UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

A study on the performance of machine learning algorithms for predicting professional football match outcomes

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Pruñonosa Soler, Guillem

Tutor: Sánchez Anguix, Víctor

Cotutor: Alberola Oltra, Juan Miguel

ACADEMIC YEAR: 2022/2023

Abstract

Sporting events, especially those involving competition between teams, involve complex dynamics among the participants that ultimately affect the final outcome. Predicting the outcome of these sporting events in advance is a complex task precisely because of these complex dynamics that arise in the event, as well as the decisions that are made by all participants. In many cases, even human experts make mistakes in their predictions and surprising results are produced against all odds. The task presents challenges from the point of view of analytics and machine learning, as it is a problem that even presents difficulties to human experts. In this TFG we study the performance of different machine learning algorithms in the task of predicting the outcome of professional football matches in order to try to predict the final result (i.e., home win, draw, away win) based on information exclusively available before the match. For this purpose, different types of features (e.g., aggregate features, line-ups) and datasets will be posed and their performance on the performance of machine learning algorithms will be studied, as well as different configurations will be tested in order to find the model that is able to best predict the outcomes of professional football matches. Once these predictive models are trained, we think they can be useful in the creation of decision support tools for coaches. For example, finding the best combination of available players and tactics in order to maximise the chances of winning.

Key words: machine learning, data mining, sport analytics, neural networks, computer science, data science, forecasting

Resum

Els events esportius, especialment aquells que involucren la competició entre equips, involucren dinàmiques complexes entre els participants que finalmente afecten el resultat final. Predir per endavant el resultat d'aquestos events esportius resulta una tarea complexa precisament per aquestes complexes dinàmiques que apareixen durant l'event, així per les decisions preses per tots els participants. En molts casos, inclús experts humans fallen en les seues prediccions i es produixen resultats sorprenents contra tot pronòstic. La tasca presenta desafiaments des del punt de vista de l'analítica i l'aprenentatge automàtic, ja que és un problema que fins i tot presenta dificultats als experts humans. En aquest TFG estudiem el rendiment de diferents algorismes d'aprenentatge automàtic en la tasca de predir el resultat de partits de futbol professional per intentar predir el resultat final (i.e., victòria local, empat, victòria visitant) sobre la base d'informació exclusivament disponible abans del partit. Per això, es plantejaran diferents tipus de característiques (e.g., característiques agregades, alineacions) i conjunts de dades i se n'estudiarà el rendiment sobre l'exercici dels algorismes d'aprenentatge automàtic, així com també es provaran diferents configuracions per tal de trobar el model que és capaç de predir millor els resultats dels partits de futbol professional. Un cop entrenats aquests models predictius, pensem que poden ser útils en la creació de ferramentes de suport a la decisió per a entrenadors. Per exemple, trobar la millor combinació de jugadors disponibles i tàctiques per tal de maximitzar les probabilitats de victòria.

Paraules clau: aprenentatge automàtic, mineria de dades, analítica esportiva, xarxes neuronals, ciències de la computació, ciència de dades, predicció

Resumen

Los eventos deportivos, especialmente aquellos que involucran la competición entre equipos, involucran dinámicas complejas entre los participantes que finalmente afectan en el resultado final. Predecir de antemano el resultado de estos eventos deportivos resulta una tarea compleja precisamente por estas complejas dinámicas que aparecen en el evento, así por las decisiones que son tomadas por todos los participantes. En muchos casos, incluso expertos humanos erran en sus predicciones y se producen resultados sorprendentes ante todo pronóstico. La tarea presenta desafíos desde el punto de vista de la analítica y el aprendizaje automático, pues es un problema que incluso presenta dificultades a los expertos humanos. En este TFG estudiamos el rendimiento de diferentes algoritmos de aprendizaje automático en la tarea de predecir el resultado de partidos de fútbol profesional con el fin de intentar predecir el resultado final (i.e., victoria local, empate, victoria visitante) en base a información exclusivamente disponible antes del partido. Para ello, se plantearán diferentes tipos de características (e.g., características agregadas, alineaciones) y conjuntos de datos y se estudiará su rendimiento sobre el desempeño de los algoritmos de aprendizaje automático, así como también se probaran diferentes configuraciones con el fin de encontrar el modelo que es capaz de predecir mejor los resultados de los partidos de fútbol profesional. Una vez entrenados estos modelos predictivos, pensamos que pueden ser útiles en la creación de herramientas de apoyo a la decisión para entrenadores. Por ejemplo, encontrar la mejor combinación de jugadores disponibles y tácticas con el fin de maximizar las probabilidades de victoria.

Palabras clave: aprendizaje automático, minería de datos, analítica deportiva, redes neuronales, ciencias de la computación, data science, predicción

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The importance and impact of Big Data and Machine Learning in sports cannot be overstated. The vast amount of data generated by athletes, teams, and fans can be analyzed to gain insights into performance, strategy, and fan engagement. Machine learning algorithms can help coaches and analysts to identify patterns, predict outcomes, and optimize player performance. This information can be used to make data-driven decisions about team composition, game strategy, and fan engagement. The impact of Big Data and Machine Learning is evident in the way that sports teams operate and compete, enabling them to achieve better results and engage fans in new and exciting ways.

In football, Big Data and Machine Learning have transformed the way teams operate and compete, from analyzing player performance to predicting game outcomes. Related to this, the aim of the following report is to describe the study carried out on the application of Machine Learning methods for the prediction of football match results based on the past performance and statistics of the involved teams.

Large amounts of data have been collected during matches by TV channels, football clubs and betting agencies in the last few years. This enables us to undertake the detailed analysis of the players' and teams' performance and, therefore, to attempt the creation of predictive models of their future performance and, consequently, of the results of the matches. Nowadays there are still few research studies on the matter, but increasingly more researchers are deciding to apply their knowledge in the domain of sport analytics.

That is why in this work it has been performed a study of how Machine Learning and Big Data techniques can be applied in the prediction of football results. The predictive models developed predict whether a match will end in a draw, home win or away win. To this end, a set of Machine Learning models have been trained using event data and match statistics from the last seasons of the five major European leagues (Spain, England, Germany, Italy and France).

This document describes the complete process that has been followed, as well as the different techniques that have been used to create a new database, process the data, perform feature engineering, build a baseline model trained with bookmaker odds, and finally build and train some predictive models based on Artificial Neural Networks and Gradient Boosting techniques. Prior to all this, we will offer a revision and critique of the previous relevant related literature.

In order to reach this objective, an iterative spiral methodology has been followed. Starting by proposing simple and basic solutions, and after a subsequent rigorous analysis on the results the experiments, more complex solutions are gradually suggested. Then, these new experiments are evaluated and new solutions are proposed again, bringing the study towards the optimal solution. In this case, we have started training Multi-layer Perceptron networks of few layers with an initial dataset, and, have been implementing new models based on more complex ANNs, regarding both their architecture and their hyper-parameters, while the data have been enhanced using feature engineering techniques.

1.1 Motivations

The reasons that led me to conduct this research have been several. First, my passion for football motivated me to consider doing this work. Since I was a child I have been passionate about football and even more about the statistics and data of teams and players.

Predicting match outcomes can help teams prepare for upcoming matches by adjusting game strategy, training, and player selection. So, it is a critical information football teams would like to have. It allows teams to adjust their game strategy and tactics based on the expected outcome of the match, which can increase their chances of winning. Moreover, it can help teams determine the best players to select for the match based on their skills and performance history and it also helps them allocate resources such as training time, team meetings, and rest periods based on the expected outcome of the match. It is also essential for bookmakers, given that they need predicting match outcomes to set odds, manage risk and ensure a profitable book.

Secondly, from a professional point of view, it was a great opportunity for me to introduce myself into research and Data Science, the field that I want to dedicate to from now on. Also the fact that the scope of Sport Analytics is still seldom studied motivated me. As the world of football is gradually incorporating Big Data and Artificial Intelligence, I think that it can be a professional career path for me to work in since it will bring together both my passions: football and Data Science.

As I said, it was a great opportunity for me to get into an area of Data Science that has not been studied in depth, so I was also encouraged to see it as a challenge. I was conscious of the complexity of the problem, since finding a correlation between the events of a football match and its outcome is no trivial problem given its high component of randomness and the data limitations that I had.

1.2 Objectives

It was difficult to set an initial objective due to the extremely wide-open nature of the study. From the beginning, however, the target was to design and train a model that could predict whether one team - either the home team or the away team - would win or they would draw. But given that objective, the question arose as to whether it is really possible to predict all football matches? And if not, to what level is it possible? In order to answer this question, we took as a reference the predictions made by experts in the field, the bookmakers, to try to train models able to predict football games.

Another goal has been to approach or beat the results of past similar work. This objective is hard to achieve as each work uses a different data source and therefore different data, so it is not straightforward to compare. Nonetheless, we have trained some of the models used in previous works with our data in order to compare and improve our model.

Another question that arose is whether it is possible to identify the factors that define the destiny of a football match. Are there patterns that decide the outcome of a match or is it a matter of luck? If there are, is it possible to find such patterns with the statistics of the games? That's why it is also a challenge to find out the relationships that exist between the events, strategies and statistics that take place in a football match regarding its outcome. It is thus our intention to model these interactions in order to understand the factors that decide the destiny of football.

As for my personal goals, they also include gaining experience and skills in data processing and Machine Learning technologies employed, as well as consolidating theoretical knowledge learned on statistics and Machine Learning over the last six months.

CHAPTER 2

State-of-the-Art

The analysis and study of football patterns and strategies that make a winning team is an activity that has been carried out for more than six decades. Back in the early 1950s, Charles Reep collected by hand statistics which suggested that the key to score goals was to pass the ball as quickly as possible from back to front, which indirectly led to the beginning of the "long-ball movement" in English football. [Reep and Benjamin, 1968].

However, in recent years, technology has made it possible to collect large volumes of data, allowing for a more exhaustive analysis, and thus for algorithms to be designed to extract information and learn from this data. There are three current technologies that allow the collection of the events that take place during a football match: *soccer-logs*, *video-tracking* and *GPS data* [Pappalardo et al., 2019].

- **Soccer-logs** technology describes the events that occur during a match and are captured through proprietary *tagging* software.
- **Video-tracking data** describes the movement of the players during the games and is collected from match recordings.
- **GPS data** describes the players' trajectory during training sessions and is obtained from GPS tracking devices embedded in the players' equipment.

This vertiginous increase in the amount of data has led to the opposite problem. Today, massive amount of data itself has become an obstacle for data analysis, due to a lack of methodological guidelines and theoretical models of tactical decision-making in football [Rein and Memmert, 2016].

Despite the wealth of data generated, these data bases are often difficult to access for scientific research [Pappalardo et al., 2019]. However, this has not prevented studies on tactical analysis or outcome prediction. Many researchers have questioned the relationship between performance and sporting success, whether in football or other sports [Yucesoy and Barabási, 2016]. The conclusion of these studies is that there is a strong relationship in sport between performance and success or failure. However, Pappalardo, for instance, talks about the fact that, while victory or defeat can be explained by performance, it is difficult to detect draws using Machine Learning techniques [Pappalardo and Cintia, 2018, Ulmer et al., 2013, Tax and Joustra, 2015].

In the following sections we will discuss some areas of study related to football analytics. First, we will discuss work related to the analysis of tactics and strategies and how statistical and Machine Learning methods have been applied to the study and identification of these. Then, we will talk about the works focused on the prediction of football match results, how they influence the different events that take place in a match and how they are related to the aforementioned.

2.1 Tactical analysis

One of the main areas of study in football is the analysis of tactics to improve the team's performance. Tactics and strategy, although they are inter-conditioned, are different concepts. Rein and Memmert define them as follows: *"while the team strategy describes the decisions made before the game with respect to how the team wants to play, the tactic is the result of the ongoing interactions between the two opposing teams"* [Rein and Memmert, 2016]. Tactics determine how a team manages space, time and individual actions in order to win the match, and are dependent on the status of the team and the opponent, and on external factors such as who is playing at home or away, the weather or the half-time break.

Automatic detection of a team's tactics and behaviour has been object of research over the last few years. Factors such as "possession directness" correlated with ball possession and passes from the defensive third of the field to the offensive third are important in identifying playing styles [Fernandez-Navarro et al., 2016]. For the study of tactics, many studies have focused on identifying team formations and the distribution of players on the field. Bush et al., for example, investigated the relationship of formation with players' physiological performance and technical skills, and found that players ran longer distances when playing in 4-2-3-1 formations compared to 4-4-2 [Bush et al., 2015]. Another approach, given by Silva et al., is based on analysing numerical superiority in a particular part of the field. This approach resulted in the control of space being a central aspect of tactics [Silva et al., 2014].

Related to this approach is the Team Centroid method, which is based on the geometric centre of a team's player positions, and is used to analyse team behaviour during key match events such as goals [Rein and Memmert, 2016]. Recently, this method has been developed further by calculating the Approximate Entropy (ApEn), a non-linear time-series measurement technique. Goncalves et al. used this method to investigate team coordination between and within defenders, midfielders and attackers. The research showed that movements were more regular with respect to the centroid of their respective subgroups compared to the other groups [Gonçalves et al., 2014]. All these works, however, focus on a very specific aspect, so at the moment it is not clear how team formations interact with individual technique or tactics [Rein and Memmert, 2016].

Two other emerging approaches in football tactics analysis are based on network approaches and Machine Learning methods (Figure 2.1). The core idea of network approaches is to model the players as nodes and the passes as vertices, with weights which represents the number of passes between them. This representation allows to easily find the key players in a team by considering their connections [Gama et al., 2014]. Wang et al. used a Bayesian latent model that was able to automatically identify tactical patterns which, combined with information about successful plays, detected the most effective tactics [Wang et al., 2015]. Machine Learning methods , such as EM, have also been used for automatic formation detection, showing for example that teams tend to use defensive formations during away matches [Bialkowski et al., 2014].

It is, however, in the field of football results prediction where Machine Learning-based methods are most widely used. In the next section we will talk about the works related to

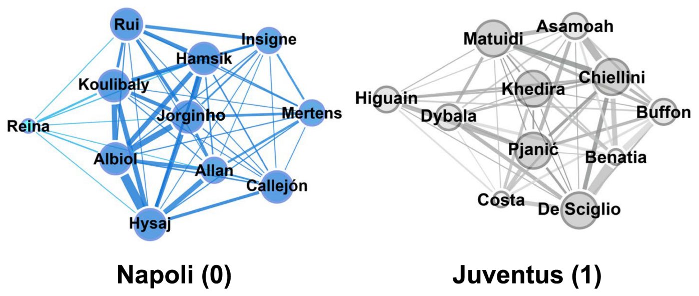


Figure 2.1: Football tactics analysis based on network approaches.

the prediction of football match results. We will talk about how different Machine Learning methods have been applied and which type of data and algorithms have provided the best results.

2.2 Forecasting football matches

Football match prediction is the prediction of the outcome of a football match or the prediction of events occurring during a football match. Regarding the former, the prediction can have different appearances. The prediction can be the result of the match, i.e. how many goals each team has scored, or it can also be a prediction of what will happen in the match, whether it will end in a draw, a home win or an away win. Other approaches try to predict major events during matches, such as whether an attacking situation will result in a goal.

All these predictions are based on data from current or past matches. However, the data can have different forms and nature. Some papers have looked at how to predict the final outcome of a match given the events that are occurring during the match. Other works make their predictions given the previous matches played by both teams and taking into account the statistics of those matches. In addition, the data can be related to team statistics, such as the number of fouls and corners they had in the past; and also to strategy patterns, such as line-up and formation.

Humans are still better at predicting the outcome of a match, as they take into consideration the technical quality of the players as well as the emotional factors that affect the final result. Generally, humans, provided they have knowledge of the match and the teams, have the ability to generalise and summarise this knowledge well enough to make their bets; however, they have a limited amount of information that they can summarise, and their predictions are often influenced by their emotions, which often leads to mistakes [Jain et al., 2021]. The accelerated progress of Machine Learning, and more specifically Deep Learning, over the last ten years has contributed to the increase in research on the development of predictive models of football matches.

Ulmer and Fernandez conducted a study consisting of a 3-class classification problem (home-win, away-win and draw) using Naive Bayes, Hidden Markov models, SVM, Random Forest and OneVsAll SGD [Ulmer et al., 2013] with English Premier League matches (EPL). They used 10 seasons (from 2002-03 to 2011-12) of the EPL for training and other two seasons (2012-13 and 2013-14) for testing. They obtained the date from a historical data-base about football matches called Football-Data[1]. Ulmer and Fernan-

[1] https://www.football-data.co.uk/

dez say the biggest challenges they faced were the large random component of data and the large number of outliers in football, such as Leicester's 2016 title. In terms of their methodology, they first followed a feature selection process based on previous literature and intuition. They selected characteristics such as whether the team played at home or away, the ELO Ranking or the "streakness" of the last n games. Given the teams' streaks, they faced the dilemma of what to do with the first n games. The two approaches they considered were to scale the data or to ignore the first n matches, which gave better results. The optimal number of matches, n^*, was trained as a hyper-parameter of the model in a range between 2 and 7.

The best performing models was the OneVsAll SGD with a classification error in the test of 0.48, and an accuracy in the test of 52% (Figure 2.2). As for the SVM, an RBF-SVM (0.52 error) was used first, but it overfitted, so a simpler model, a Linear SVM (0.49), was tried, which solved the overfitting and allowed more features to be added to the model. Naive Bayes (0.56) and Hidden Markov (0.56) were the worst performing models, in the first case due to the false assumption of sample independence and in the second case due to the model assumption that past states (the result of the matches) are hidden when, in fact, they are known. As can be seen in the Figure 2.2, their models struggled to predict draws accurately, and were also overconfident in predicting away wins. The model that best predicted draws was Random Forest, although it only managed to achieve an AUC score of 0.52 for draws, close to 0.5, which indicates that the model cannot identify draws at all.

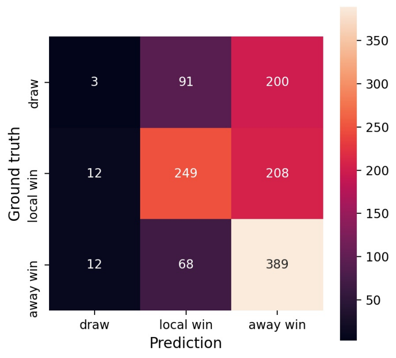


Figure 2.2: OneVsAll SGD confusion matrix in [Ulmer et al., 2013].

More recently, Artificial Neural Networks (ANN) have been used to predict match outcomes. Rudrapal et al. proposed an MLP-based prediction model, and compared it with classical Machine Learning models [Rudrapal et al., 2020]. Their approach consisted in a 2-class classification problem, predicting whether the home team wins or not. Using an MLP model with 10 hidden states, they achieved an accuracy of 73.57%, compared to 72.92% for Random Forest, 58.77% for SVM and 65.84% for Gaussian Naive Bayes. The models were trained on 11,400 Premier League matches, between the 2000-01 and 2015-16 seasons, but they performed feature selection prior to training. The input of the models consisted of features associated with each team, such as attacking, midfield and defensive ability, a team rating and an indicator of the team's streak; features associated with each player, such as their/his rating, potential, market value and indicators of goalkeeper, defence, midfield and attack; and features associated with the match context, such as average home and away points scored, performance over the last five matches and goal difference.

However, it is not only the English Premier League that has been the subject of study, other leagues such as the Dutch league have also been studied. Tax and Joustra, used thirteen seasons of the Dutch Eredivise to train several classification algorithms (Naive Bayes, LogitBoost, MLP, RandomForest, CHIRP, FURIA, DTNB, Decission Tree (J48) and HyperPipes) combined with dimensionality reduction techniques. They approached it as a 3-class classification problem. The best results were using the combination of PCA (with 15% variance) with Naive Bayes or a MLP with 56% of accuracy in the test data (Figure 2.3). The models were trained with a hybrid dataset of betting odds and public data features. It can be observed at Figure 2.3 how their models struggled to predict draws, having over confidence in local wins. However, they do not use Cross-Validation to train their models, due to the temporal nature of the data, and they used the first seven seasons for training and the other six for testing. The high success with Naive Bayes, they say, may be due to the fact that the assumed dependence between features disappears with the application of PCA with low variance. [Tax and Joustra, 2015].

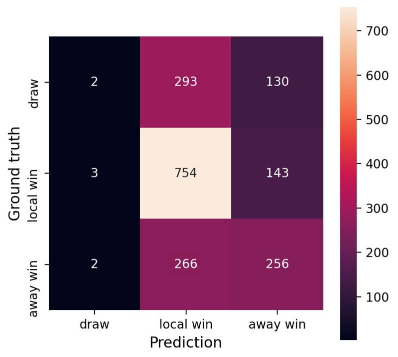


Figure 2.3: PCA + MLP confusion matrix in [Tax and Joustra, 2015]

Another interesting study is the one conducted by Guan and Wang using a very innovative approach, a combination of grey prediction algorithm and extreme learning-machine algorithm [Guan and Wang, 2022]. A grey-box model combines a partial theoretical structure with data and is opposed to black-box models where no theoretical model is assumed, and to white-box models which are purely theoretical. On the other hand, extreme learning machine is a single hidden-layer feed-forward neural network with a completely different learning method from traditional iterative methods. In this model the input weights are randomly generated, while the output weights are obtained by analysis and computation, thus avoiding the difficulties of non-linear optimisation of the input weights[2]. The two models are combined using a function that uses as input the output of the other model. The combination can be linear (equal weighted average, weighted based on the prediction error or based on the covariance) or non-linear (weighted geometric average or weighted harmonic average). [Guan and Wang, 2022].

Nevertheless, most of the recent publications have focused on the application of RNN and LSTM models to predict the outcome of football matches. A success rate of 80% has been achieved with LSTM models. [Jain et al., 2021]. LSTM models are RNN models that are designed to remember both the oldest and the most recent values. If the model decides that a feature is important for deciding the output, it stores that feature (the model remembers it) for a longer period of time [Jain et al., 2021]. In this work, the approach was to infer the best features from the results of previous matches, so they made use of an LSTM model trained on data from the English league between 2010 and 2018. For each match the data included the goal difference of each team and data generated from other

[2]read more about Extreme learning machine here: https://link.springer.com/article/10.1007/s11042-021-11007-7

attributes, such as the generation of goals scored by each team, the streak of each team in the last 3 and 5 matches or the point difference of each team.

