Train loss', linewidth=2)
axes[1].fill_between(epochs_range,
                mean_train_loss - std_train_loss,
                mean_train_loss + std_train_loss,
                alpha=0.2, color='b')
axes[1].plot(epochs_range, mean_val_loss, 'r-', label='Validation loss', linewidth=2)
axes[1].fill_between(epochs_range,
                mean_val_loss - std_val_loss,
                mean_val_loss + std_val_loss,
                alpha=0.2, color='r')
axes[1].set_title('Loss over epochs (10-fold CV)', fontsize=14, fontweight='bold')
axes[1].set_xlabel('Epochs', fontsize=12)
axes[1].set_ylabel('Loss', fontsize=12)
axes[1].legend(loc='upper right', fontsize=11)
axes[1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# ------------------------------
# 7. Calculation of basic metrics and confusion matrix
# ------------------------------
y_true_all = np.array(y_true_all)
y_pred_all = np.array(y_pred_all)
y_prob_all = np.array(y_prob_all)

cm = confusion_matrix(y_true_all, y_pred_all)   # order: [GroupStage, Advanced]
(0,1)

class_names = ['GroupStage', 'Advanced']  # 0,1 respectively

# Absolute values of confusion matrix
tn, fp, fn, tp = cm.ravel()   # tn=cm[0,0], fp=cm[0,1], fn=cm[1,0], tp=cm[1,1]

print("\n" + "="*80)
print(" CONFUSION MATRIX (Absolute values) ".center(80, "="))
print("="*80)
print(f"{":20} Predicted GroupStage   Predicted Advanced")
print(f"True GroupStage{":8} {tn:>6}              {fp:>6}")
print(f"True Advanced{":10} {fn:>6}              {tp:>6}")
print("-"*80)
```

```
# Heatmap (absolute counts) - SWAPPED AXES
plt.figure(figsize=(8, 6))

cm_swapped = cm.T   # transpose matrix

ax = sns.heatmap(cm_swapped, annot=True, fmt='d', cmap='Blues',
                 xticklabels=class_names,
                 yticklabels=class_names,
                 cbar_kws={'label': 'Count'},
                 linewidths=2,
                 linecolor='black',
                 annot_kws={'size': 16})

total = np.sum(cm_swapped)

for i in range(2):
    for j in range(2):
        ax.text(j + 0.5, i + 0.7,
                f'({cm_swapped[i, j]/total*100:.1f}%)',
                ha='center',
                va='center',
                fontsize=11,
                fontweight='bold')

plt.title('Confusion Matrix - 10-fold CV',
          fontsize=16,
          fontweight='bold')

# SWAPPED LABELS
plt.xlabel('Actual', fontsize=14)
plt.ylabel('Prediction', fontsize=14)

plt.tight_layout()
plt.show()

# ------------------------------
# 8. Calculation of all requested metrics
# ------------------------------
accuracy = accuracy_score(y_true_all, y_pred_all)
sensitivity = recall_score(y_true_all, y_pred_all, pos_label=1)   # TP/(TP+FN)
specificity = tn / (tn + fp) if (tn+fp) > 0 else 0                # TN/(TN+FP)
precision = precision_score(y_true_all, y_pred_all, pos_label=1)  # TP/(TP+FP)
f1 = f1_score(y_true_all, y_pred_all, pos_label=1)
youden = sensitivity + specificity - 1
kappa = cohen_kappa_score(y_true_all, y_pred_all)
mcc = matthews_corrcoef(y_true_all, y_pred_all)
auc = roc_auc_score(y_true_all, y_prob_all)

# Print aggregate results
```

```
print("\n" + "="*80)
print(" AGGREGATE RESULTS 10-FOLD CV ".center(80, "="))
print("="*80)
print(f"{'Metric':<30} {'Value':>10}   {'Interpretation'}")
print("-"*80)
print(f"{'Accuracy':<30} {accuracy:>10.4f}   Overall correctness")
print(f"{'Sensitivity (Recall)':<30} {sensitivity:>10.4f}   Correctly identified
Advanced")
print(f"{'Specificity':<30} {specificity:>10.4f}   Correctly identified GroupStage")
print(f"{'Precision':<30} {precision:>10.4f}   Precision of Advanced predictions")
print(f"{'F1-score':<30} {f1:>10.4f}   Harmonic mean of Precision & Recall")
print(f"{'Youden's Index':<30} {youden:>10.4f}   Sensitivity + Specificity - 1")
print(f"{'Cohen's Kappa (κ)':<30} {kappa:>10.4f}   Agreement beyond chance")
print(f"{'MCC':<30} {mcc:>10.4f}   Matthews correlation coefficient")
print(f"{'ROC-AUC':<30} {auc:>10.4f}   Area under the ROC curve")
print("="*80)

print(f"\n Total samples: {len(y_true_all)}")
print(f" Class distribution:")
print(f"   • Advanced (Qualification): {np.sum(y_true_all==1)} samples
({np.sum(y_true_all==1)/len(y_true_all)*100:.1f}%)")
print(f"   • GroupStage (Elimination): {np.sum(y_true_all==0)} samples
({np.sum(y_true_all==0)/len(y_true_all)*100:.1f}%)")
print("="*80)
```