A very different approach but also based on LSTM networks was done by Pettersson and Nyquist. The paper describes their study of different possible formats for network input [Nyquist and Pettersson, 2017]. They trained the LSTM model with data from 63 different countries, and the data consisted of events that occurred during matches, such as goals, fouls or penalties. To use this data as input for their model they had to transform the events so that they all had the same size. The options they propose are, in the first place, to use Deep Embedding, inspired by *word2vec*, which work very well for NLP; in the second place, One-Hot Vector, with all attributes of all events, which means that for each event there are many empty columns as all attributes except player and team identification are one-hot encoded; the last option is Concatenated Embedding Vectors for all attributes, a slight variation of the one-hot vector, as we use one embedding for each event type and one lookup for all values in the attribute vector and concatenate the resulting embedding. The format finally chosen was the latter, and each sample entered in the model is an event of a match. The input consists of a vector of ten variables. These variables are then transformed either to a one-hot encoding for each attribute and then concatenated, or to an embedding lookup for each feature and then concatenated to form a larger vector [Nyquist and Pettersson, 2017, Rahman et al., 2020].

In this paper two different approaches are also applied to calculate the error during training. In the first one, called "Many-To-One", the error is only calculated once, in the last event of the match to predict the outcome, and then used for the BP. The second, called "Many-To-Many" or *sequence loss*, is an average of the error over the entire sequence, calculating the error for each match event and averaging it before using it in the BP [Nyquist and Pettersson, 2017, Rahman et al., 2020]. During the training and evaluation of the model they propose seven case studies, the first six are LSTM models with different architecture (different number of hidden layers and LSTM units) but all of them use the "Many-To-Many" error, while the last case study uses "Many-To-One". In addition, in order to evaluate and validate the models, they predict the final score for each match every fifteen minutes from minute zero. Up to minute 45 the best results are obtained by a model with 1 hidden layer and 256 LSTM units using "Many-To-Many", with accuracy results of 44% (min 0), 45% (min 15), 47% (min 30) and 52% (min 45). From min 45 to the end the best model is the one using "Many-To-One" with 2 hidden layers and 256 LSTM units, with results of 63% (min 60), 74% (min 75), 88% (min 90) and 98% (at the end of the match). Following this work, Rahman conducted a study to predict the outcome of the FIFA World Cup 2018 matches using the same architecture, input format and error functions with a success rate of 63% [Rahman et al., 2020].

Other approaches have been taken to attempt to predict high-impact events during matches or the results of football matches, such as by analysing the sentiment of tweets posted during the match [Godin et al., 2014, Yu and Wang, 2015]. Also, statistical models have been developed to predict the results [Koopman and Lit, 2015], as well as hybrid models based on more than one algorithm that increase the overall accuracy of the system [Guan and Wang, 2022, Jain et al., 2021].

Another interesting question is to keep in mind what the purpose of these models is. Some of these models are evaluated by comparing them with bookmakers' odds, also with the aim of checking whether the model can compete with the bookmakers and make money. However, in order to do so, it is necessary to take into account which matches to bet on. For example, there will be odds with a threshold where, even if the model pre-

reference	data	model	classes	acc.
[Tax and Joustra, 2015]	13 seasons, odds + features	PCA + Naive Bayes, PCA + MLP	3	54.7
[Rahman et al., 2020]	World Cup 2018 10 embedded feats.	LSTM 2 layers, 10 & 5 units	3	63.3
[Jain et al., 2021]	2010-2018 EPL past matches features	LSTM 1 layer, 64 units	2	80
[Rudrapal et al., 2020]	11,400 EPL matches 20H-20A features: team, player and head-to-head features	MLP 1 layer, 10 units	2	73.57
[Guan and Wang, 2022]	Not specified	Grey prediction method + Extrene Learning Machine	2	over 80
[Ulmer et al., 2013]	Train: 10 EPL sns. Test: 2 EPL sns. match features and time-dependence	NB, Markov, SVM (lin. & RBF), RF and SGD (best). Underpredic. draws	3	52
[Nyquist and Pettersson, 2017]	Embedded events of 63 leagues Note: got underpredicted draws	LSTM M-to-1 1 layers 256 units	3	min 0: 44
		LSTM M-to-M 2 layers 256 units		end: 98

Table 2.1: Results of related works

dicts a win for a team, the odds are so low that it will not be worth betting on the match at all [Bunker and Thabtah, 2019]. Models have been proposed that manage to beat the bookmakers and achieve a certain profitability [Koopman and Lit, 2015, Tax and Joustra, 2015]. Godin et al. claim that with their model they have been able to make a profit of 30% in matches in the second half of the 2013-2014 English Premier League season [Godin et al., 2014].

2.3 Critique

It is convenient to make a critical analysis of the current state of the scientific study of football before presenting a proposal. As mentioned above, the analysis and study of football and its patterns and strategies, although it has been carried out for several decades, only recently has it become more common and has been updated to more up-to-date Machine Learning techniques.

However, and probably as a consequence of this recent boom, there are still no precise methodological guidelines for this field of study, and each researcher, in the lack of references, must define their own methodology. Related to this, the areas of study within sports analytics and sports forecasting based on Machine Learning methods are barely well-defined and most of them are of very recent creation. This is why many studies have a lack of references and are independent studies between which there is no flow of information whatsoever, which means that many studies end up covering similar topics or proposing similar solutions.

Another problem is the secrecy of a lot of projects in sports analysis. This is due to the fact that most high-level projects are carried out by private entities with strong interests in football. This is the case for football teams, which are increasingly performing advanced studies at all levels on the performance of their teams and their opponents. There are also studies, slightly more available to the public, by private entities such as the big media and producers of sports audiovisual content, but they are still quite discreet in their publications. This means that most of the latest work does not reach the academic field, so conducting research in this field is not straightforward.

Closely related to this is the lack of open-access databases. It is impossible to draw solid conclusions without data. The fact is that there are very few available open-access football databases, and those that are available contain very little data or offer little variability of information that focuses on mere general match statistics. In order to access more complex data such as match events, player positions or team formation, databases must be purchased and a considerable licence fee must be paid to access them. The fact that there are no standardised datasets for benchmarking predictive models means that comparing models against each other is not straightforward. The use of such benchmarks would help to improve the State-of-the-Art.

There is a lack of studies applying cutting-edge Machine Learning methods, most likely due to the issues mentioned so far. As a consequence of the lack of accessible data and related previous studies, many researchers decide to apply well-known - and not very new - Machine Learning methods in order to perform well, such as Logistic Regression, SVM or Random Forests. Few papers decide to make use of techniques and methods that have never been applied to football match prediction before.

Regarding the content of the papers related to the forecasting of results, most of them focus on building models that are capable of predicting match results and evaluate them with the overall accuracy or error that they have. However, many of the papers describe how the error distribution of matches is not uniform and happens mostly in matches that end in a draw, in matches that are, a priori, evenly matched, or in matches with a very unexpected outcome. Few studies focus on these more critical and complex matches.

An additional aspect that is rarely treated in most works is the explanation of the factors that influence the result of a match. Emphasis is put on the random factor as an essential piece in the puzzle; however, not many conclusions have been taken regarding the influence of tactical or strategic factors such as the line-up, the level of competitiveness or the absence of key players in a match.

2.4 Possible solutions

In this research work, in order to achieve the proposed objective, some of the aspects previously discussed in related works will be addressed. To do so, useful ideas and results from some previous works will be used as a starting point and new methods and ideas not previously proposed will be introduced.

The solution proposed in this dissertation starts with the creation of a dataset from several open-access data sources. This dataset should contain as many matches as possible, and for each of them as much information as possible. The idea is that each match should contain information on different historical and strategic aspects. Based on these characteristics, an analysis and experiments would be done to evaluate which Machine Learning algorithms and which characteristics would give the best results.

When evaluating the results, the goal is to obtain good general results, but especially to improve the accuracy of the model in the most difficult matches to predict, the draw matches. To do so, the results will be evaluated by applying different appropriate metrics to measure the accuracy in multi-class classification problems. In the experiments, dataset re-balancing and weighting techniques will be applied in order to give more importance to the matches that end in a draw so that the model is able to learn better the relationships that occur in these matches; regularisation methods will also be applied so that the ML algorithms do not overfit the training data and generalise correctly when new data is provided to them. During the experiments, we will optimise the configuration parameters of the algorithms so that they perform as we want them to.

Examples of Machine Learning algorithms that we will use for prediction are weak classifiers as baseline, such as decision trees or SVM, trained with bookmakers' odds. Later, in the experiments, more sophisticated algorithms will be used for training, namely ANN, with different architectures and complexity, and the gradient boosting algorithm, LightGBM, which is widely and successfully used with tabular data.

CHAPTER 3 Methodology

Defining a methodology is crucial when conducting a research work, especially in the field of Sport Analytics and forecasting of football results. It provides a structured and systematic approach to collecting, analyzing, and interpreting data, which ultimately ensures the reliability and validity of the research findings.

In the context of Sport Analytics and forecasting of football results, a well-defined methodology helps researchers to identify the key performance indicators (KPIs) that are most relevant to predicting match outcomes. It also enables them to choose appropriate statistical models and techniques for analyzing the data, such as regression analysis or machine learning algorithms. Additionally, a clearly defined methodology helps researchers to communicate their findings effectively, allowing other researchers to replicate the study and build upon the results. Overall, a well-defined methodology is essential for ensuring the rigor and credibility of research in Sport Analytics and forecasting of football results.

The methodology we have followed to carry out the work has been the following:

1. Review of previous related work.
   (a) Notation of the designed models and results obtained and/or conclusions reached in each work.
   (b) Record of the highlights of each paper, to be possibly used later in this document.
   (c) Brainstorming and design of a possible predictive model architecture based on previous literature.
2. Collection and generation of datasets.
   (a) Search and gathering of free and open soccer databases available on the Internet.
   (b) Inspection and analysis of the data content of each database so as to identify which ones are compatible with each other, provide a reasonable amount of data and can be used to train our model.
   (c) Creation of the bookmaker odds dataset from a database that contains the betting odds of the matches.
   (d) Creation of datasets from multiple compatible databases, merging datasets and generating new variables from existing ones. This step consists of a long-term work of programming, creativity and exploratory analysis of data.

3. Training of the baseline model using the bookmaker odds.
4. Data analysis
   (a) In-depth statistical analysis of the dataset.
   (b) Dimensional reduction and feature selection based on the data analysis.
5. Iterative process of experiments approaching progressively to the optimal model (Figure 3.1).
   (a) Study of the solution proposed for the problem, carrying out a research if necessary. In the first iteration, starting with a simple solution.
   (b) Implementation of the solution.
   (c) Experiments with the model, data and configuration proposed.
   (d) Rigorous analysis of the experiment results, focusing on the results and the performance of the model, especially analysing its learning metrics.
   (e) Comparison of the results with the previous experiments.
   (f) Summary of conclusions and brainstorming of possible enhancements of the model and the data. Back to a) if possible improvement considered.
6. Final analysis of optimal model.
   (a) Detailed analysis of final results and comparison with baseline models and related works results.

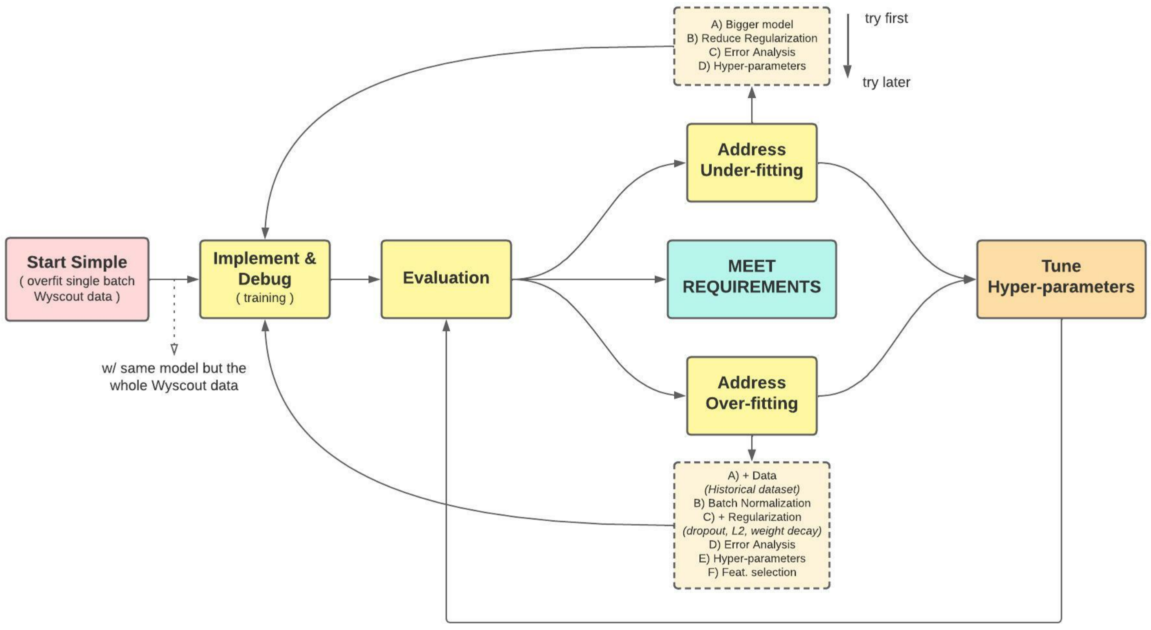


Figure 3.1: Pipeline diagram of the methodoly used during the experiments phase in order to state the decission making process when training and evaluating a model.

CHAPTER 4

Thoretical background

4.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a type of computational model inspired by the structure and function of the human brain. ANNs consist of a large number of interconnected processing nodes or "neurons" that work together to solve complex problems.

These networks are able to learn and generalize patterns from data, making them a powerful tool for solving a wide range of problems, including classification, regression, prediction, and decision-making tasks. ANNs have been successfully applied in fields such as image and speech recognition, natural language processing, computer vision, robotics, and finance, among others.

ANNs are particularly well-suited for solving problems that are difficult to model using traditional statistical or algorithmic techniques, such as those involving non-linear relationships, complex interactions, or noisy or incomplete data. They can also be used for tasks where the underlying relationship between inputs and outputs is not well understood, and the goal is to learn a function that maps inputs to outputs based on examples provided in a training set.

There are many types of Artificial Neural Networks (ANNs), each with its unique architecture and application. Multi-Layer Perceptron (MLP) is one of the most popular types of ANNs, and it is widely used in various machine learning applications, such as in problems that consider tabular data.

4.1.1. Multi-layer Perceptron

There are many types of Artificial Neural Networks (ANNs), each with its unique architecture and application. Multi-Layer Perceptron (MLP) is one of the most popular types of ANNs, as it is a powerful tool for tabular data processing and they are also commonly used in applications such as classification, regression, and time series prediction, among others.

An MLP (Figure 4.1) is a type of feedforward neural network consisting of multiple layers (l) of neurons. The first layer is the input layer (x_n), which receives the input data. The output layer (y) produces the final output, while the layers between the input and output layers are called hidden layers. Each neuron in the input layer connects to each neuron in the first hidden layer, and each neuron in the hidden layer connects to every neuron in the following layer until the output layer is reached.

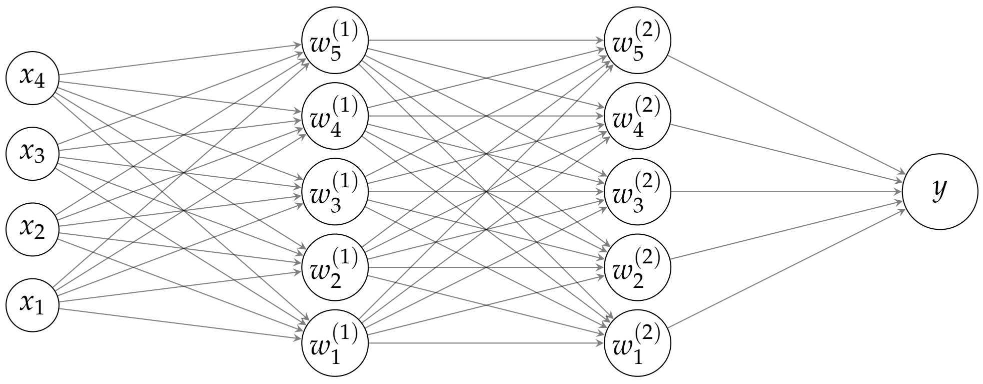


Figure 4.1: Architecture of a MLP with an input layer with 4 neurons, 2 hidden layers with 5 neurons each and the output layer with 1 neuron.

The connections between the neurons (n) in the MLP are weighted (w_n^l), and the network learns to adjust these weights during the training process. Each neuron in the MLP receives an input, applies a linear transformation to it, and passes it through a non-linear activation function such as the sigmoid or ReLU. This activation function introduces non-linearity into the network. If MLPs used linear activation functions, the output of the network would be a linear combination of the input features, regardless of the number of layers in the network. This would limit the network's ability to capture complex and non-linear relationships between the input and output variables. Moreover, the non-

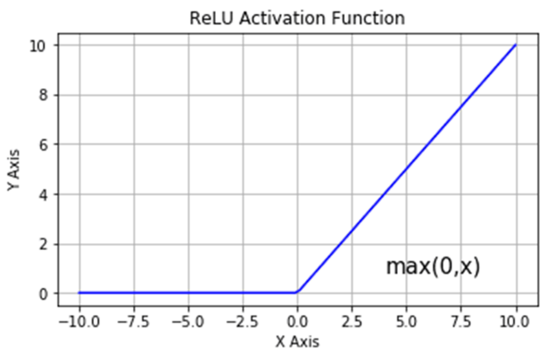


Figure 4.2: ReLU non-linear activation function.

linear activation functions introduce the property of "universal approximation" in MLPs. This means that MLPs with a single hidden layer and a sufficient number of neurons can approximate any continuous function to arbitrary accuracy. This property makes MLPs very powerful and effective for a wide range of machine learning problems.

MLPs are trained using a process called backpropagation. This process involves the following steps:

1. **Initialization of weights and biases**: the weights and biases of the MLP are initialized randomly.

2. **Forward propagation**: input data is fed into the input layer of the MLP, and the output is computed using the current weights and biases. The output is then compared to the target output to calculate the error.

3. **Backpropagation**: the error is propagated backwards through the network using an optimization algorithm, such as gradient descent. The algorithm adjusts the weights and biases of each neuron in the network to minimize the error.
4. **Update weights and biases**: the updated weights and biases are used in the next iteration of the forward propagation step.
5. **Repeat**: steps 2-4 are repeated for a number of epochs until the MLP has learned the patterns in the training data.
6. **Validation and testing**: the performance of the trained MLP is evaluated using a validation set, and the final performance is measured on a testing set.

In the next part, the Backpropagation algorithm will be explained in more detail.

4.1.2. Training

MLP neural networks are trained and optimized using a process called backpropagation. Backpropagation is an algorithm that uses the chain rule of calculus to calculate the gradients of the network's parameters with respect to a loss function, which measures the difference between the network's predicted output and the true output. The gradients are then used to update the network's parameters and , such as the weights and biases of the neurons, using an optimization algorithm such as gradient descent. Gradient descent is a first-order optimization algorithm that minimizes the loss function by iteratively adjusting the parameters in the direction of the negative gradient (Figure 4.3).

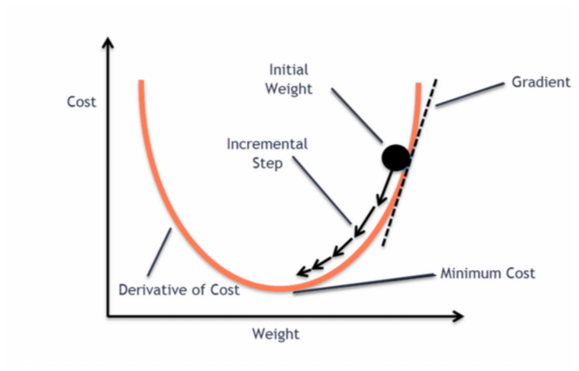


Figure 4.3: Convex loss function optimized with Gradient Descend. We can see how the incremental step movement caused by the gradient descend moves the weights of the model towards the minimum point of the loss function.

Let's assume we have a MLP with L layers, where the input to the network is denoted as x, the output is denoted as y, and the weights and biases of the network are denoted as w and b, respectively. The forward propagation equation for a layer l is given by:

$$z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]} \tag{4.1}$$

$$a^{[l]} = g^{[l]}(z^{[l]}) \tag{4.2}$$

where $z^{[l]}$ is the weighted sum of inputs to layer l, $a^{[l]}$ is the activation of layer l, $w^{[l]}$ and $b^{[l]}$ are the weights and biases of layer l, and $g^{[l]}$ is the activation function of layer l. The error function is given by:

$$E(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2}\sum_{l=1}^{L} |w^{[l]}|^2 \tag{4.3}$$

where m is the number of training examples, $L(y^{(i)}, \hat{y}^{(i)})$ is the loss function, $\hat{y}^{(i)}$ is the predicted output for the ith training example, λ is the regularization parameter, and $|w^{[l]}|^2$ is the squared of the norm of the weights of layer l. One calculated the error the optimization and update of the weights starts computing the gradients with the backpropagation algorithm. The backpropagation equations for a layer l are given by:

$$\delta z^{[l]} = \frac{\partial E}{\partial z^{[l]}} = \delta a^{[l]} \cdot g^{[l]\prime}(z^{[l]}) \tag{4.4}$$

$$\delta w^{[l]} = \frac{\partial E}{\partial w^{[l]}} = \frac{1}{m}\delta z^{[l]} a^{[l-1]} + \frac{\lambda}{m} w^{[l]} \tag{4.5}$$

$$\delta b^{[l]} = \frac{\partial E}{\partial b^{[l]}} = \frac{1}{m}\sum_{i=1}^{m} \delta z^{l} \tag{4.6}$$

$$\delta a^{[l-1]} = \frac{\partial E}{\partial a^{[l]}} \frac{\partial a^{[l]}}{\partial a^{[l-1]}} = w^{[l]} \delta z^{[l]} \tag{4.7}$$

where $\delta z^{[l]}$ is the derivative of the error function with respect to $z^{[l]}$, $\delta w^{[l]}$ and $\delta b^{[l]}$ are the gradients of the error function with respect to the weights and biases of layer l, respectively, $\delta a^{[l-1]}$ is the derivative of the error function with respect to the activation of layer $l-1$, and $g^{[l]\prime}$ is the derivative of the activation function of layer l. The notation $\cdot$ denotes element-wise multiplication, and (i) denotes the ith training example.

Finally, the update equations for gradient descent are given by:

$$w^{[l]} = w^{[l]} - \gamma \delta w^{[l]} \tag{4.8}$$

$$b^{[l]} = b^{[l]} - \gamma \delta b^{[l]} \tag{4.9}$$

where γ is the learning rate, $\delta w^{[l]}$ and $\delta b^{[l]}$ are the gradients of the error function with respect to the weights and biases of layer l, respectively.

During training, after the forward propagation (Eq. 4.1), the output of the network is compared to the true output using the loss function (Eq. 4.3). The gradients of the loss function with respect to the network's parameters are then calculated using backpropagation (Eq. 4.4), and the parameters are updated using gradient descent (Eq. 4.8). This process is repeated for multiple iterations, or epochs, until the network's performance on a validation set reaches a satisfactory level. In addition to gradient descent, other optimization algorithms, such as Adam and Stochastic Gradient Descend, can also be used to optimize the MLP's parameters. Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam) are both optimization algorithms commonly used to train artificial neural networks, including multilayer perceptrons (MLPs).

Stochastic Gradient Descent is a variant of gradient descent that randomly selects a subset of the training data, known as a mini-batch, to calculate the gradient of the loss function. The gradients are then used to update the network's parameters, such as the weights and biases of the neurons. This process is repeated for multiple mini-batches until the entire training set has been processed, constituting one epoch. SGD is more computationally efficient than regular gradient descent and is less prone to getting stuck in local optima.

Adam is an adaptive optimization algorithm that combines ideas from both SGD and root-mean-square propagation (RMSprop). It computes adaptive learning rates for each parameter based on the historical gradients, allowing the algorithm to converge faster and more reliably than SGD. Adam also includes a bias correction term that corrects the bias introduced in the early stages of training when the estimated moments of the gradients are far from their true values. Adam is often used as a default optimization algorithm in many deep learning frameworks.

During the backpropagation phase of training Multi-layer Perceptron (MLP) neural networks, the gradients can suffer from two common problems: vanishing gradients and exploding gradients.

- **Vanishing gradients** occur when the gradients become very small, making it difficult to update the weights and biases of the lower layers of the network. This can cause the lower layers to become unresponsive to changes in the input, leading to poor performance. Vanishing gradients are often caused by the use of activation functions such as sigmoid or hyperbolic tangent, which have very small derivatives when their inputs are far from zero. To mitigate this problem, alternative activation functions, such as ReLU, have been developed that have larger derivatives and can prevent gradients from vanishing.

- **Exploding gradients**, on the other hand, occur when the gradients become very large, causing the weights and biases to be updated too drastically. This can lead to instability in the training process and cause the network to diverge instead of converging to a solution. Exploding gradients are often caused by numerical instability in the optimization algorithm or the network's architecture, such as very deep networks.

One of the phenomenons that can cause one of this gradient problems is Internal Covariate Shift (ICS), where the distribution of the inputs to each layer of a multilayer perceptron (MLP) neural network changes during training, causing the network to converge more slowly. This change in the distribution of the inputs to each layer can create a mismatch between the statistics of the inputs to each layer and the statistics of the activations, which can slow down the convergence. Normalization techniques, such as Batch Normalization, can help mitigate ICS by normalizing the inputs to each layer to have zero mean and unit variance before passing them to the activation function. This helps reduce the impact of ICS by making the statistics of the inputs to each layer more consistent and leading to more stable and efficient training of MLP neural networks.

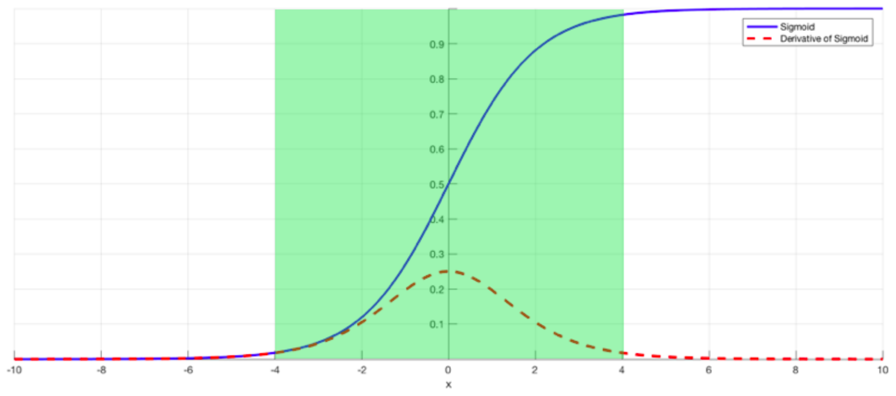


Figure 4.4: Sigmoid function and its derivative. We can see within the green box the non-saturate space of the function. Inputs outiside this space will saturate the function and will bring the derivative to 0.

4.2 Gradient Boosting algorithms

In machine learning, a weak learner is a model that performs only slightly better than random guessing on a given task. While a weak learner may not be accurate on its own, it can be combined with other weak learners to form a strong learner through techniques such as boosting.

Gradient boosting algorithms are a family of machine learning algorithms that are designed to create predictive models by iteratively combining weak learners to form a strong learner. The basic idea of gradient boosting is to train each new model to correct the errors made by the previous models, using a gradient descent optimization process.

LightGBM is a gradient boosting algorithm that is designed to be both highly efficient and highly accurate. It uses a histogram-based approach to binning the data, which allows for faster training times and lower memory usage than other gradient boosting algorithms. Additionally, LightGBM uses a technique called Gradient-based One-Side Sampling (GOSS) to reduce the number of samples used in each iteratio.

The basic idea of GOSS is to use a combination of gradient information and random sampling to identify and prioritize the most informative samples for each iteration of the algorithm. The training data is first sorted based on the absolute value of the gradient of the loss function with respect to the predicted value. Then, a small fraction of the samples with low gradients are randomly sampled, while the remaining samples with high gradients are used in each iteration. By doing this, GOSS prioritizes the samples that are more informative for the model while still including some randomly selected samples to prevent overfitting.

One of the key features of LightGBM is its ability to handle large datasets with high-dimensional features. It uses a technique called Leaf-wise Tree Growth, which grows the tree in a leaf-wise manner instead of level-wise, resulting in fewer leaf nodes and a shallower tree. This can help to reduce overfitting and improve the speed of the training process

4.3 Underfitting vs Overfitting

Underfitting and overfitting are two common problems in machine learning that can affect the performance of models, including MLP and Gradient Boosting algorithms.

Underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and test data (Figure 4.5). In the case of MLP and Gradient Boosting algorithms, underfitting can occur when the model is too small or not trained for enough iterations, resulting in insufficient learning of the data patterns.

Overfitting occurs when a model is too complex and memorizes the training data instead of learning the underlying patterns, resulting in high performance on the training data but poor generalization to new data (Figure 4.5). In the case of MLP and Gradient Boosting algorithms, overfitting can occur when the model is too large or trained for too many iterations, resulting in excessive learning of the data patterns.

To address underfitting, one can increase the model's complexity, such as increasing the number of layers or trees in the case of MLP and Gradient Boosting algorithms, respectively. To address overfitting, one can reduce the model's complexity, such as reducing the number of layers or trees or applying regularization techniques like weight decay or dropout.

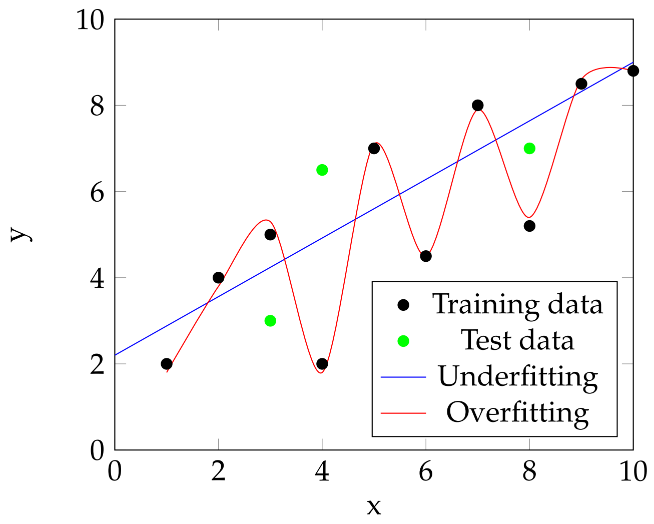


Figure 4.5: Visualization of underfitting and overfitting of a regression curve. The data points are the training data points.

Both underfitting and overfitting can be identified and mitigated through techniques like cross-validation, which involves splitting the data into training and validation sets to assess the model's performance on unseen data. Cross-validation is a technique used in machine learning to evaluate the performance of a model by testing it on a set of data that was not used during training. The idea behind cross-validation is to divide the available data into multiple subsets, called folds, and then train and test the model on each fold while holding out the remaining folds for validation. Cross-validation can also help to find the optimal hyperparameters for the model, such as the number of layers, neurons, and trees, to balance the trade-off between underfitting and overfitting.

CHAPTER 5
Implementation

In this chapter, we will discuss the technologies used to carry out this work. We will begin by explaining the technologies used for each part, giving the reasons why we have opted for one technology over another. We will explain the technology employed at different levels, such as infrastructure level and functional level of analysis and experiments. Finally, we will describe the framework infrastructure that we have created to enable easy and reproducible dataset creation, experimentation, and analysis of results, all following a systematic, clear, and straightforward process.

5.1 Chosen technologies

Different tools and technologies have been used to carry out this research work. Each one has been used for a specific reason and context, some have been used throughout the whole work and others have been necessary only for a part of it. All of them, independently and together, have formed the working environment.

The work has been mainly undertaken on a personal computer, on which all the necessary software has been installed and configured. Windows has been the main Operating System I have worked on, but the experimental phase was carried out on a virtual machine provided by VRAIN[1] running on a Linux system.

Python has been the programming language used, and it was chosen because it is currently the most used language in Data Science projects and, therefore, it has many advantages when you use it as there is more documentation and more libraries available. Along with Python, a virtual environment manager for Python, called Anaconda, has been used to create an independent environment for this project with the libraries and their respective versions, without affecting the versions of other environments in other projects.

Anaconda has made it possible, in a simple way, to use the same Python environment on both Windows and Linux. The parts of data analysis and exploration, dataset generation, proof of concept and analysis of results were carried out on Windows. For these tasks we used Jupyter Notebooks integrated with Visual Studio Code. Jupyter Notebooks is an application that allows you to implement and run code by execution cells, so it is very useful for analytical tasks or code proofs, and it also permits to take notes in Markdown and Latex language.

[1] https://vrain.upv.es/

However, Notebooks are not the most suitable tool for running experiments, so the necessary code was implemented in a self-made Python library to have an automated pipeline in which the data is preprocessed, the training is executed and the results are generated. This pipeline is executed by receiving as input the configuration and parameters of the experiment. At the end, the different results of each configuration are generated, as well as running logs to verify that it has worked correctly and to analyse its performance. Finally an Excel file is generated to summarise the whole experiment. All this pipeline is also made to be executed in the Linux BASH terminal, passing the configuration through parameters in the terminal call or through a JSON configuration file. We have also used TMUX, a multiplexer for UNIX systems that allows a terminal to be divided into multiple independent sessions and thus be able to run several experiments in parallel on a single terminal.

Some of the Python libraries that have been used throughout the work have been NumPy, an open source library that supports the creation of vectors and multidimensional arrays as well as providing a large collection of high-level mathematical functions to operate with them; and Pandas, a library specialised in the manipulation and transformation of data in the form of tables, called Dataframes. Both libraries are generic libraries that are widely used for data analysis and manipulation, and therefore have a large documentation and support community behind them. They have been used both in the database creation stage, cleaning and transforming the original data and creating new features, as well as for data analysis and exploration and data processing before and during model training. In addition, Pandas has been used for more in-depth analysis of the results.

On the other hand, there are libraries that have been used for more specific functions. This is the case of the graphic generation libraries, Matplotlib and Plotly. These two libraries have been used to visually analyse data from the matches as well as data about the performance of the prediction models, such as the learning curve. Matplotlib has been used more for data analysis, as well as for creating the graphical content shown in this document. On the other hand, Plotly, which provides interactive graphics, has been used more for the analysis of the data in a more interactive and comfortable way.

The main Machine Learning library that has been used is Scikit-Learn, which provides a wide palette of functions for data preprocessing and scaling, which have been used during data preparation before training or for feature selection, such as with ANOVA or Variance Threshold. This library has also been used for Baseline models, as Scikit-Learn offers an API for classical ML algorithms such as SVM or Decision Trees. We have also used Scipy, a library that offers a collection of mathematical functions and algorithms similar to Matlab, which has been used mainly during the data analysis phase.

Two libraries have been used to implement the predictive models, each one specialised in a different Machine Learning algorithm. On the one hand, PyTorch has been used for the ANN-based models. This library is one of the two most widely used libraries for ANN, and it was decided to use PyTorch rather than the other, Tensorflow, because we already had some academic experience with PyTorch. On the other hand, for predictive models based on Gradient Boosting models, the LightGBM library has been used, which implements an efficient version of this algorithm and has recently outperformed other versions.

Finally, for the optimisation of the hyperparameters of the ANN and GB models, we have used Optuna, a framework that automatically performs a Bayesian search for hyperparameters by simply defining and passing an objective function to be minimised or maximised.

In order to be able to carry out all this work, not only did we need the appropriate technology to carry it out, but also an infrastructure and a software platform on which to conduct the entire workflow. In the next section we will talk about how this infrastructure has been implemented so as to ease the work undertaken.

5.2 Development platform

In this section we will describe the software platform on which we have carried out all our work. Each part of the total has required specific tools and infrastructure, so in many cases this infrastructure has been implemented as we have reached that part.

First of all, as we have already mentioned in the previous section, the data analysis part has been carried out entirely in Jupyter Notebooks, because it is more suitable for this kind of exploratory tasks that require a lot of creativity to perform, and the fact of running the code in parts (cells) makes it much easier.

The generation of the datasets has also been done with Notebooks. This is due to two main reasons. On the one hand, Notebooks are a really convenient tool for testing or developing code by monitoring the state of execution as it is being executed and developed at the same time. In the case of dataset generation, in which many transformations and filters are applied to the data, it is very interesting to see at which point the execution is in order to validate the code. The second reason is that the generation of the dataset only takes place once, since the dataset is invariant, except if we want to modify some characteristic, for which we would go to the Notebook. If we wished to add data to the dataset periodically in an automated way, we would have been interested in moving the code to other more convenient code formats.

Finally, the experiments have been the most complex part to implement. Unlike the dataset that is run once and that's it, the experiments are a part that you want to run multiple times, with different configurations and maintaining reproducibility when running the same configuration. Reproducibility in Data Science experiments implies that the results obtained must be able to be replicated using the same data, code and parameters at any time. To ensure this reproducibility, a seed is used, which is an initial value used in the generation of pseudo-random numbers to guarantee that the results are reproducible. By setting a seed, the same sequence of pseudorandom numbers can be obtained in different executions of the code.

In addition, the experiments, as well as receiving an input, which would be the configuration and data, return an output, which are the results and logs of the execution. These experiments must be able to be executed on both local and remote machines, even if they have different operating systems.

To set up the compatible infrastructure on both systems, the anaconda virtual environment has been cloned from the local machine to the remote one, and the paths from which files are read and written have been translated on both systems, dynamically selecting them using a system variable that indicates their type of system. On the other hand, the file transfer protocols SCP (Secure Copy) and SSH (Secure Shell) have been used to send files and launch executions on the remote machine.

When carrying out an experiment, we do not want to waste a lot of time indicating its parameters to the execution, nor later generating and searching for its results. That is why this whole process has been implemented to be simple, monotonous, error-avoiding and efficient.

The parameters of the experiment are indicated through a YAML file where there are several fields: Data, Model and Results. In the Data field, you indicate which dataset you want to train the model with, how many features you want the model to train - indicating which features you want to keep, which ones you want to delete and which dimensionality reduction technique you want to apply - and other parameters such as the scaling method to apply to the data. In the Model section, you indicate which algorithm you want to run with, which type of optimisation you want to train the hyperparameters (Random search or Bayesian search). It is also possible to manually set hyperparameters to specific values in case you want to analyse a specific configuration. In the Results section we can indicate if we want to save the model predictions, as well as the logs and metrics. We also indicate the path in which we want it to save as well as the possibility of indicating some metrics that we want it to calculate, such as the F1-score or to group different runs and calculate grouped statistics (such as the performance of a hyperparameter in different experiments).

In order to execute the experiments, two libraries have been created, one for each algorithm, ANN and LightGBM, which are invoked from another file that is being executed with the Bash terminal. In the ANN library, functions have been declared to train and validate a model, to train with Cross-Validation, the function that performs the different experiments of a hyperparameter optimisation process (RS or GS) has been implemented and logging functions have also been implemented, both some using native PyTorch functions such as hooks to log the performance of a model (gradients and activation functions) and others to log the error and the error in each epoch. In the LightGBM library, Bayesian search training has been implemented with Optuna by defining a function to optimise (minimise error) given a model metric.

All these results and logs are written in a single, equally structured directory for each experiment, allowing multiple results to be read in the same way. In addition, after the execution of a set of experiments, a script is run in which an Excel is created summarising all the experiments done and giving an overview of how the experiments went.

These results are then transferred to the local machine where, with a Jupyter Notebook, they are analysed with the generation of dashboards or the exhaustive analysis of predictions.

CHAPTER 6
Data engineering

A fundamental part of building any type of predictive model is the training data. This chapter details the process by which the data is obtained, processed, analysed and selected in order to feed the model with the most appropriate and necessary information to obtain better results both in training and in inference.

6.1 Data sources

Following an exhaustive search on the Internet, two datasets were finally used for the generation of the training data.

6.1.1. Football-Data

The first dataset was taken from the Football-Data website[1], which has data for all the matches of the major European and American leagues, as well as other more exotic European leagues, such as the Greek, Finnish or Swiss leagues. It contains results, statistics, bookmaker odds and other features such as the date of the match. For this analysis, data from the five major leagues - English, Spanish, Italian, German and French - from the 1993-1994 season to the 2021-2022 season was used. Table 6.1 shows the main attributes available for each party in this dataset. Attributes with the $*$ symbol are not available in all leagues, and will therefore be omitted for the generation of the dataset. Some attributes omitted have been odds from other bookmakers and other types of betting odds, such as the Asian Handicap.

This dataset has been employed in some of the related works we have talked about in the chapter of State-of-the-Art. These works are [Rudrapal et al., 2020] and [Ulmer et al., 2013], both using only games from the English Premier League.

6.1.2. Wyscout

The data for the second data set, obtained from the work [Pappalardo et al., 2019], was provided to this paper by Wyscout, a world-leading company in the football industry. Data was collected by Wyscout using video expert analysis, called *operators*, one for each team and an extra one supervising. They are trained with a football data collection with a proprietary software *tagger*. Tagging consists of three phases [Pappalardo et al., 2019]:

[1] https://www.football-data.co.uk/

Football-Data		
attribute	**type**	**description**
Div	string	League division
Date	string	Match date (dd/mm/yy)
HomeTeam	string	Home team
AwayTeam	string	Away team
FTHG	integer	Full Time Home Team Goals
FTAG	integer	Full Time Away Team Goals
FTR	string	Full Time Result (H=Home Win, D=Draw, A=Away Win)
HTHG	integer	Half Time Home Team Goals
HTAG	integer	Half Time Away Team Goals
HTR	string	Half Time Result (H=Home Win, D=Draw, A=Away Win)
Referee*	string	Match referee
HS	integer	Home Team Shots
AS	integer	Away Team Shots
HST	integer	Home Team Shots on Target
AST	integer	Away Team Shots on Target
HF	integer	Home Team Fouls Committed
AF	integer	Away Team Fouls Committed
HC	integer	Home Team Corners
AC	integer	Away Team Corners
HY	integer	Home Team Yellow Cards
AY	integer	Away Team Yellow Cards
HR	integer	Home Team Red Cards
AR	integer	Away Team Red Cards
B365H	float64	Bet365 home win odds
B365D	float64	Bet365 draw odds
B365A	float64	Bet365 away win odds
BSH	float64	Blue Square home win odds
BSD	float64	Blue Square draw odds
BSA	float64	Blue Square away win odds
BWH	float64	Bet&Win home win odds
BWD	float64	Bet&Win draw odds
BWA	float64	Bet&Win away win odds
WHH	float64	William Hill home win odds
WHD	float64	William Hill draw odds
WHA	float64	William Hill away win odds

Table 6.1: Attributes of a match in Football-Data dataset.

1. **Setting formations**: an *operator* lists the starting line-ups and the positions and numbers of players on the field.

2. **Event tagging**: for each ball touch, *operator* selects a player and creates a new event in the timeline. Then, it adds an event type and a subtype. Finally it adds the coordinates in the event field and some additional attributes.

3. **Quality control**: two checks are carried out. The first one is automatic, an algorithm matches the events tagged by the two *operators* to validate that the events makes sense. The second check is manual.

This database contains all matches and their events from the five major European leagues (English, Spanish, German, Italian and French) during the 2017-2018 season, as well as from the World Cup 2018 and European Cup 2016. Table 6.2 shows the total amount of matches, events and players available.

competition	#matches	#events	#players
LaLiga (Spain)	380	628,659	619
Premier League (UK)	380	643,150	603
Serie A (Italy)	380	647,372	686
Bundesliga (Germany)	306	519,407	537
Ligue 1 (France)	380	632,807	629
World Cup 2018	64	101,759	736
European Cup 2016	51	78,140	552
	1,941	3,251,294	4,299

Table 6.2: List of competitions with their total number of matches, events and players.

This dataset is stored in multiple JSON files (Figure 6.1). For each competition there is a JSON file for the matches and another one for the match events. There are other JSON files that store information about players, teams, coaches and competitions. Table 6.4 shows the match features and events, which are the most remarkable datasets in this data base.

type	subtype	tags
pass	cross, simple pass	accurate, not accurate, key pass, opportunity, assist, goal
shot		accurate, not accurate, block, opportunity, assist, goal
duel	air duel, dribbles, tackles, ground loose ball	accurate, not accurate
free kick	corner, shot, goal kick, throw in, penalty, simple kick	accurate, not accurate, key pass, opportunity, assist, goal
offside touch	acceleration, clearance, simple touch	counter attack, dangerous ball lost, missed ball, interception, opportunity, assist, goal
foul		no card, yellow, red, 2nd yellow

Table 6.3: Event types, subevent types and tags en el Wyscout dataset.

```
{
  "eventId": 8,
  "subEventName": "Simple pass",
  "tags": [{"id": 1801}],
  "playerId": 25413,
  "positions": [{"y": 49, "x": 49}, {"y": 78, "x": 31}],
  "matchId": 2499719,
  "eventName": "Pass",
  "teamId": 1609,
  "matchPeriod": "1H",
  "eventSec": 2.7586489999999912,
  "subEventId": 85,
  "id": 177959171
}
```

Figure 6.1: Example of an event (pass) recorded in JSON format in the Wyscout database.

Matches		
attribute	**type**	**description**
wyId	integer	identifier of the match
competitionId	integer	identifies the competition on the *competition* document.
dateutc	string	date and time with format YYYY-MM-DD hh:mm:ss
roundID	integer	match-day of the competition
winner	integer	id of the team that won the game, or 0 if ended with a draw
label	string	name of the two clubs and the result of the match
venue	string	stadium where the match was held
dateutc	string	date and time with format YYYY-MM-DD hh:mm:ss
teamsData	dictionary	several subfields describing information about each team
teamsData subfields (for each team):		
hasFormation	integer	0 if no formation, and 1 otherwise
score	integer	number of goals scored by the team during the match
side	string	home or away
teamId	integer	identifier of the team
coachId	integer	identifier of the coach
bench	list	list of players that started on the bench and some basic statistics about the performance (goals, own goals, cards)
lineup	list	list of players that started on the lineup and some basic statistics about the performance (goals, own goals, cards)
substitutions	list	list of team's substitutions during the match, describing the players involved and the minute

Events		
attribute	**type**	**description**
eventId	integer	identifier of the event's type (Table 6.3)
subEventId	integer	identifier of the event's subtype (Table 6.3)
eventName	string	name of the event's type
subEventName	string	name of the subevent's type
tags	list	list of event tags, additional information about the event (Table 6.3)
eventSec	integer	time when the event occurs (in seconds)
id	integer	unique identifier of the event
matchId	integer	identifier of the match the event refers to
matchPeriod	string	1H (first half of the match), 2H (second half of the match)
positions	list	origin and destination positions associated with the event. Each position is a pair of coordinates (x, y).
playerId	integer	identifier of the player who generated the event
matchId	integer	identifier of the match the event refers to
teamId	integer	identifier of the player's team

Table 6.4: Attributes of matches and events on original Wyscout datasets.

6.2 Exploratory Data Analysis

This section consists of the analysis of the source data, before creating the datasets, at a statistical level, first in order to know and understand how data is distributed, what patterns it follows and what outliers has, and in order to be able to make better decisions when creating the datasets, giving us hints about which features may influence the outcome of a match and which ones may not influence it at all.

The first thing to know is how balanced the dataset is. Table 6.5 shows that the dataset is unbalanced since almost half of the data, 45.7%, were home wins, while 29.5% ended up in away wins and only 24.8% in draws.

Draw	Home win	Away win
24.8%	45.7%	29.5%

Table 6.5: Distribution of the match results.

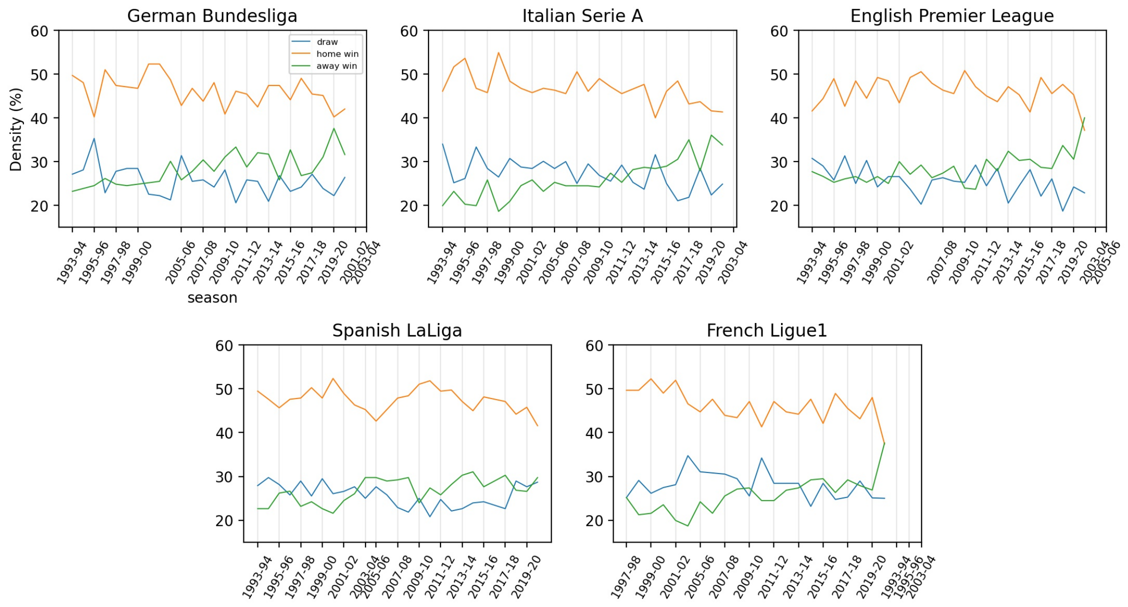


Figure 6.2: Distribution of match results since 1993 to 2021 in the five leagues.

Local victories have always been predominant (Figure 6.2), surpassing in some seasons and leagues more than half the number of matches, as in the Bundesliga 1996-1997 and LaLiga 2010-2011. Nevertheless, some surprising conclusions can be taken from this graph. One of them is the significant general increase in the number of matches that end in away wins. While 25 years ago, away wins were around 20%, in the last few seasons it has consistently exceeded 30%, and even in the 2020-21 season in the Premier League and Ligue 1 it has exceeded and equalled the number of home wins respectively. It can be seen how 20 years ago the away wins line was below the number of draws and for the last 10 to 15 years the away wins have overtaken the number of draws.

In contrast, home wins and draws have fluctuated over time, though with a slight decreasing trend over time. It is possible that the sudden drop in the 2019-20 and 2020-21 seasons in home wins and the rise in away wins is related to the capacity ban at football stadiums during the Covid-19 pandemic. In the French league graph, for example, we would see that in 2019-20 there is no significant change compared to 2018-19 as this competition was suspended when the pandemic and lockdown was declared. However, the following year, in 2020-21, when the league was played without a capacity ban, away wins spiked and home wins dropped.

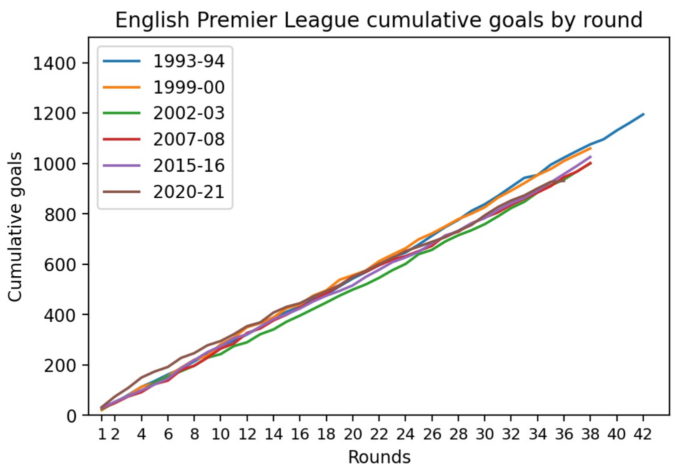


Figure 6.3: Cumulative goals distribution in different English Premier League seasons.

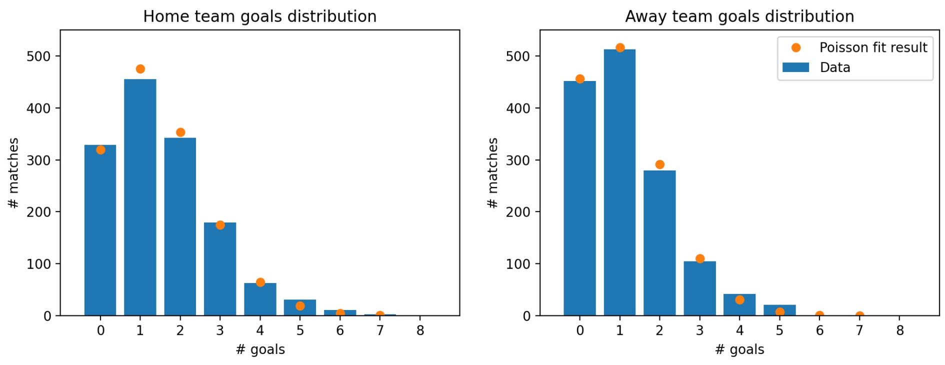


Figure 6.4: Goal distribution with fitted Poisson distribution

As we have seen, the distribution of results is not uniform over time or in each competition. As well as the results, other statistics have their own evolution over time. Figure 6.3 shows how over the last 30 years the distribution of goals scored per matchday in the Premier League remains very similar. While it is true that 25 years ago it seems that more goals were scored in the final rounds of the season, it can be assumed that goals scored follow a very similar distribution over time, so using goals scored in 1995 matches to predict today's matches should not be a mistake.

Figure 6.4 shows the distribution of goals scored per match for the five leagues in the 2017-2018 season. In the case that goals are considered to be events that can happen with a given probability during the 90 minutes, and it is assumed that matches are independent of each other and that goals in a match are independent of each other, it could be said that they follow a Poisson distribution. As shown in Figure 6.4 both distributions are very close to Poisson. Home goals follow a distribution with $\lambda = 1.485$ and away goals with $\lambda = 1.13$. Another conclusion that can be made from the two distributions is that the home team is slightly more likely to score than the away team. From Poisson Probability Cumulative Function (PCF), it is calculated that the away team has nearly a 70 percent probability of scoring none or one goal, while the home team has only a 56% percent probability of scoring none or one goal.

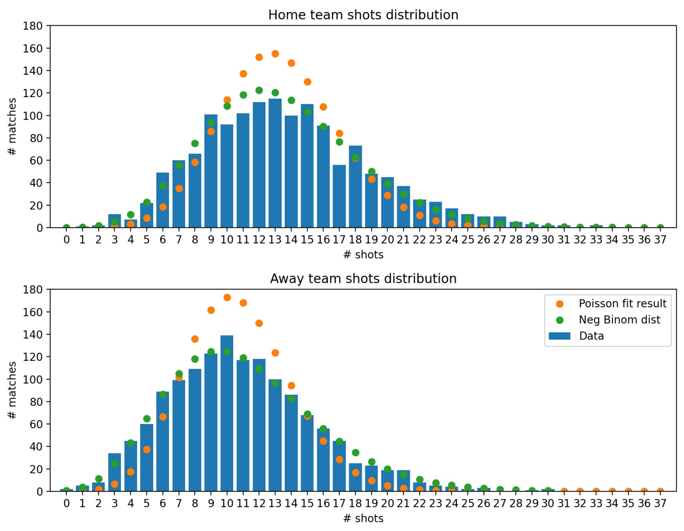


Figure 6.5: Shots distribution for season 2017-2018 with fitted Poisson and Negative Binomial distribution.

It is also interesting to see the distribution of the number of shots taken in a match. Figure 6.5 shows that this feature fits better with a Negative Binomial distribution, with parameters $r = 20, p = 0.6$ and $r = 11, p = 0.5$ respectively. The mean of both distributions are 13.7 and 10.86 respectively. The conclusion that can be taken from this is that home teams are more likely to take more shots than away teams. This gives us an idea of the bias that the goal features can provide to the model.

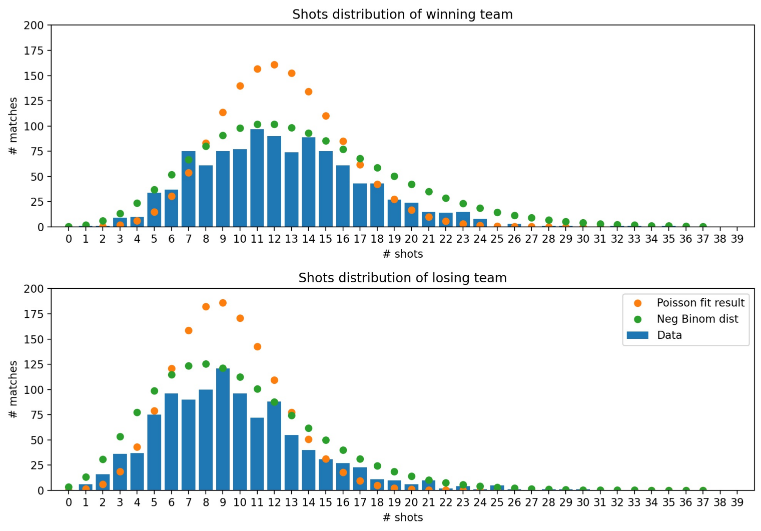


Figure 6.6: Shots distribution for season 2017-2018 of winning and losing teams with fitted Negative Binomial distribution

The distribution of shots taken by the winning and losing teams can give a (fairly intuitive) idea of whether there is a clear relationship between the number of shots taken and the outcome of a match. Excluding matches that ended in a draw, we can assume that it follows a Negative Binomial distribution (Figure 6.6). It can be seen how the losing team shot distribution is left-shifted compared to the winning team shot distribution. The mean and median of winning team shots is around 12 while for the losing team is around 9. Given this and the distributions of home and away shots, it is very likely that our model gives a lot of importance to the number of shots taken by each team, as the teams that take the most shots tend to win.

Finally, in terms of shots, it is worth comparing the distribution of the difference in shots between the winning and losing team. There are two scenarios here, when the team with the most shots wins and when the opposite happens, the team with the fewer shots wins (Figure 6.7).

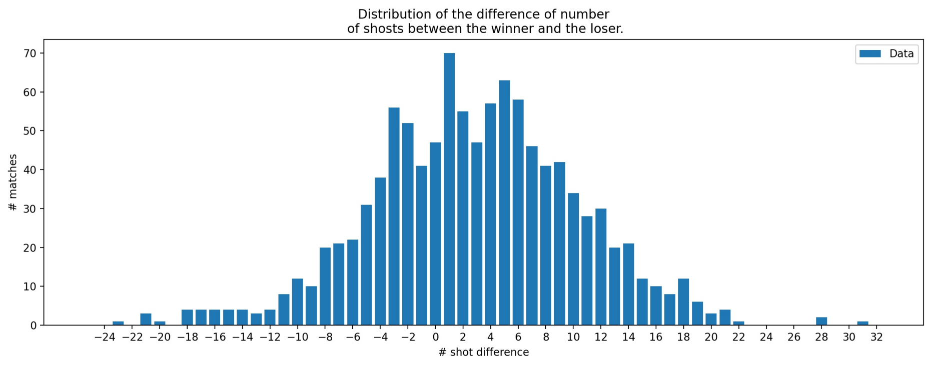


Figure 6.7: Shot distributions between winner and loser at season 2017-2018.

It can be observed that there are slightly more matches in which the winning team takes more shots, but the difference is not very large, the mean is 2.85 and the median is 3, so the distribution is just slightly right-shifted. This means that there are many matches that are very evenly matched on the pitch and that the result ends up going one way or the other by little details. It is not uncommon for a team to take more than 10 shots and end up winning; however, there are matches where the opposite happens. These cases are especially difficult to predict for a predictive model that takes as input a time series of events [Nyquist and Pettersson, 2017]. The team that takes the fewest shots ends up winning in 32% of games.

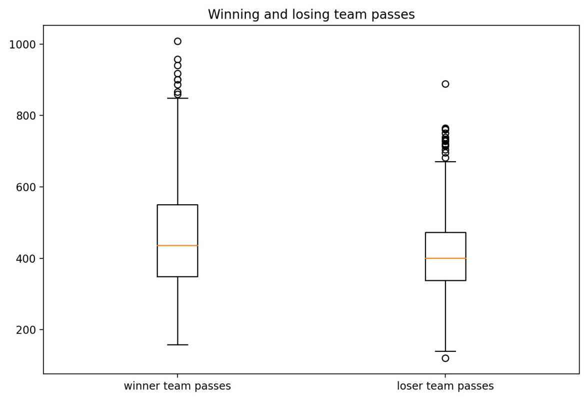


Figure 6.8: Winning and losing team passes box plot.

Another fundamental tactical aspect of football is passing. Figure 6.8 shows how winning team tends to make more passes during the match. In almost 25% of the matches the winning team makes between approximately 550 and 850 passes, while the losing team makes between 450 and 650 passes. It is true that the median and the first quartile

(Q1) of each distribution are practically the same, but the third quartile (Q3) and the upper range of typical values vary a lot between the two distributions. The conclusion that can be drawn is that if a team makes more than 600-700 passes it is very unlikely to lose the match.

rank.	team	pos.	acc.
1	Manchester City	1º	0.9
2	PSG	1º	0.89
3	Nice	8º	0.89
4	FC Barcelona	1º	0.88
5	Bayern Munchen	1º	0.88
...			
94	Leganés	17º	0.78
95	Stoke City	14º	0.77
96	Augsburg	12º	0.77
97	Crotone	18º	0.76
98	Getafe	8º	0.73

Table 6.6: Top 5 teams with most pass accuracy and its position in the 2017-2018 league table and the top 5 teams with the less accuracy at the pass.

But it is not only the quantity of passes that is important, but also their quality. Table 6.6 shows how in the top five teams with the best passing accuracy there are 4 league champions. In the five teams with the least passing accuracy it is more difficult to draw conclusions because although they are teams in the lower-middle part of the table, there is only one team among them that was relegated, Crotone. It is curious to see how Nice and Getafe both finished eighth in their respective leagues. The fact that one is the third with the most goals and the other is the least (with a big difference from the rest) is due to the style of play of both teams, while Nice bets more on possession and control of the game (like the styles of play of Manchester City or Barcelona), Getafe has a more defensive, aggressive and counter-attacking style of play. We can therefore see how a team's style of play can be inferred from some of their statistics.

rank.	team	goals	acc.
1	FC Barcelona	99	0.48
2	Hertha Berlin	43	0.44
3	PSG	108	0.42
4	Real Betis	60	0.42
5	Manchester City	106	0.41
...			
94	Sassuolo	29	0.3
95	Amiens	37	0.29
96	Nantes	36	0.28
97	Hellas Verona	30	0.28
98	Caen	27	0.27

Table 6.7: Top 5 teams with more and less shot accuracy and the total amount of goals they scored in 2017-2018.

However, football is all about goals, and although, as we have already seen, passing is important to win, shooting is even more essential to win matches. Table 6.7 shows the teams with the most and least goals scored and the goals they have scored during the season. It can be seen that the top scoring teams have very good shots, which means that they have very good strikers. It is remarkable that Hertha Berlin appears in second position. The teams with the least shot accuracy are also the teams that score the least goals.

The style of play of an opponent is one of the characteristics that coaches consider most often when studying their opponents. The strategies, tactics and line-ups that form a team's style of play are not static over time, but vary over time, depending on the coach, the players, the opponent and the context of the match. A team will not apply the same strategy on the first match day, when there is nothing at stake, as on the last match day when they are playing for avoiding relegation; in the same way, a team in the bottom half of the table will not apply the same strategy against a similar team as against a team in the top part of the table.

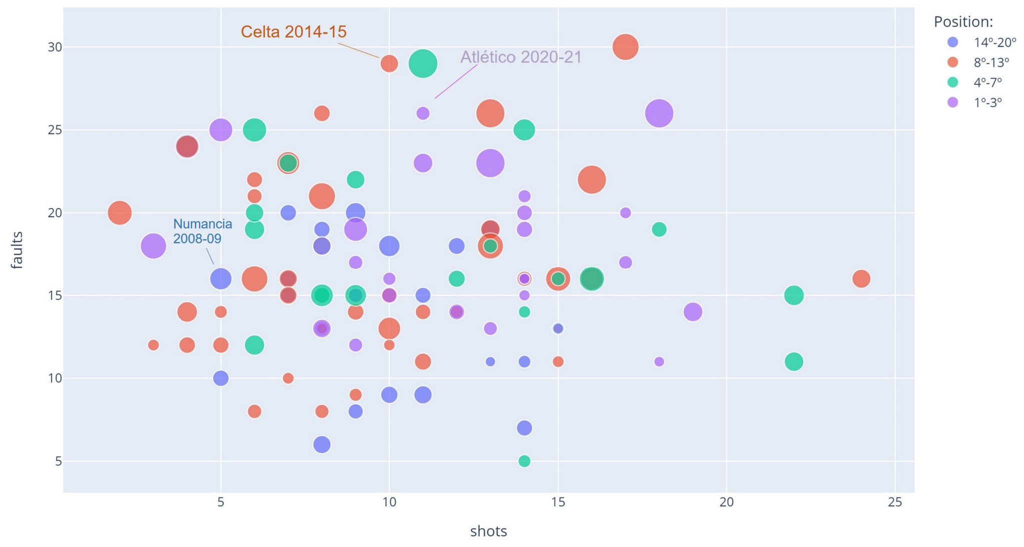


Figure 6.9: Scatter plot distribution of shots and fouls commited by teams that played against Barcelona and won in the last 15 seasons. The size of the point shows the mean of the fouls of that team in the match season, and the colors shows the position of the team in the table that season.

Therefore, styles of play interact with each other, and influence the final outcome of a match. In Figure 6.9 it can be seen how opponents who in the last 15 years have beaten Barcelona - a team that has a very dominant, attacking possession style of play - use different playing strategies. The teams at the top of the table rarely have fewer than 10 shots for the whole match, suggesting a more attacking approach due to the equal level of the two teams. Very few teams have beaten Barcelona by dominating the game offensively (with more than 15 shots), and none of these teams were ranked lower than 14th in the table. This may suggest that teams prefer to use a defensive and counter-attacking style of play against Barcelona.

Another interesting aspect of the style of play to analyse is the fouls committed by the team that beat Barcelona, and thus the aggressiveness of the team. Figure 6.9 shows, through the smaller points, how many teams that normally do not have a particularly aggressive style of play, make a lot of fouls against Barcelona, surely due to the defensive strategy they follow. This is the case of Atlético de Madrid in the 2020-2021 season or Celta in the 2014-2015 season. Both were teams that during the season did not commit many fouls; however, they used a defensive and aggressive strategy that gave them a good result against Barcelona. Almost every team that beat Barcelona committed more fouls in that match than the season average, suggesting that employing an aggressive and very intense style of play increases the chances of winning against Barcelona.

These patterns show us that it would be convenient to provide the model with as much information as possible about the playing style of the teams, as we have seen that these playing styles interact in influencing the outcome of a match. For this reason, we will consider features that indicate this aspect, such as fouls committed or yellow cards received.

Moreover, the style of play is not constant over time. Over the course of a season, despite different strategies are used against different opponents, there is always a certain style of play that predominates. Figure 6.10 shows the evolution over time of goals scored and conceded, as well as yellow cards received by three teams from different leagues and backgrounds: Atalanta (Italy), Getafe (Spain) and Schalke 04 (Germany).

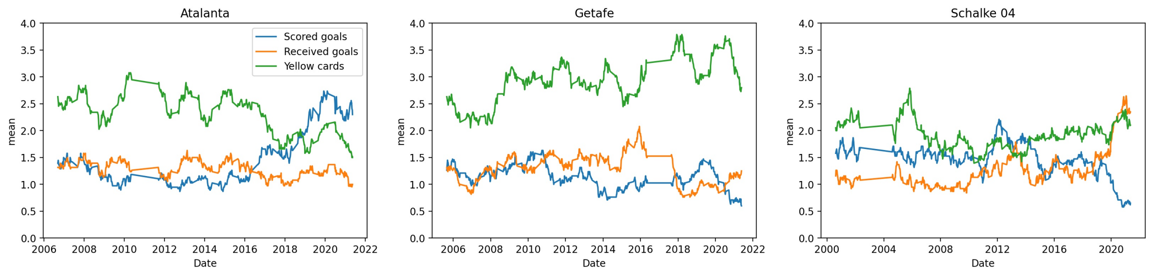


Figure 6.10: Evolution of the scored and received goals and yellow cards for PSG, Getafe and Schalke 04.

It can be seen in the graph on the left (Figure 6.10) the trend towards offensive style of play at Atalanta Bergamo since 2016, just since Gasperini's arrival as coach. In 10 years Atalanta went from 2nd Division to the top positions of Serie A by 2020 while being the highest scoring team and competing in the UEFA Champions League playing one of the most attacking and entertaining football in the old continent. It is interesting to see how the trend towards ultra-offensive, match-dominating football meant that the team needed to commit fewer fouls, going from an average of 3 fouls per game in 2012 to 1.5 in 2020.

A completely opposite success story, but one that also shows how a coach can radically transform the way a team plays and performs, is that of Getafe. Between 2017 and 2019 they experienced a big transformation in their style of play and managed to balance an aggressive game with a great defensive strength that led them back to competing in European competitions. This change coincided with the arrival on the bench of Bordalás (characterised by this style of play) during the team's stay in the 2nd Division in 2016-2017 and until his departure in 2020.

Finally, a case of a negative trend in performance is shown in the case of German Schalke 04. During the first decade of the 21st century and until 2015, it lived a golden era being one of the best German clubs of the new century, and although not winning the league, it was the runner-up in the league in 2005, 2007 and 2010, cup winner in 2001, 2002 and 2011, and participated 6 times in the UEFA Champions League, reaching the semi-finals in 2011. However, since 2018, they fell into a spiral of bad transfer policies and poor results that culminated in their relegation as bottom club and making only 13 points in the league, a negative record in the Bundesliga. You can see how the few last bad years have changed the way the team plays, and surely the loss of quality in the squad has led them to play a less offensive and aggressive style of football.

To conclude the analysis of the data, the relationship between the betting odds of *BET365* and the outcome of a match is shown. The aim of this is to see whether the odds hide a direct relationship with the probability of a team winning, and the degree of this relationship. In reality, the odds represent the inverse of the probability assigned by the bookmaker to an outcome. Therefore, the Figure 6.11 shows the inverse of the odds.

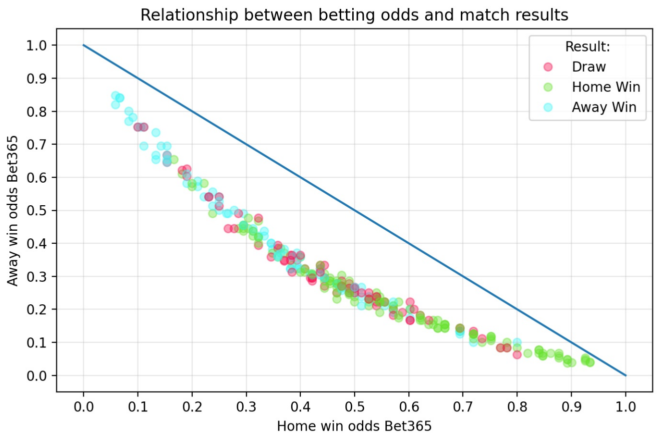


Figure 6.11: Bet365 probabilities based on betting odds.

Bookmakers predict accurately most of the matches in which there is a high probability that one or the other team will win. They predict almost all the matches from a probability of 0.7. Line with function $f(x) = 1 - x$ represents probability 1. When compared to the points on the probability curve of *BET365*, it is found that the distance between the curve and the points is the probability implicitly assigned by the bookmaker to the draw. It should be noted that bookmakers always assign a higher probability to the victory of one of the teams rather than a draw, making many mistakes in the middle of the graph where the matches are more evenly-matched.

With this analysis we have been able to see the distribution of some of the main variables of our datasets, such as goals, shots, fouls and passes. We have seen what patterns these variables follow and how they behave according to the outcome, how they vary over time and how they interact with each other. Now that we have an idea of what the data in the databases we have is like, we can move on to generating our datasets.

6.3 Dataset generation

In order to create a suitable dataset to train the model, different transformations had to be applied, the data from the databases had to be cleaned and processed and merged to obtain a more complete database. In total we have created three databases:

1. **Betting odds dataset:** created to train our baseline model. It consists of the Bet365 odds of the 2017-18 season matches of the five major European leagues.

2. **Event based dataset:** contains the matches of the 2017-18 season of the five main European leagues. Each record will correspond to a match, and 20 attributes represent the statistics corresponding to the last n matches by the players of each team's line-up. The features will be obtained either directly from the Wyscout dataset or generated from Wyscout events or statistics.

3. **Historical database:** contains data from the five major European leagues from the 1993-1994 season to the 2021-22 season. Each match will contain general statistics taken from the Football-Data dataset.

The main problem in the creation of the datasets has been the linking of both databases as their primary keys do not match, nor does the format of the team names match and, therefore, there was no straightforward way to connect the two databases. The solution that was proposed is an iterative solution. In the first iteration the team names that match in both datasets are detected, in the second iteration with the help of the Python library *difflib* the club names that are very similar in both datasets are detected, and finally the remaining teams (a dozen) were changed manually. Thus we could now associate matches from both datasets, as we can now identify in the Football-Data (Table 6.1) dataset the name of a club with its id in Wyscout (Table 6.4).

When matching both databases only the names of the teams have been considered as we are only interested in the matches of the 2017-2018 season, which are the ones that appear in Wyscout (Table 6.2).

dataset	#samples	#attributes
Betting odds dataset	1826	3
Event based dataset	1826	20
Historical database	45383	29

Table 6.8: Number of samples and attributes of the datasets.

6.3.1. Event based dataset

This dataset will contain data from the Wyscout database (Table 6.2). In order to predict a match the model will only receive as input some information about the last n matches of both teams. So, each samples in the dataset will have as a features some aggregated statistics and data about events and results of the last n games for both teams. In this section, some characteristics of this dataset and its features will be explained.

In this generated dataset, all samples are independent due to the transformations performed on the original dataset and the generated attributes that eliminate the temporal dependency between samples. The generated attributes have in common that they are all generated from features of the last n matches within a season. Therefore, each sample (match) already contains all the past information necessary to predict its outcome, actually it does not contain any features other than those of the past matches. This removes the temporal dependency between samples, making them independent.

The previous 4 matches have been used for the calculation of the attributes of each sample. It was decided to use this number of matches in order not to further reduce the limited number of samples that are available, since by using the last 4 matches, the first 4 matches of every season of each team must be used, thus losing quite a few records. Specifically, with 4 matches we reduce from 1826 to 1414 samples. Moreover, in previous works with a similar approach, the best results have been obtained with values between 4 and 7 matches [Ulmer et al., 2013, Jain et al., 2021].

It has been questioned in the literature whether there is a relationship between teams' winning streaks and the outcome of matches, especially when the streak is one of victories [Heuer and Rubner, 2009]. Heuer and Rubner argue that a negative streak has a negative impact on the probability of winning, while a winning streak has no impact on a team's probability of winning. However, other studies contradict these claims and the intuition itself, showing how a winning streak results in a decrease in the probability of winning, and a losing streak in an increase in the probability of winning [Goddard, 2006]. The relationship between streaks and results remains uncertain.

In the generated dataset each sample contains 20 attributes, which will be described and discussed below:

- **mins4_S**[2]**:** average number of minutes played by each team's line-up in the last 4 games. The values will therefore be between 0 and 90. This attribute should tell the model whether a team is playing with starters, substitutes or missing regular players from a team.
- **shots_11S:** absolute number of shots taken by each team's starter in the last 4 games.
- **shots_acc_11S:** average number of shots on target of the line-up in the last 4 games.
- **goals_S:** sum of goals scored by each team's starting players in the last 4 matches.
- **goals_ratio_S:** ratio of goals scored by the starting eleven over the total goals scored by the team in the last 4 games. This attribute will also be an indicator of whether important players are missing from the line-up or playing in the starting line-up.
- **passes_11S:** sum of passes made by each team's starting eleven in the last 4 matches.
- **passes_acc_11S:** passing success rate of each team's starting eleven in the last 4 matches.
- **keyPasses_S:** sum of key passes made by each team's starting eleven in the last 4 matches. Key passes are passes that create situations that result in a clear chance for the team.
- **ataque_S:** indicator of the attacking power of each team over the last 4 matches. It is calculated from the weighted average of the goals scored in the last 4 matches, in which the most recent match has more weight than the oldest.
- **defensa_S:** indicator of the defensive consistency of each team over the last 4 matches. It is calculated from the weighted average of goals conceded in the last 4 matches, where the most recent match carries more weight than the oldest.

It is important to point out that the statistics of a team in the last 4 matches only affect home and away matches. For example, in the Malaga vs Celta de Vigo, the sample will contain data from Malaga's last 4 home matches and Celta's last 4 away matches.

[2]Attribute with "S" have two versions, one for each side team: home ("H") and away ("A")

6.3.2. Historical dataset

The idea of this dataset is similar to that of Event based dataset, to create a dataset in which each sample is independent and contains as much information as possible about the match. The purpose of this database is to train this model with a larger number of matches and, therefore, be able to generalise better in the predictions. This will be discussed later in the Experiments section.

This dataset has been created only from the Football-Data database. In order to create it, data from the seasons between 1993 and 2021 from English Premier League, German Bundesliga, Spanish LaLiga and Italian Serie A have been used, and also data from French Ligue 1 since 1997 as no data from previous seasons was available.

This dataset was created by merging all available seasons from the five leagues and preprocessing the primary keys to match the Wyscout dataset. The team IDs in both datasets have been identified by mapping the team IDs from the Wyscout dataset to this dataset, and then the matching games have been identified, being those from the 2017-2018 season to which the ID from the Wyscout dataset has been mapped. For teams and matches that did not appear in the Wyscout dataset, a new unique ID has been generated. In total this dataset contains 45,383 matches, which is considerably larger than the Wyscout dataset.

As for the features that this dataset contains, all of them have been generated from the variables shown in Table 6.1. Apart from the match metadata shown in variables such as competition, date, home and away team and result, statistics of the match such as shots, fouls or corners have been taken to create new variables for the dataset. The new variables that have been created are values that are intended to show the trend of a team in different time lags, i.e. for each team aggregations of these variables have been calculated in different time lags.

It is also worth mentioning that the source data did not include all data for all leagues and seasons, for example, we did not have information on the number of fouls committed in a match in the Spanish league until the 2005-2006 season, while in the English league we can know the number of fouls committed even in a match in the 2000-2001 season.

It is because of this difference in features between competitions and seasons that it has been decided to make different versions of the dataset with different variables and sizes, Table 6.9 and 6.10 show the variables and the version of the dataset in which they appear.

As in the creation of the Wyscout dataset, the characteristics of this dataset bring together certain historical statistics of a team in a given time lag. This lag can be represented in two ways (Table 6.10). The first way is to use the last n matches of a team within the same season, which will be called from now on n_lag. The second way is to use a time interval such as the last d days, which will be called d_lag. It has been decided to use experimenting with d_lag to perform the $lags$, because it is possible to keep more matches from the beginning of each season. With the version of n_lag the first games of each season are lost, since it is calculated intra-season. However, d_lag is calculated inter-seasonally, so even if there is no data for a team for the last 2 months ($60D\_lag$), there will certainly be data for the last year ($365D\_lag$).

In the Wyscout dataset, where there were fewer samples (only one season), if large lags were made, many matches from the beginning of the season would be lost. On the con-

Feature	Basic	Basic w/ Derby	Complete	Basic Longterm	Basic Longt. w/ Derby	Comp. Longterm
Side (home/away)	X	X	X	X	X	X
Scored goals	X	X	X	X	X	X
Received goals	X	X	X	X	X	X
Points	X	X	X	X	X	X
Shots			X			X
Shots on target			X			X
Corners			X			X
Fouls			X			X
Yellow Cards			X			X
Red Cards			X			X
Derby features		X			X	
# features (columns)	80	104	200	56	80	140
# matches (rows)	32.997	23.267	21.635	17.373	13.536	11.895

Table 6.9: Historical database features and versions.

dataset	time characteristics	lags of time
Basic	Lags are done exactly in the same way as in the Wyscout dataset, taking the last *n* matches of a team	*n_lag*: Range from 1 to 10 previous matches.
Complete		
Basic Longterm	Streaks are represented by the matches played within the last *d* days. Note: *Derby* features are represented with the last *n* matches.	*d_lag*: *d* is 15, 30, 60, 180, 365 (1 year), 730 (2 years) and 1825 (5 years) days.
Complete Longterm		

Table 6.10: Description of lag features.

trary, in this dataset, there are many more samples and, therefore, it has been decided to create longer lags, in order to give the model information not only about the recent performance, but also about the medium and long term performance. To achieve this, several versions of the dataset have been created where features have been added to show the performance of a team over a very long period of time, up to 5 years (1825 days).

In order to create the lags, the minimum required number of records for a particular lag to be valid has to be determined. It would make no sense for the third match of a team to use the lags of the last 2 weeks, 6 months or 5 years, since we only have records of two matches of that team, so the three lags would have the same value when they represent totally different time intervals. The same is true using the last *n* games. Therefore, a minimum number of historical matches has been set for each lag.

Table 6.11 shows an example. Here it is shown how for the first two matches of the season we do not have data for the last 15 and 60 days, because a minimum of two matches are required to create the value of these lags. The same applies to the 15-day lag for matchdays 5 and 6, as we do not have enough data for the last two weeks due to the international break. However, the 365-day lag does contain value for these matchdays because, although the minimum is 20 matches in the last year, this is something that is satisfied as long as the team has participated in the league the previous season. If a team has just been promoted that same season, we would not have data for this field (365 days) until the 21st matchday.

Regarding the variables of the history between the two teams (*Derby*), all the matches have historical values, but it could happen that a team makes its debut in the category or has been in the competition for a few seasons (at least since we have data) and the last n games between the two teams are not registered.

Date	Round	Home team	Away Team	Result	Points			Derby			
					15D	60D	365D	1	3	5	10
12/08/2007	1	Arsenal	Fulham	2-1			1.79	3.0	3.0	3.0	3.0
19/08/2007	2	Blackburn	Arsenal	1-1			1.82	3.0	1.0	1.8	1.5
25/08/2007	3	Arsenal	Man City	1-0	2.0	2.0	1.82	3.0	3.0	3.0	2.71
02/09/2007	4	Arsenal	Portsmouth	3-1	2.0	2.33	1.89	1.0	2.0	2.0	2.0
15/09/2007	5	Tottenham	Arsenal	1-3		2.5	1.95	1.0	1.0	1.0	1.4
22/09/2007	6	Arsenal	Derby	5-0		2.6	1.95	3.0	2.33	1.6	1.83

Table 6.11: Example of lags of points gained for Arsenal FC on the season 2007-2008.

A description follows of the different features in the different dataset versions and of the way in which they have been obtained:

- **Side_*t*_*S*[3]:** represents how much the team S has played as home or away team in the matches within the last t time. To calculate this, an auxiliary variable was created in the original dataset in which 0 was assigned to the home team and 1 to the away team. The value of this new attribute is the average of the auxiliary attribute of each team during this time period t. Therefore, 0 as the value of this attribute means that all the matches have been played at home and 0.5 means that half have been played at home and half away. The attribute tends to 0.5 as the time lag gets bigger.
- **Scored_*t*_*S*:** sum of scored goals by the team within the t time lag.
- **Received_*t*_*S*:** sum of received goals by the team within the t time lag.
- **Points_*t*_*S*:** mean points gained by the team within the t time lag.
- **Shots_*t*_*S*:** sum of shots by the team within the t time lag.
- **Target_*t*_*S*:** sum of target shots by the team within the t time lag.
- **Corners_*t*_*S*:** sum of corners taken by the team within the t time lag.
- **Faults_*t*_*S*:** sum of fouls committed by the team within the t time lag.
- **YellowCards_*t*_*S*:** sum of yellow cards received by the team within the t time lag.
- **RedCards_*t*_*S*:** sum of red cards received by the team within the t time lag.
- **Derby_*feature*_*n*_*S*:** mean of the points, goals and *Side* features of both teams in the last n matches against each other. This variable is meant to provide the model with historical information on the performance of both teams in the last 3, 5 and 10 matches between them.

Finally, it is important to highlight the reasons why the different versions change in number of matches with respect to the original dataset. The reason why matches are lost when training the model is that all records must have the same dimensionality; however, there are many matches for which there are no values available for some variables (lags) and, therefore, those matches that do not contain all the values are discarded.

[3] t represent the lag time. As it is shown in Table 6.10, depending on the dataset version, it can be either last n games or the last d days. S represents the home and away team.

First of all, in order to create the different lags, we lose the entire 1993-1994 and 1994-1995 seasons, as for many lags the minimum number of matches required is not reached. In the following seasons we also miss quite a few matches, especially for teams that are new to the competition. Finally, there are matches that are missed during all seasons, such as the first matchdays of the season or matches after the international football breaks.

In order to minimise the damage of dropping many matches during the experiments, different dimensionalities and different values of minimum matches required to create a lag have been tested to find a balance between number of records and performance of the model. Furthermore, the experiments have been implemented in such a way that it is possible to pass as a parameter the features we want the training data to have. Thus, we can eliminate features (such as the 15 days or the derby of the last 10 matches) that make us discard many matches. This functionality is used to find that balance between the number of features, the information provided and the number of records to be trained, and it can also be useful to eliminate certain variables that we detect empirically that do not add value to the model. This, however, will be discussed in detail further on.

6.4 Dataset visualization

In this section, some of the features generated in both datasets are statistically analysed and compared with the original features.

First of all, if comparing the distributions that follow the shot features generated in the first dataset, *shots_11H* y *shots_11A*, with the shot distributions in the source dataset (Figure 6.5) it can be seen that both distributions have equal shapes. This is because these generated attributes are the sum of the shots of a team's line-up players in the last 4 home and away matches (Figure 6.12), so it is logical that they follow similar distributions. Two

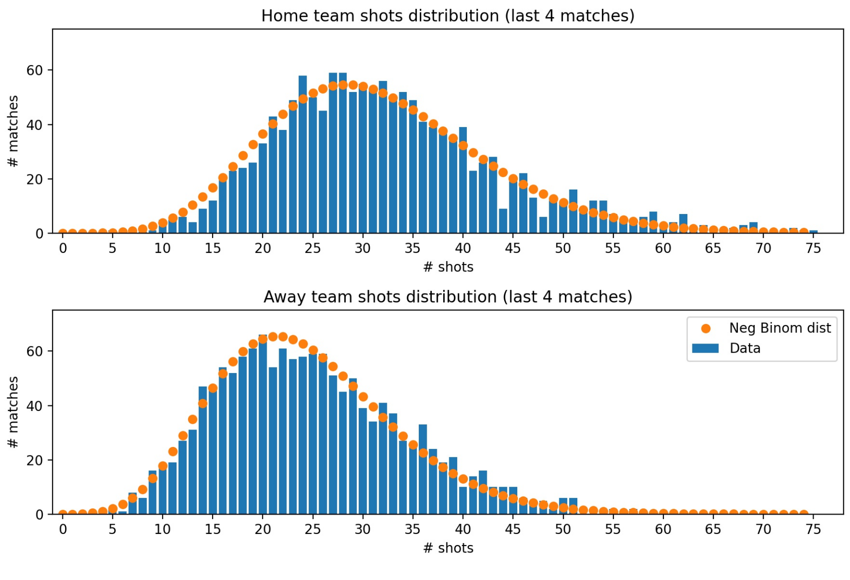


Figure 6.12: *shots_11H* y *shots_11A* distribution and fitted with Negative Binomial distribution.

completely new features generated in the new dataset are *mins4_H* and *mins4_A*, which are the mean of the minutes played by the line-up in the last 4 games. In Figure 6.13 it can be seen that they follow a distribution that can be fairly close to a Normal distribution, especially in the "slopes" of the PDF of the distribution, since in the area of the mean (the "peak") it fits worse and the distribution underestimates those values. The Figure 6.14

shows the relationship between both attributes and the match result, giving as a conclusion that there seems not to be an evident relationship between both variables, so they appear to be independent. Note that the purpose of these attributes is to indicate to the model whether a team plays with its usual eleven or with new players, a factor that is likely to influence the outcome of a match.

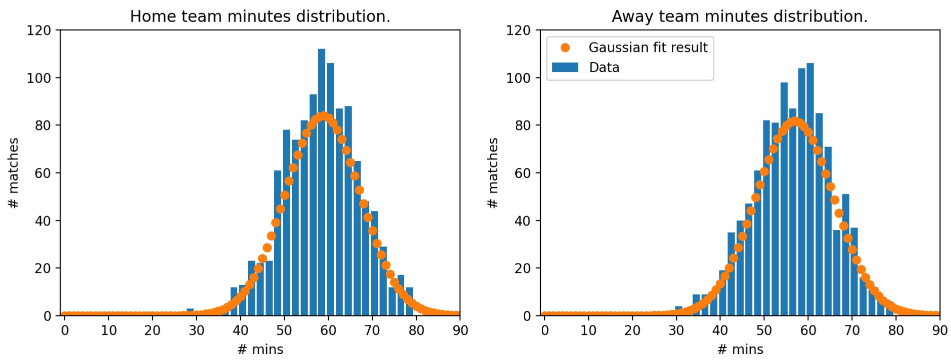


Figure 6.13: *mins4_H* y *mins4_A* distribution and fitted with Gaussian distribution.

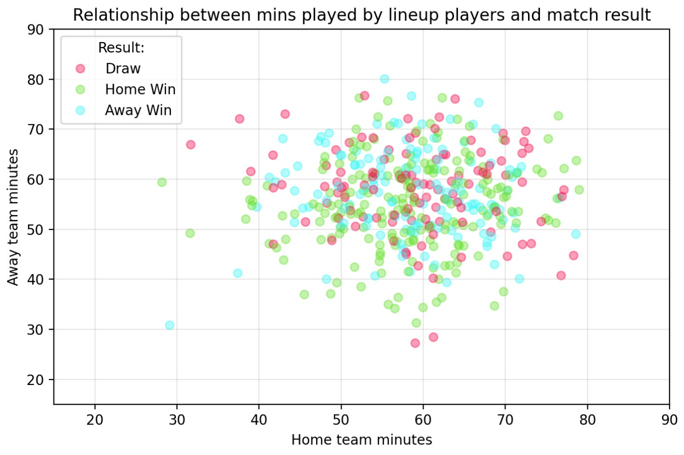


Figure 6.14: Relationship between *mins4* and *mins4_A* with match results.

rank.	team	pos.	att.-def.
1	PSG	1º	2.02
2	Juventus	1º	1.99
3	Bayern Munchen	1º	1.98
4	Manchester City	1º	1.94
5	FC Barcelona	1º	1.83
...			
94	Huddersfield Town	16º	0.69
95	Málaga	20º	0.68
96	Hellas Verona	19º	0.66
97	Caen	16º	0.65
98	UD Las Palmas	19º	0.57

Table 6.12: Top 5 teams with more and less attack-defense indicator in season 2017-2018.

Table 6.12 shows an indicator created to measure the attacking strength and defensive weakness of each team. For the model we will use two separate attributes for each team, *attack H or A* and *defence H or A* which will use the goals scored and conceded respectively to indicate this. However, in this table we show the ranking of the indicator (unifying attack and defence) considering all the matches of the season, in order to show how this indicator shows very well the level and the results of the teams.

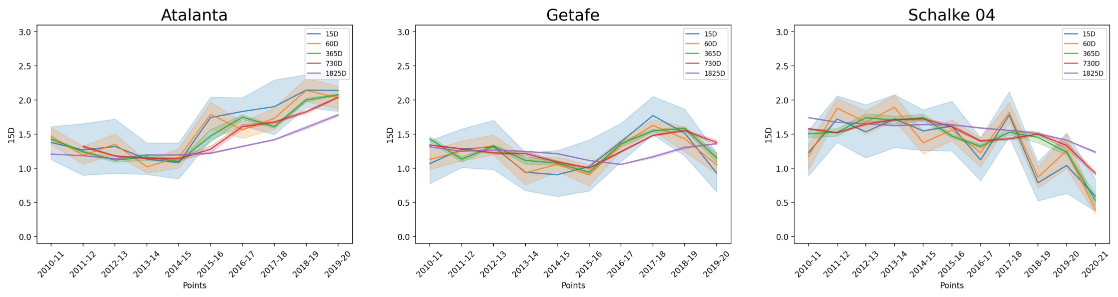


Figure 6.15: Time Series of the different lags of points gained by a team in historical dataset for every season. The dark line represents the mean of a lag in a season, and the shaded area represents the variability of that lag during a season.

On the other hand, the variables based on "lags" created in the Historical training dataset are intended to show the model the short, medium and long term trends of the teams. Short lags will show the model the most recent form of the team, while longer lags will show what the historical status of the team is, thus having complete information over time. Figure 6.15 shows how the lags of Atalanta, Getafe and Schalke 04 evolve over the seasons, which can be compared with the time series in Figure 6.10. The shaded area represents the variability within each lag and season. It can be appreciated that the shorter lags have much more variability, while the longer lags are much more constant and their line has a much smaller and more regular slope. In other words, a 15D lag will variate a lot depending on the week because it only takes into account two matches, while the 750D lag, for example, takes into account many matches so the mean of that lag will be more stable. The model will have to learn which is more important for deciding the outcome of a match, the recent streak or the long-term streak.

Date	**Home team**	**Away team**	**Result**	**points lag 60D home**	**points lag 60D away**	**points lag 730D home**	**points lag 730D away**
2015-10-24	West Ham	Chelsea	2-1	2.33	1.16	1.25	2.12
2015-12-14	Leicester	Chelsea	2-1	2.43	1	1.38	2.03
2018-04-19	Burnley	Chelsea	1-2	2.67	1.17	1.3	2.09

Table 6.13: Example of lags of average points gained and results for three Chelsea matches where Chelsea arrived with in a bad shape against another a priori smaller team that arrived in a very good shape.

Table 6.13 shows three games in which Chelsea, one of the best English teams in the last 20 years, came from a poor form and faced teams that were historically worse but were on a very good run of form. The first two matches are from the 2015-2016 season and Chelsea had been league champions the previous year, but this season they suffered a setback and finished 10th. In contrast, Leicester City, who had only been in the Premier League for one year, achieved one of the greatest triumphs in football history by winning the league that year. Therefore, both indicators of the streaks of the two teams are important in predicting an outcome. This example shows how the model will receive information about both Leicester's good year and Chelsea's bad year, as well as knowing that Chelsea and Leicester's level are likely to be temporary. The model will know this because of the long term level of the two teams (lag of 730 days).

In Table 6.13 two matches are shown which are won by the teams that were less favourite (less points in the long term) but which arrived with a better short term run; however an example is also shown of a match, Chelsea vs Burnley, which is won by the favourite team because that is why they are the favourites. Therefore, the model should learn from these and the other variables whether the short-term or long-term streak is more decisive for a match.

In addition to the lags, variables have been created based on the history of the match concerned (Table 6.9). There is a trend between the outcome of a match between two teams and their previous matches against each other (Figure 6.16). When the home team has dominated the last meetings (more than 2.5 points on average) it is very likely to win the next one. If the last meetings have been closer (between 1 and 2 points on average) the relationship with the outcome of the next match is weaker and any outcome is likely, although it still seems that the home advantage has a considerable influence and the home team is (a priori) more likely to win.

Also the home advantage has an influence on the fact that it is easier to break a winning streak of the away team than one of the home team. It can be seen in the last 3 games how it is easier for the home team to win despite the away team having dominated the last few games than vice versa.

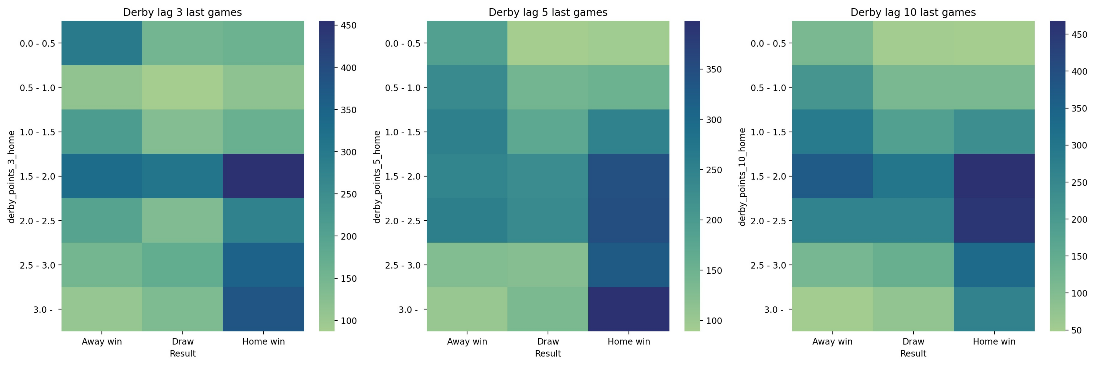


Figure 6.16: Relationship between the final result of the match (x axis) and the distribution of home points in the last *n* games between the two teams (y axis) in top 5 european leagues from 2018-19.

In this analysis we have seen the shape and distribution of the variables generated in the new datasets and how they relate to each other and to the outcome of the match. We especially highlight the potential of the different short and long term lags that give us a variety of information about each team. The model will have information on the team's recent streak on the one hand and on the other hand information on the long-term status of the club. We have also seen that there are variables (such as the average number of minutes played in the last 4 games by the starting team) that have a lower correlation with the result, however, this does not mean that the Machine Learning models that we apply will not be able to find a relationship that we do not see and add value to these variables.

The Event Based dataset contains valuable information about the players who are going to play the match, providing the model with key information about what might happen during the match. This dataset has the shortcoming that it only shows information about the last 4 matches, so the information about the trend of results over time is limited. On the other hand, the Historical dataset provides valuable information on just the team's trend over time through different lags of different lengths. However, this dataset contains much more aggregated and general characteristics, losing information about the starting players.

The model must learn to model match outcomes from the variables given in each dataset.

6.5 Dimensionality reduction

As part of the feature engineering and feature analysis, it is explained below how the dimensionality reduction process of the data has been performed. Dimensionality reduction is responsible for identifying, selecting and removing features that do not contribute to or even degrade the performance of the model. Two main types of dimensionality reduction techniques are shown below. These techniques will be applied to both datasets generated.

6.5.1. Feature selection

There are many techniques for feature selection but three techniques will be used, a univariate analysis technique, *Variance Threshold*, and two multivariate analysis techniques, *ANOVA* and *Pearson correlation*. Before applying the feature selection techniques, data is normalised so that it is on the same scale. The normalisation has been performed with the *normalize* (Eq. 6.1) function of the *preprocessing* package of the Machine Learning *Scikit-Learn* library.

$$x_{normalized} = \frac{x}{||x||_2} \tag{6.1}$$

Where x is the feature vector, $||x||_2$ is the L2 norm (Euclidean length) of the feature vector.

Variance Threshold The first and simplest feature selection method used is Variance Threshold. Variance Threshold is often used as a baseline to compare with more complex methods, and uses a threshold to drop all those variables whose variance does not surpass it. This reduces the dimensional data space D to a space of d dimensionality, with $D > d$. The variance measures the spread of the values of a variable, it measures how far a value is from the mean of the variable. This technique is used with the approach of training the model with the variables that have more variance, since it is likely that the fact that the data are more separated in the space of dimension d increases the separability of the data with respect to a space of the same dimensions but with variables with less variance.

During the training of our model, different thresholds will be used to determine which threshold selects, according to the variance, the optimal number of variables.

Some conclusions can be drawn from Figures 6.17 and 6.18. First, in Figure 6.17, pass success rate may not be a feature that can contribute much to our model because there is no significant variance in it. In Table 6.6 it was already apparent that this was the case due to the small range of the data, between $0.73 and 0.9$. A similar thing happens with shots, passes and shots on target, although in this case there is some variance and we will have to see the results of other methods to draw conclusions. On the other hand, goals are the

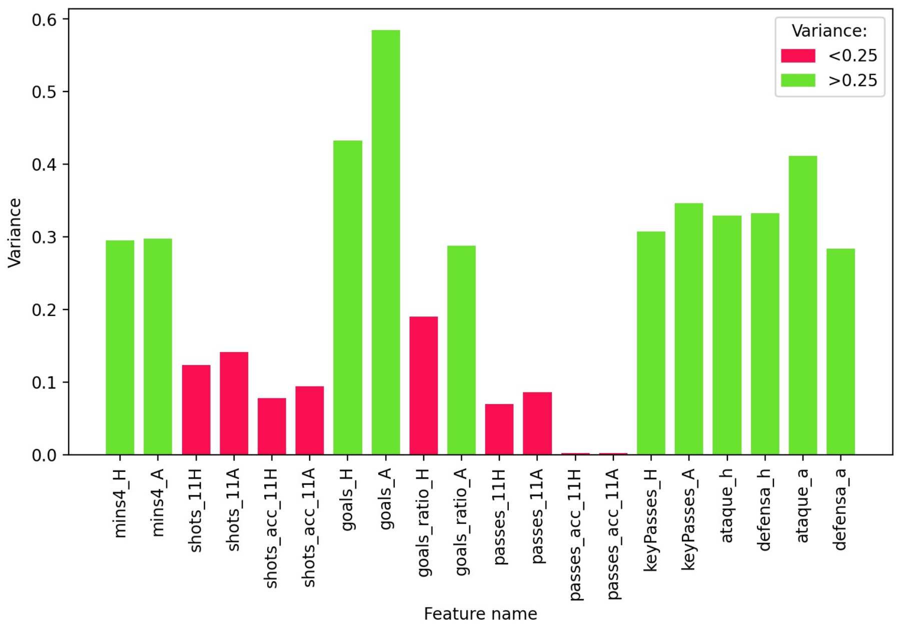


Figure 6.17: Normalized Event based dataset feature variances

variables with the highest variance, and therefore it is possible that it is the variable that helps our model most in order to know which teams are in the best shape.

Figure 6.17 shows in green the features that are selected with a threshold of 0.25, a total of 11 features. It mostly selects in pairs, e.g. from goals it selects both home and away goals, as the distributions that follow are quite similar as we have seen above (Figure 6.12).

Regarding the variance of the Historical features, it can be seen in Figure 6.18 how the shortest lags are the ones with higher variance, as it has been explained in 6.15. It is also interesting to highlight how points, goals and red cards are the features with more variance accross all the lags, while features like fouls or corners are more uniform and similar in every match.

ANOVA **An**alysis **of** **Va**riance is a parametric statistical hypothesis test for determining whether the means from two or more samples of data come from the same distribution or not. ANOVA shows the relationship between a categorical variable and a continuous variable, i.e. whether one depends on the other. More specifically, it shows how the continuous variables change with respect to the categorical variable. In this case, assuming independence of the samples, One-Way ANOVA will be used to find the relationships between the characteristics (continuous variables) and the class or outcome (a single categorical variable).

The ANOVA method works by for feature selection comparing the mean values of each feature across different groups defined by the target variable. If a feature shows a significant difference in its mean value across the groups, then it is considered important for predicting the target variable.

The following steps can be used to perform feature selection with ANOVA:

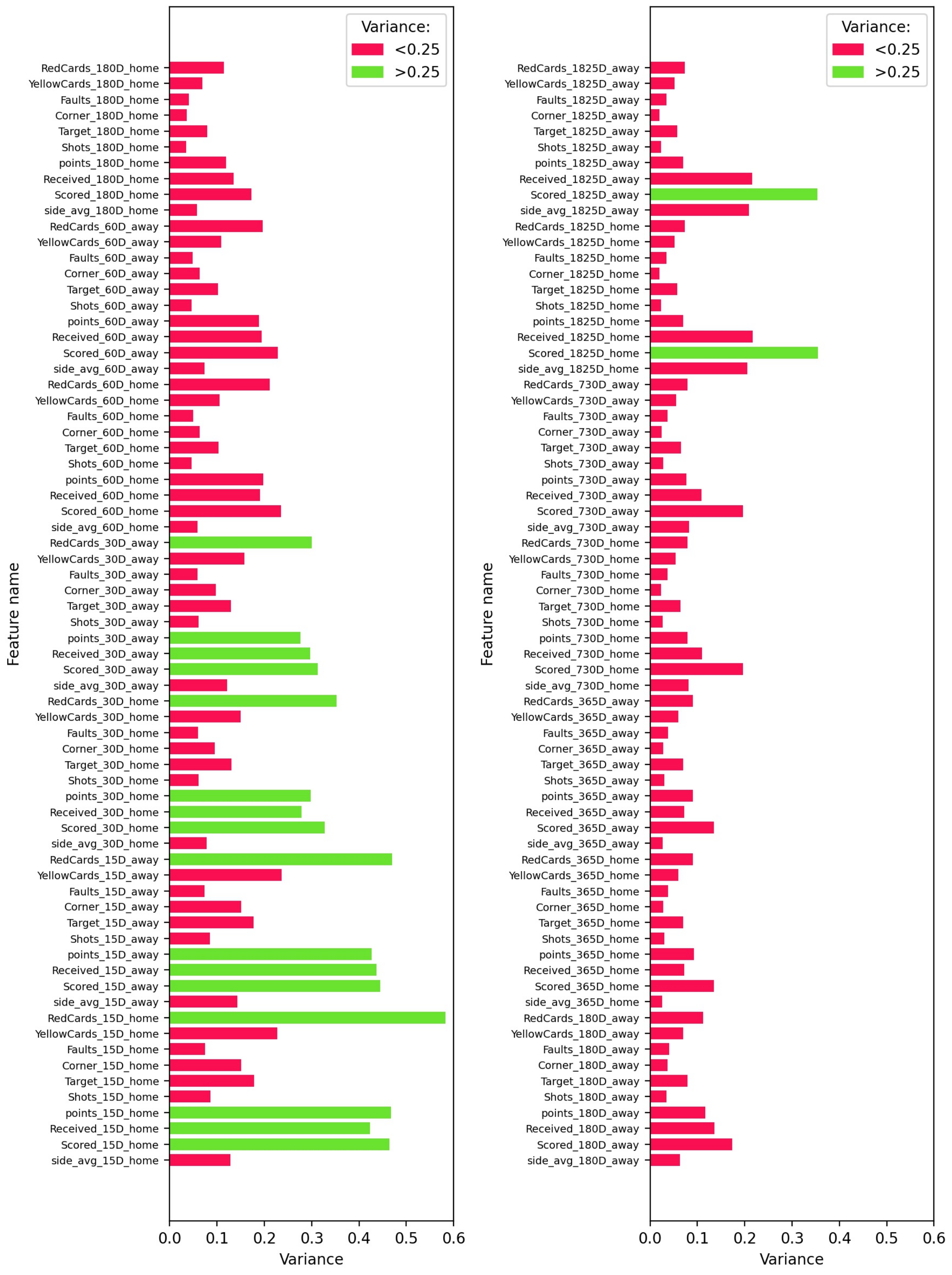


Figure 6.18: Normalized Historical dataset feature variances

1. Divide the dataset into groups based on the target variable (draw, home and away win).
2. For each feature, calculate the ANOVA F-value, which is a measure of the difference between the means of the groups.

$$F = \frac{MS_B}{MS_W} \tag{6.2}$$

   where MS_B is the mean square between groups, and MS_W is the mean square within groups.
3. Rank the features based on their F-values, with the highest F-value indicating the most significant feature.
4. Select the top-ranked features based on a pre-defined threshold or a desired number of features.

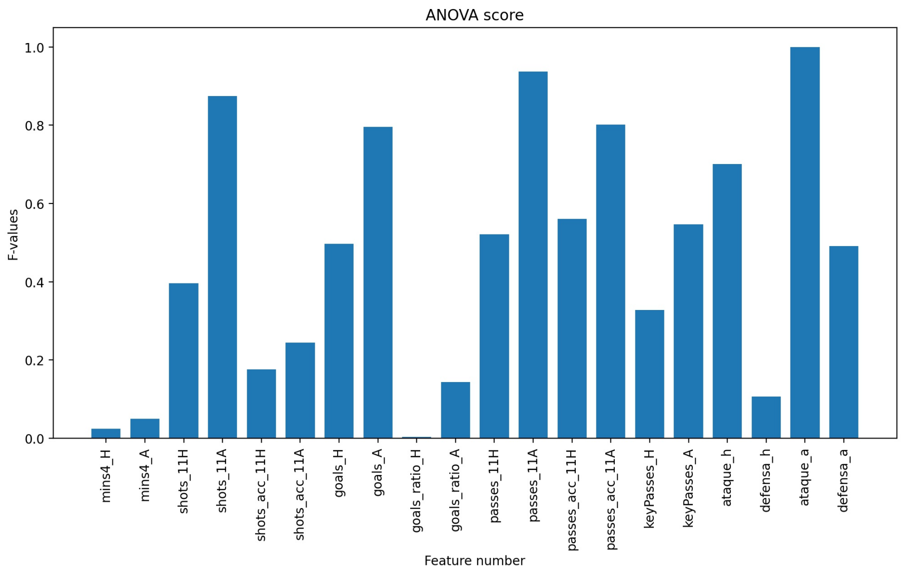


Figure 6.19: ANOVA F-value score showing the dependence between continuous variables and categorical target variable.

Figure 6.19 shows the F-values resulting from applying ANOVA to our data. Values close to 1 indicate that the means of each group are significantly different, so the variable and the outcome of the match (categorical feature) as closely related, while values close to zero show independence. These independent variables will decrease the performance of our model as they do not provide information to classify the match.

The ANOVA analysis shows that, surprisingly, the characteristics of the visiting teams vary more depending on the outcome of the match. No explanation for this has been found so far. Another surprising observation is the dependence between the result and the passing success. It was seen in the Variance Threshold (Figure 6.17) how these variables had a variance close to zero. There also exists a strong dependence of the results on shots, goals, total passes, key passes and attack indicator in the last four games, possibly due to the strong dependence between key passes, shots, goals and the attack indicator. It is also remarkable that there is very little dependence on the minutes played by the players in the last four matches, as already seen in the dataset analysis (Figure 6.14), and on the ratio of goals scored in the last four matches by the players in the starting eleven.

Figure 6.20 shows the ANOVA scores for the Historical dataset, and the top 50 features with more correlation with the target are coloured in green. It can be seen immediately that the features selected are completely different from the ones in Variance Threshold (Figure 6.18). This may be because the favourite team - the one with a better long term streak - usually wins the matches. However, this ANOVA score is not a reason to immediately remove the non-selected features from the dataset (red ones in the figure), as these features can provide meaningful information to the model in some cases, as it has been analysed previously. It is worth mentioning that there are features that may be strongly correlated between them - such as goals, shots or corners - providing tautological information to the model. But this will be discussed in the following subsection.

For selecting the features most dependent on the target feature we have used the *SelectKBest* class from the *feature selection* package of the *Scikit-Learn* library, and we have trained ANOVA with our data. The number of variables to be selected is indicated by the parameter k which is passed as a parameter to the initialisation of the *SelectKBest* object. This parameter k will be trained with our model to identify the optimal number of variables.

Pearson Correlation The Pearson correlation coefficient measures the linear relationship between two continuous variables. This one varies between -1 and +1, with 0 implying no correlation. Correlations of -1 or +1 imply an exact relationship. Positive values imply that as x increases, so does y, and they are positive correlations. Negative values imply that as x increases, y decreases, and they are negative correlations. The Pearson (ρ) is essentially a normalized measurement of the covariance, it is the ratio between covariance of two variables ($cov(X,Y)$) and the product of their standard deviations (σ).

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{6.3}$$

All attributes have been compared with each other to find dependencies between two characteristics to see if it is possible to discard variables due to a high dependency on another. These are the strongest and weakest dependencies (Table 6.14):

It can be seen in Table 6.14 the strong dependence between the variables of goals and attack, as had been assumed in the ANOVA analysis (Figure 6.19). Figure 6.21 compares the p-values of several local team variables. The strong dependence between passes, key passes, shots and goals can also be seen. The smooth negative dependence between passes and the defence indicator is also observed, meaning that when a team makes a lot of passes it is likely to concede few goals. Pearson also highlights the independence of some variables as we can see both in the figure and in the table, for example independence between the number of shots and goal scoring, the minutes played by the starters and the ratio of goals scored by the team in the last four games. In the table of lower dependencies (Table 6.14) you can see, as you might have guessed, that the most independent variables are those of one team with respect to the variables of the other team. This analysis applies both for the Wyscout and the Historical dataset.

From this analysis some interdependent variables can be discarded, also taking into account the Variance Threshold and ANOVA analyses, and assuming that this will improve the performance of the model. Therefore, for the initial dataset (6.3.1), if we made a manual selection of variables from the analysed, we would take these twelve variables:

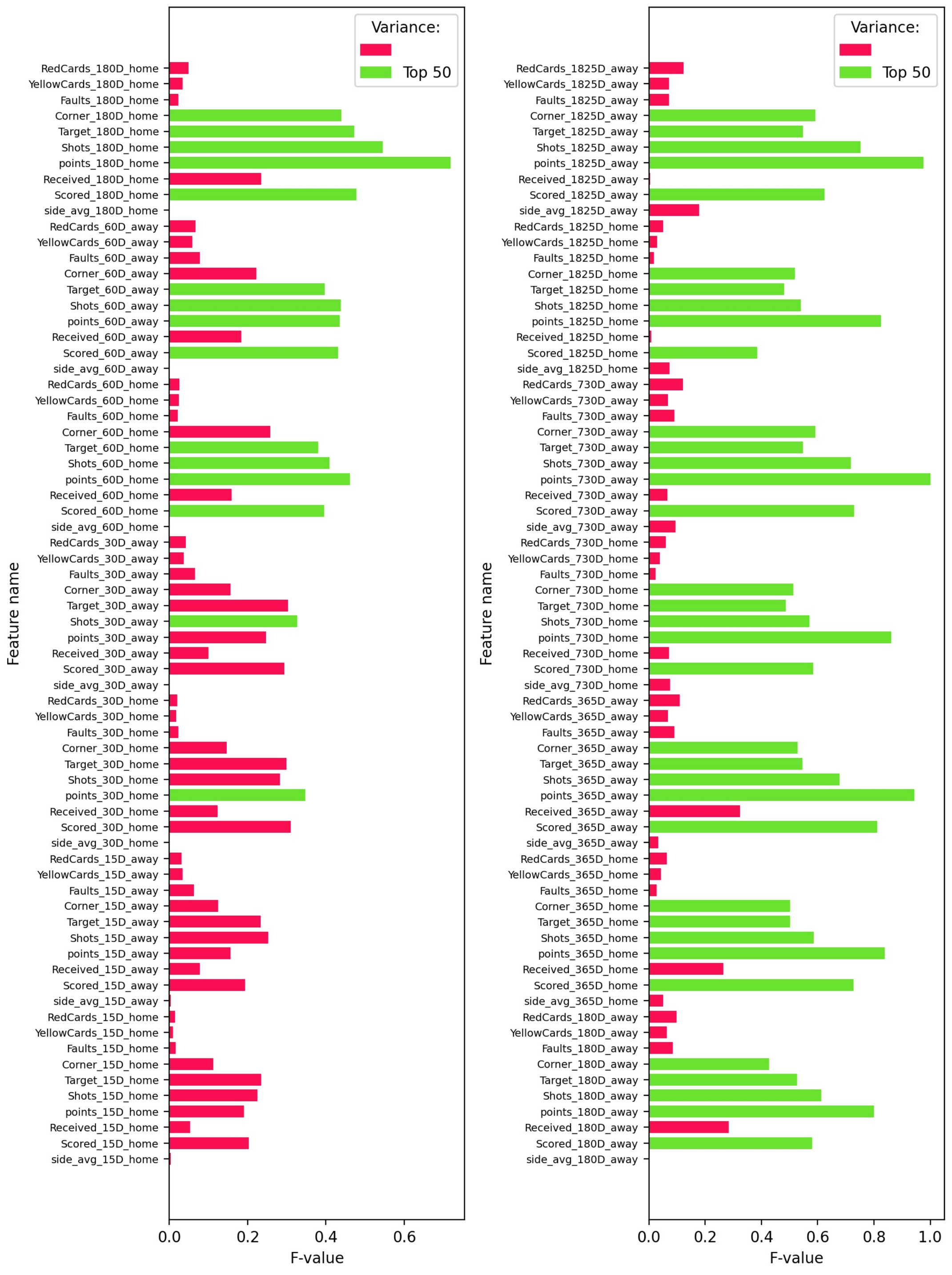


Figure 6.20: ANOVA F-value score showing the dependence between continuous variables and categorical target variable in 6.3.2

Highest dependency			
rank	**var 1**	**var 2**	**p-value**
1	mins4_H	mins4_A	0.9
2	goals_A	ataque_A	0.77
3	goals_H	ataque_H	0.75
4	passes_11A	passes_acc_11A	0.68
5	passes_11H	passes_acc_11H	0.65

Lowest dependency			
rank	**var 1**	**var 2**	**p-value**
1	goals_H	passes_11A	1.0e−5
2	goals_A	ataque_H	1.97e−4
3	shots_acc_11A	goals_ratio_H	1.1e−3
4	shots_acc_11H	goals_A	1.2e−3
5	shots_acc_11A	passes_11H	3.3e−3

Table 6.14: Pearson correlations for Wyscout dataset.

shots_11H	shots_11A	shots_acc_11H	shots_acc_11A	goals	goals_A
passes_11H	passes_11A	ataque_H	defensa_H	ataque_A	defensa_A

Table 6.15: Manually-selected features based on intuition and feature selection analysis

The performance of the model with these selected variables will be compared with the performance of the variables selected with the other dimensionality reduction methods.

6.5.2. Principal Component Analysis (PCA)

This data dimensionality reduction technique is a different approach to feature selection techniques. It is a supervised learning technique in which the dimensionality of samples is reduced by calculating the first n principal components, which transforms the original data to a smaller dimensional space while keeping as much information as possible contained in the original data.

The principal components of a set of samples or points in a d-dimensional space are a set of d unit vectors, where the i-th vector indicates the direction of the line on which, if the points are projected, they maintain the maximum possible variance of the projected points and in turn the vector is orthogonal to the previous $i-1$ unit vectors. Therefore, the problem faced by PCA is the mapping of the data into a lower-dimensional space, maximising the variance of the projected data in this low-dimensional space. In other words, the aim is to reduce the number of features with the minimum possible loss of information. It can also reduce existing noise in the original data.

PCA is usually explained and implemented with the calculation of eigenvectors (matrix **V**, Eq. 6.5) and eigenvalues (diagonal matrix, **L**, Eq. 6.5) of the data covariance matrix (matrix **C**, Eq. 6.5) which we pretended to transform. The first d eigenvectors correspond to the n-th first principal components to which the data are projected (matrix **X**) obtaining then the transformed data into a d-dimensional space (Eq. 6.6). The percentage of variance retained in the transformation can be calculated as the sum of the first d eigenvalues

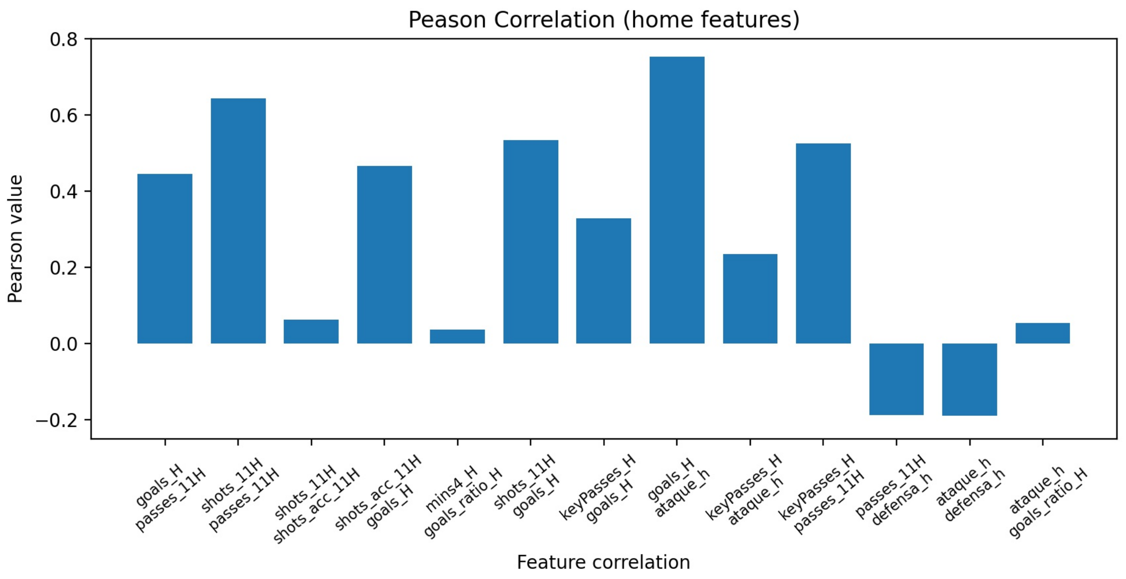


Figure 6.21: Pearson correlation between some home team features of the last four matches as home team.

divided by the sum of all eigenvalues.

$$\mathbf{C} = \mathbf{X}^{\top}\mathbf{X}/(n-1), \tag{6.4}$$

$$\mathbf{C} = \mathbf{V}\mathbf{L}\mathbf{V}^{\top}, \tag{6.5}$$

$$\mathbf{Y} = \mathbf{X}\mathbf{V}, \tag{6.6}$$

However, in this case, an SVD-based implementation of PCA has been used. If SVD is applied on **X**, it is obtained:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}, \tag{6.7}$$

where **U** is a orthonormal matrix and **S** is the diagonal matrix of singular values. Replacing in the covariance matrix:

$$\mathbf{C} = \mathbf{V}\mathbf{S}\mathbf{U}^{\top}\mathbf{U}\mathbf{S}\mathbf{V}^{\top}/(n-1) = \mathbf{V}\frac{\mathbf{S}^2}{n-1}\mathbf{V}^{\top}, \tag{6.8}$$

which means that the right matrix of singular values (**V**) are the eigenvectors and principal directions, and eigenvalues are related with the singular values (**S**) [Shlens, 2014]. The principal components, i.e. the projection of the data in their new space is given by:

$$\mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}\mathbf{V} = \mathbf{U}\mathbf{S} \tag{6.9}$$

During the experiments, the hyperparameter d of the projection dimensions will be trained in order to obtain the optimal dimensional space in which to map the data.

CHAPTER 7

Experiments

This chapter describes the models implemented and the experiments performed to train and evaluate them. First, the baseline and benchmark experiments will be explained. Then we will explain the iterative process with which different experiments have been performed in order to find an optimal prediction model and dataset.

7.1 Baseline and benchmark forecasting models

A baseline model is a very simple, prototype model that serves as a reference to evaluate the performance and learning of more complex models. We will prepare two different approached baselines. The first baseline will be predict home win for every match, and the second one predict that the winner is the team with the most points in the last month. Later, we will build models that will have to have a higher accuracy than this ones, showing that they are learning from the data.

A benchmark model will also be prepared. The benchmark will receive as input the draw, home-win and away-win odds from the bookmaker *Bet365*. Based on this input, the model will predict the outcome of the matches. Later, the results of more complex models will be compared with this model, which will be used to see if the new models perform as expected, because if a new model matches or exceeds the accuracy of the baseline, it will mean that it has a better predictive ability than the odds assigned by the bookmakers.

Baseline I: only home wins As we have said, it simply consists of predict a Home win for every match. We have seen in the data analysis how home wins are the most common result in football, so this baseline will consist in predict as home win without considering any statistic or feature of the match. The accuracy achieved with this baseline is 45.6%, as its the percentage of home wins in our test set.

Baseline II: best in the last month This baseline takes the average points gained by each team in the last month before the game, and predicts the winner based on the one with more points. The accuracy achieved with this baseline is 42.1%. The models that will be trained will receive in the input this one and many more features, so it will have to learn the patterns and interactions between them to learn something and beat this baseline.

We can see in the confusion matrix (Figure 7.1) how this baseline manages to predict some draws, and predict home wins easily, however, it struggles in predicting accurately the away wins.

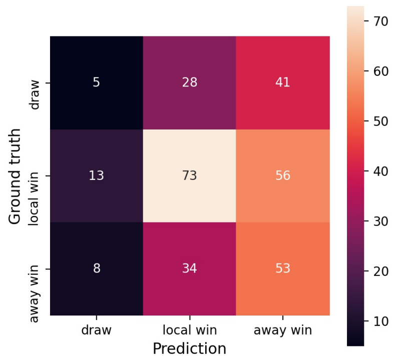
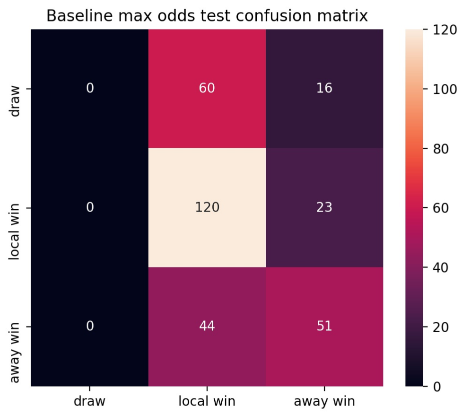
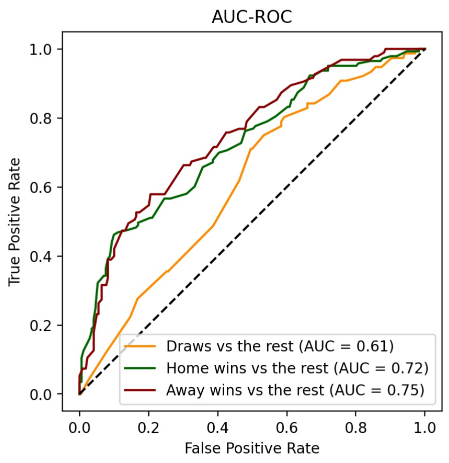


Figure 7.1: Confusion matrix of baseline: best in the last month.

Benchmark: max odds It consists of the prediction made by *Bet365*, as the model chooses the maximum probability between draw, home-win and away-win. As mentioned in the odds analysis (Figure 6.11) bookmakers never predict that the match will end in a draw, as they always assign a higher probability to the victory of one of the teams. The success rate was 54.45%. Figure 7.2 shows the confusion matrix of the results and the AUC-ROC curve. It can be seen in both figures that this simple model has great difficulty in distinguishing draws, with this class having the smallest Area Under Curve (AUC) of the three 0.61.



(a) Confusion matrix



(b) AUC-ROC

Figure 7.2: Confusion matrix and AUC-ROC curve of max-odds baseline.

These baseline and benchmark success rates will try to be outperformed by more complex Machine Learning models. It is remarkable to note that no baseline or benchmark predicts any draws, possibly because of the unbalanced nature of the dataset (Table 6.5). AUC-ROC is shown in Figure 7.2, where we can see that the draws are the most difficult for bookmakers to identify. Therefore, it will also be the objective of the models to be developed to improve the accuracy in the prediction of draws.

	Only home wins	Best of last month	Bookmaker odds
%	45.6	42.1	54.45

Table 7.1: Baseline and benchmark test accuracy.

This section explains the experiments carried out in which the created datasets, explained in the previous chapter, have been trained with two different Machine Learning algorithms: Neural Networks and Gradient Boosting. As explained previously in the Methodology, the approach taken was to start by carrying out experiments with simple solutions and models, and then build up the necessary software infrastructure to analyse the results and implement increasingly complex algorithms and solutions if the analysis so indicated.

We will now explain the experiments that have been carried out with simple MLP models and default configurations. We will start by explaining the experiments to debug the correct functioning of the infrastructure created to train the models.

7.2 Artificial Neural Networks experiments

The first Machine Learning algorithm that will be used to predict the results will be Neural Networks, specifically MLP networks. The aim of this part is to find the optimal neural network model to predict the results of football matches. During this part, therefore, we will perform a series of experiments in which neural networks with different configurations (architecture and hyperparameters) will be trained. The different configurations will be decided according to a heuristic based on the analysis of the previous experiments (Figure 3.1).

The neural network designed will be a MLP that will receive as input in the first layer the features of a football match, and will obtain three values in the output, one for each class: home win, away win or draw. The three output values will be normalised with a softmax in order to represent the probability that the sample belongs to each class. More precisely, the output of the softmax would be an estimate of the probability that the target (home win, away win or draw) is y given a match x, i.e. an estimate of $p(y|x)$.

It is true that the softmax output of the output layer of an ANN does not exactly represent a probability, mainly because although ANNs often make good predictions it is not accurate to use this output as true probabilities. In practice, the output of ANNs does not produce calibrated probabilities, i.e. they do not represent the probability that a prediction is correct, but contain a bias that tends to predict higher probabilities by being overconfident in the class with higher probability. This phenomenon is called overconfidence [Guo et al., 2017].

However, although [Guo et al., 2017] presents a solution for the calibration of the output of an ANN, this problem is outside the scope of this work, so we will assume the output of the softmax function as probabilities that have a margin of error. In addition, an error analysis of these probabilities will be performed using a confidence interval.

We have talked about the interpretation of the ANN output as probabilities, however it is essential to talk about the metrics the model uses to learn from the error it makes during training. The networks we will train will use Cross Entropy (CE), the function normally used for training models in multiclass classification problems. For an n match we seek to minimise the CE function, which looks like this:

$$L_n = -\sum_{c=1}^{C} w(c) \cdot t(n,c) \cdot \log p(n,c) \tag{7.1}$$

being $w(c)$ the weights assigned to each class in the loss-function, $t(n,c)$ the target class and $p_{(n,c)}$ the ouput softmax function for the match n and class c (draw, local or away win):

$$p_{(n,c)} = \frac{\exp x_{(n,c)}}{\sum_{j=1}^{C} \exp x_{(n,j)}} \tag{7.2}$$

The target values against which the model compares its prediction to measure its error are represented in a One-Hot Encoder vector where the index of the correct class has a 1 while the others have a 0. Our One-Hot Encoder vector has a length of three values, one for each class (home win, away win and draw), with draw being the first position of the vector (index 0), second position being home win (index 1) and away win the last position (index 2). The labels of each class of each match (i.e. each One-Hot Encoder index of each match) are represented as $t_{(n,c)}$ in Eq. 7.1. In this way, only the error in the target class is penalised, since the other labels of the One-Hot Encoder are set to 0 and vanish the error in those outputs. How to improve the model's evaluation of the training error will be discussed later.

Coming back to the estimation of the probabilities, these are integrated in the Cross-Entropy implementation provided by the PyTorch library (which we have used), so in the output of the model we have not used any softmax and we directly pass the value returned by the output layer to the Cross-Entropy function, which by the way uses a Log-softmax, which provides advantages such as being computationally cheaper and penalises the error more; and on the other hand, we use the output value of the network to calculate with softmax the probabilities of each class and then select the class with a maximum probability as prediction. The class that we finally predict $\hat{t}$ is calculated with the following formula:

$$\hat{t}_n = \arg\max_{c} p_{(n,c)} \tag{7.3}$$

Finally, the accuracy is obtained by calculating the average success rate, i.e., if $\hat{t} = t$, of all the matches. On the other hand, at first, the model uses the ADAM algorithm as an optimiser with a default configuration with a learning rate (lr) of 0.001. Furthermore, no normalisation is applied to the ANN layers, and the weights of the layers are initialised by default. The activation function used will be ReLU. The data, coming from the Event Based Dataset, have a dimensionality of 20 features. Before training, the data will be split into two random splits, one for training and one for testing, with an initial ratio of 75-25. In addition, the data will be normalised before being fed into the ANN. The network will train for e epochs, which will be a value to be optimised, and the network will be optimised, i.e. its weights will be updated for each batch of data, i.e. each b samples. The size of each batch (b) is another hyperparameter to optimise, but initially this will have a default value of 32 samples per batch. The model will be trained with the training set using 5-folder Cross-Validation to validate the model using in each iteration a part of the data to validate. Finally the model will be re-trained with the whole training set and will be validated by making inference from the testing set. It is important to clarify a seed will be assigned to the whole pipeline to ensure the reproducibility of the training.

The different experiments carried out with this family of algorithms will be detailed below:

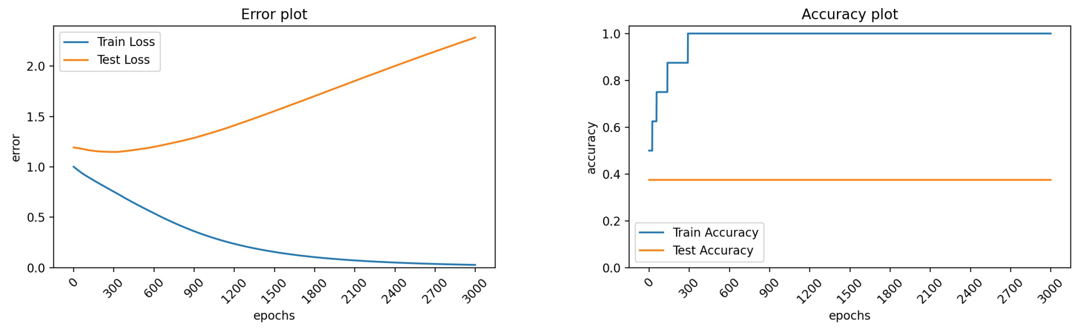
7.2.1. Bug-Free Models

In order to carry out the experiments, a software infrastructure had to be implemented to run the pipeline of experiments. The pipeline serves to have the same platform on which to train all the models (from the simplest to the most complex), to log information about the training and to save both these logs and the results. Once the basic infrastructure has been implemented and following the proposed methodology (Figure 3.1), a series of experiments have been carried out to debug this implementation and ensure that there are no errors or bugs in the training and in the log.

In order to test the correct functioning of the training pipeline, we have used a simple MLP with an architecture of 1 input layer with 10 neurons, no hidden intermediate layer and a multi-output of 3 neurons, one for each class to be predicted. With this, the neural network is intended to predict the probability of a match resulting in a home win, away win or draw. The input will be 20 dimensions, one for each feature of each record (each match). In total this network has 263 parameters to train. The other hyperparameters are the default ones, we will use ADAM as optimiser, Cross-Entropy as loss-function, ReLU as activation function and we will not use regularisation of any kind.

The test experiment consists of overfitting a training batch (8 samples, but it could be even less), and once we reach an error of 0 or very close to it, check the alignment of the predictions and the targets, and look at the logs to see if they make sense and work correctly.

If we look at the learning curve, we can see, on the one hand, how the model has over-trained the batch to the maximum, achieving an error of 0 and an accuracy of 100% in the train, while in the test the error has been increasing iteration by iteration. This is due to the fact that the model has learned the patterns that follow the train data from memory, without having any capacity to generalise when new samples are introduced. Another

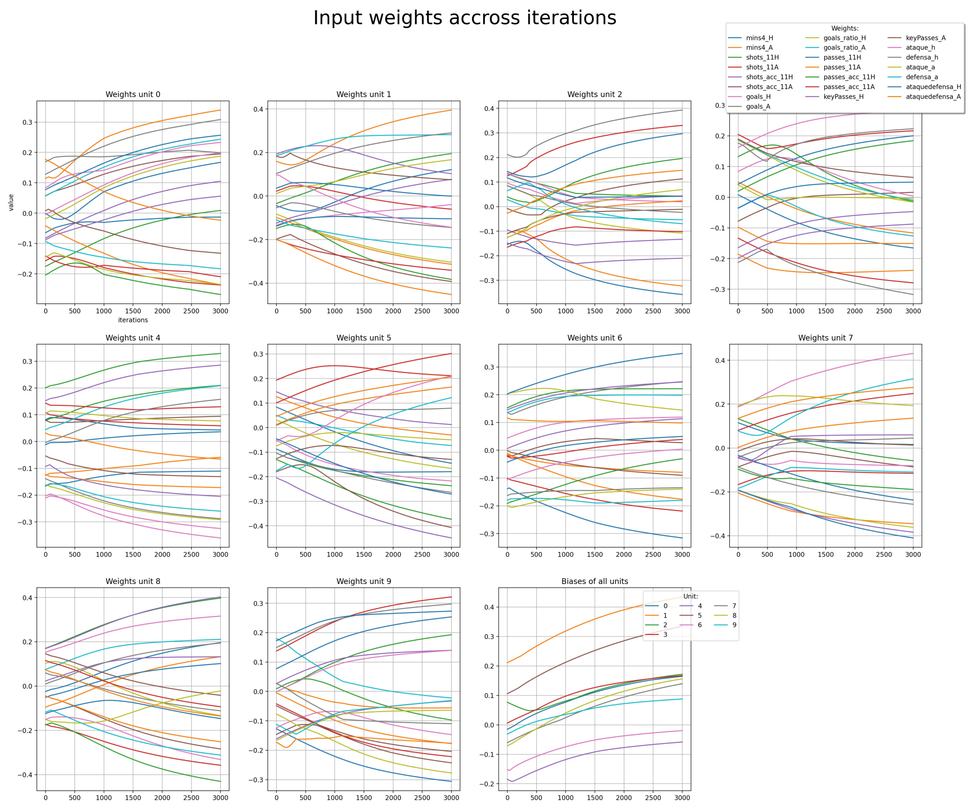


(a) Learning curve showing error accross epochs **(b)** Accuracy curve of train and test

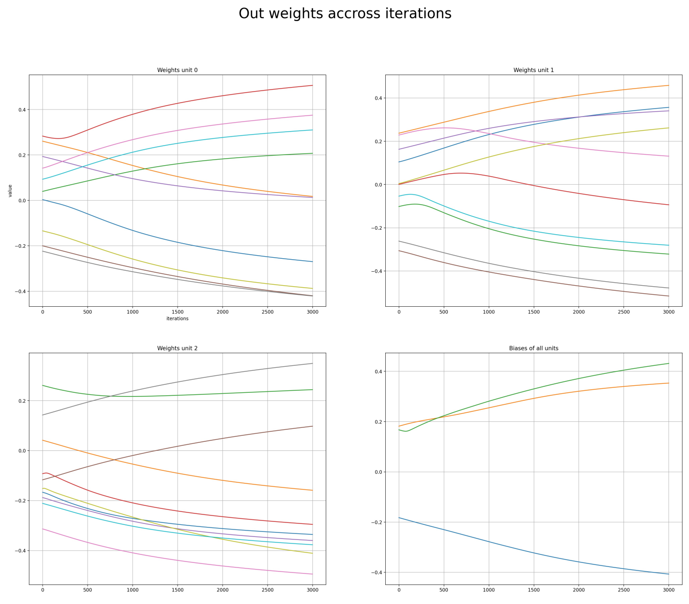
Figure 7.3: Event based data batch overfitted with a 263 parameter MLP.

observation that can be made from this test is that the accuracy obtained in an experiment can be maximised with a non-optimal error. As can be seen in Figure 7.3, an accuracy of 100% is reached in the train at epoch 500, however, at that instant the error was still quite high, close to 0.8. This gives an indication that model error is not the only indicator to look at when analysing the results and performance of a model, and that it is therefore not necessary to obtain the minimum error to achieve good or even optimal accuracy. Figure 7.4 shows the evolution of the weights during the test that has been carried out to check the correct working of the infrastructure on which the experiments will be carried out. It can be seen how, from the beginning of the training, most of the parameters converge, at different rates, towards the final weight distribution.

The experiments carried out with the entire dataset and applying the methodology proposed above are described below. These first experiments have been carried out with the Event Based dataset.



(a) Input layer weights during training.



(b) Output layer weights during training

Figure 7.4: Evolution of weights during the training. It pretends to show how the optimization of the weights is carried out during training. As it can be observed weights converge in the optimal distribution for overfitting the training set.

7.2.2. Event Based Dataset: architecture testing

The first experiments have been carried out to get an idea of how different architectures behave with this data, i.e. what results are obtained with ANNs of different depth and amplitude (neurons per layer). These experiments have been performed with the same configuration as the validation of the experiment infrastructure in the previous section, i.e. with a default configuration. The difference is that in this section we use all the samples of the dataset and the training batch that is passed to the network is 32 records. The network has been optimised with ADAM (default parameters), with a learning rate of 0.001, CE has been used as loss-function and has been validated with 5-folder CV, as explained at the beginning of this section (7.2). Furthermore, the trainings will have a duration of 250 epochs, meaning that the training dataset will be passed to the network 250 times. This value is an initial value that will have to be adequate according to the results, but, a priori, it has been chosen because we consider it appropriate given the limited number of samples we have.

The results of the architecture testing experiments are shown in Table 7.2. In this table it can be seen that for networks with more complexity, i.e. with a higher number of parameters, the model learns the training data better, although with considerable overfitting. The neurons are not distributed equally in every layer, that is the reason that we have same number of parameters with different amount of layers.

A model with 10,000 parameters achieves a success rate of 90% in the training set, but only 40% in the test. The more complex models, which have greater depth and more parameters, are probably too complex for the data we have, both in terms of the number of records (very limited) and the dimensionality of the input (20 dimensions). Another interesting point, which we have already mentioned above, is the fact that the model with the lowest test error does not coincide with the model with the highest test accuracy. This phenomenon will be analysed in more detail later. On the other hand, it is the simpler models with deeper networks (10-20 layers) that best generalise the test data, however, this error does not translate into good accuracy and they only achieve a 45% accuracy in the test, compared to 48% for the medium models of 2000 and parameters with only 3 layers.

It is important to mention that when we talk about n layers, we do not include the output layer but we do include the input layer, so in an n layer model we would have $n-1$ hidden layers in addition to the input and output layers.

layers	Train Error				Test Error				Train Accuracy				Test Accuracy			
params.	3	4	10	20	3	4	10	20	3	4	10	20	3	4	10	20
500	0.89	0.89	1.07	1.06	1.33	1.32	**1.07**	**1.07**	0.59	0.6	0.45	0.45	0.33	0.43	0.43	0.43
750	0.85	0.86	0.86	0.89	1.39	1.39	1.32	1.31	0.62	0.61	0.63	0.67	0.45	0.39	0.3	0.28
1000	0.85	0.87	0.9	0.91	1.37	1.36	1.3	1.31	0.61	0.6	0.64	0.69	0.44	0.31	0.24	0.25
1500	0.83	0.82	0.9	0.9	1.38	1.44	1.29	1.33	0.64	0.63	0.65	0.66	0.32	0.44	0.38	0.35
2000	0.8	0.8	0.89	0.85	1.4	1.44	1.3	1.40	0.65	0.65	0.61	0.59	**0.48**	0.43	0.43	0.40
3000	0.76	0.74	0.75	0.72	1.5	1.55	1.71	1.88	0.69	0.7	0.66	0.58	0.43	0.4	0.33	0.35
5000	0.67	0.68	0.75	0.7	1.57	1.64	1.82	1.07	0.73	0.72	0.67	0.46	**0.48**	0.37	0.31	0.42
10000	0.52	**0.37**	0.7	0.62	1.78	2.49	1.82	2.15	0.82	**0.9**	0.66	0.74	0.38	0.4	0.42	0.43

Table 7.2: Experiments results of different ANN architectures with default parameters on the Event Based dataset.

Figure 7.5 shows a comparison between four of the models shown in Table 7.2. These four models are detailed in Table 7.3. The comparison shows that the more complex models with more parameters learn faster and also have more flexibility in modelling the training data. However, these more complex models (3000 and 5000 parameters) have a considerably higher test error, which means that these models overtrain the training data. A contradiction, mentioned above, can be seen in the test accuracy graph where the models with the highest accuracy are also those with the highest error.

Modelo	Parameters	Layers	units perm layer
500	443	3	10-15-3
750	743	4	20-10-5-3
3000	3023	4	50-30-10-3
5000	4943	3	70-45-3

Table 7.3

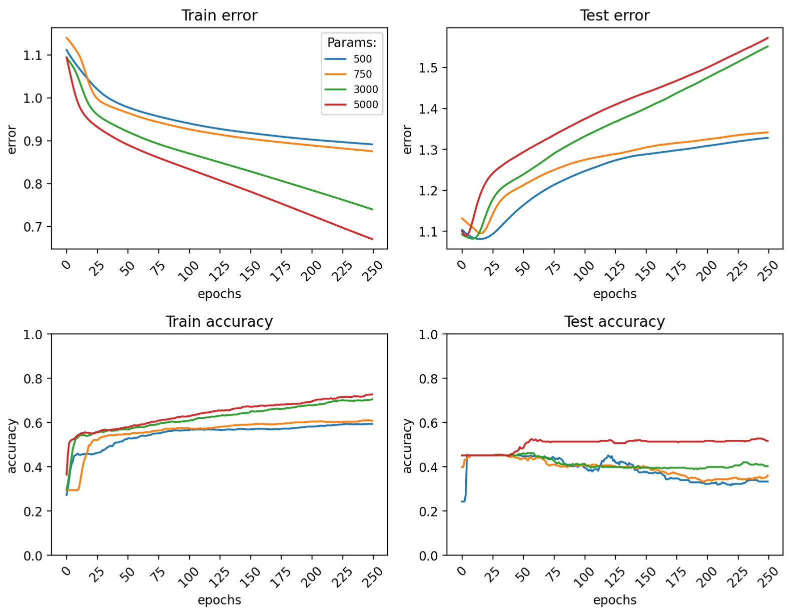


Figure 7.5: Error and accuracy learning curves for the 500, 700, 3000 and 5000 parameter ANNs.

Therefore, on the one hand, we have models that suffer from underfitting, i.e. the network is not flexible enough to model the training data and therefore converges in training with a high error. This may be due to different non-exclusive factors:

- **The model is too simple**: It may be that the model is too simple for the complexity of the data. In that case you could approach the solution by training a model with more complexity, either with more layers or more neurons in the layers. You could also reduce the complexity of the data by reducing its dimensionality. This is done by applying some dimensionality reduction technique.

- **Hyperparameters optimization**: these models have default settings, so you could modify the hyperparameters of the model to improve performance, for example, you could increase the Learning Rate to make bigger "jumps" when Gradient Descent updates the weights or increase the number of training epochs.

- **Speed of learning**: learning is rather slow in these models. Hyperparameters can be optimised to improve model performance, but a number of methods can also be applied to optimise model performance and make the model more stable and robust. Some of these methods are Batch normalisation and weight initialisation.

On the other hand, we have models that suffer mostly from the opposite, overfitting, i.e. the model learns the training data well but does not know how to generalise to new data not seen during training. Let's say that the model memorises the training data and matches the parameters to these data to the millimetre. This is the case for the 3000 and 5000 parameter models in Figure 7.5. In this case, in addition to optimising the hyperparameters such as the learning rate or the epochs, other solutions could also be applied:

- **Add more data**: this is the preferable option. It is widely accepted that more data, and more quality data, is the most effective way to reduce overfitting. However, we do not have more data in this dataset, so other solutions will be considered first before experimenting with a new dataset.

- **Regularization**: it would be useful to apply some method to regularise the ANN so that it learns to generalise better. Some of the techniques that can be applied are weight decay, drop out or applying weights to the error function. The methods used will be explained.

7.2.3. Event Based Dataset: underfitting and overfitting

The first solutions that have been applied have been aimed at reducing overfitting and improving the performance of our ANN to converge to an optimal solution more quickly. These solutions have been **Batch Normalization** (BN), **Weight decay** and **Drop-out regularization**, as well as changing certain hyperparameters in a manual and intuitive way to see what impact they have on the model.

In order to apply these methods in the experiments, the code had to be modified and extended to support the new functionalities without changing the way of training and saving the results and logs. For the other changes, however, no code modification was required, as weight decay is a parameter of the ANN optimiser.

The first method, Batch Normalization, is used to train ANN in a faster and more robust way by normalising (re-centring and re-scaling) the inputs of the network layers. As explained in a previous chapter, BN corrects the Internal Covariate Shift (ICS) of the input of each layer during training and unifies them under the same distribution, thus reducing the interdependence between layers. In addition, extra-parameters are trained in the BN layer to adjust the input distribution and exploit the non-linearity threshold of the activation function. It is important that the input of the activation function is at the non-linearity threshold to reduce the probability of Vanishing and Exploding Gradient problems occurring and for the ANN to converge faster to its optimal solution. This is why the BN is usually applied to the input of the activation function. This allows a larger learning rate to be used as the network is now more stable. In addition, BN reduces overfitting because by centring and rescaling the inputs we are applying noise, this noise makes it difficult for the network to "memorise" the training data, especially the smaller the size of each batch.

We can see in Figure 7.6 that BN does not have the same impact in all cases. In more complex networks, as in this 3000 parameter network, it has more effect and we see how the learning converges faster and is more stable. However, in simpler networks it does not have the expected effect and we see how the error and accuracy is more similar to training without BN.

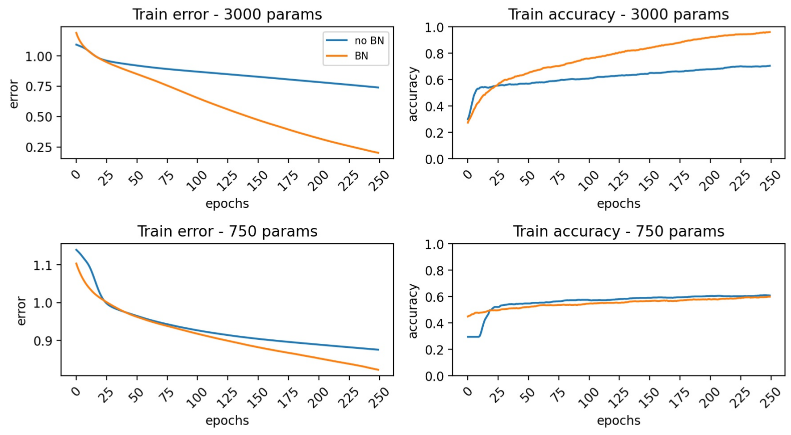


Figure 7.6: Effect of Batch Normalization on training for a 750 and a 3000 parameters model.

However, BN has not managed to reduce overfitting (Figure 7.6), in fact, it has increased it. To analyse the degree of underfitting and overfitting of a model we use bias-variance decomposition, which as its name suggests decomposes the model error into three parts, an irreducible error, bias and variance, the latter two being related to underfitting and overfitting respectively. The irreducible error would be the minimum possible error (whether human or not). In our case we will consider it as 0 since we have a way of knowing this value at the moment, and therefore we will assume that it is possible to achieve an error of 0. On the other hand, the bias is the training error and the variance is the difference between the error in the test and the train error.

Bias is related to underfitting because both are related to erroneous assumptions of the model during training, ignoring relevant relationships in the data. Variance and overfitting are both related to the sensitivity of the model to variations in the data, with the model having an excessive sensitivity (since it models even noise) in the training data that penalises it for new test data. When applying the bias-variance decomposition we assume that both data sets (train and test) come from the same distribution [Neal et al., 2018].

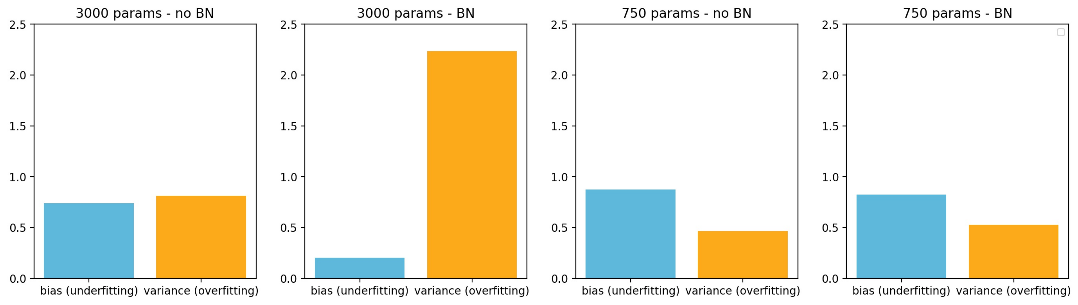


Figure 7.7: Bias-variance decomposition of trained non-BN and BN Neural Networks.

Figure 7.7 shows that BN reduces bias (underfitting) especially when the model has enough flexibility to model the data (∼3000 parameters), while when the model has few parameters (∼750) it has little room for improvement and flexibility. On the other hand, we see how BN does not reduce overfitting at all and, in fact, increases it by a factor of three.

We have seen so far that we need moderately complex models if we want to have a model with the ability to fit the data correctly. As we mentioned before, increasing the complexity of the ANN helps to reduce the error in training. However, we must now also reduce the overfitting so that our model has a better test accuracy.

To do this we have first performed several experiments with different weight decay values. Weight decay is a regularisation technique in which we apply an L2 penalty to the weights in the error function of our ANN in order to smooth them. Normally an ANN that is overtrained has weights with very large values (positive or negative) that allow it to be very flexible when modelling all the training data. The idea of weight decay is to penalise during backpropagation the fact of having weights with very high values, forcing the ANN to look for an optimal smoothie configuration.

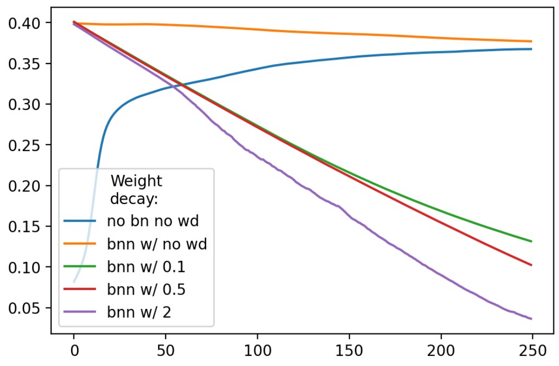


Figure 7.8: Mean of input activation function of a hidden layer during the training for 4 different models: Vanilla ANN without BN or weight decay, ANN with BNN but no weight decay and three ANN with BNN and weight decay of 0.1, 0.5 and 2 respectively.

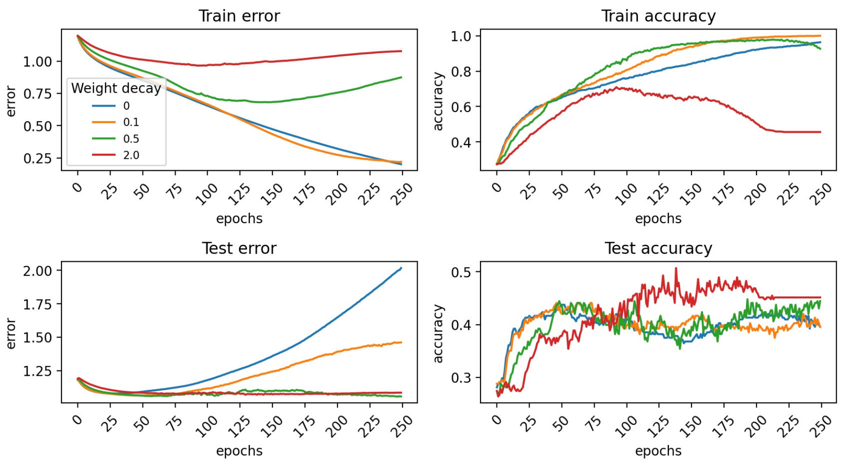


Figure 7.9: Error and accuracy for train and test experiments on weight decay.

In Figure 7.8 we can see how the larger the weight decay penalty is, the faster the values of the activation function input tend to 0. This is because in these models the weights tend towards 0 due to the L2 penalty applied. We also see how the model with BN and the one without BN converge to the same point, however, the one with BN is already initialised at a closer value so it takes less time to converge. This is another way of looking at the efficiency of training with BN versus without BN.

However, the application of weight decay to reduce overfitting has not produced the desired results (Figure 7.9). The truth is that although it does its job of regularising and has led to better test results, it seems that the penalty when updating the weights is too strong and does not allow the ANN to model the data well, especially in terms of accuracy.

We have also applied another well-known regularisation technique called Drop-out. This technique consists in temporarily deactivating some nodes of the ANN (input and hidden layers), thus creating a new architecture from the original ANN. The nodes are deactivated with a probability p. When an ANN seeks to minimise its error function, it does so by taking into account all the neurons at once, so during training, complex co-relationships are created between neurons where some "solve the error of others", which occurs intensively during overfitting. The Drop-out technique solves the problem of overfitting because by randomly eliminating certain neurons in each iteration we are creating a new architecture in each iteration (maintaining the weights of the neurons that are active) and therefore the neurons are forced to have less interdependence, as they are updated each time they are in a different architecture [Srivastava et al., 2014]. Actually, Drop-out is a variant of bagging with several models. It is also important to mention that, according to the related literature, the use of Drop-out with Batch Normalization is discouraged [Goodfellow et al., 2016].

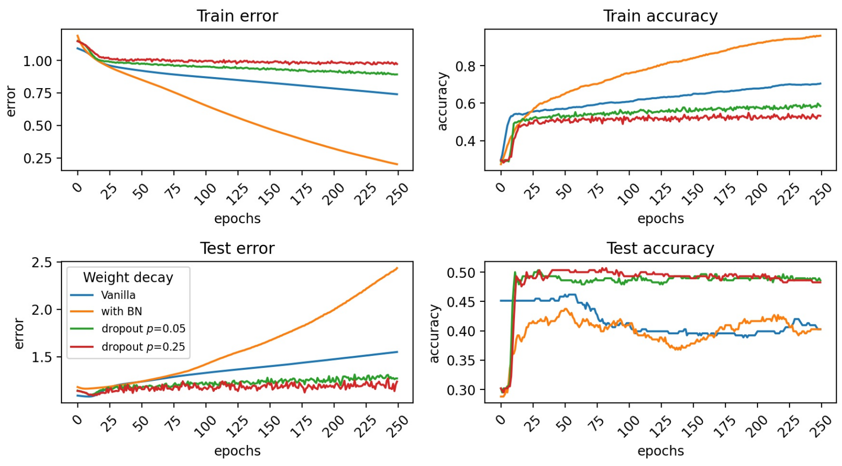


Figure 7.10: Error and accuracy for train and test experiments on drop-out.

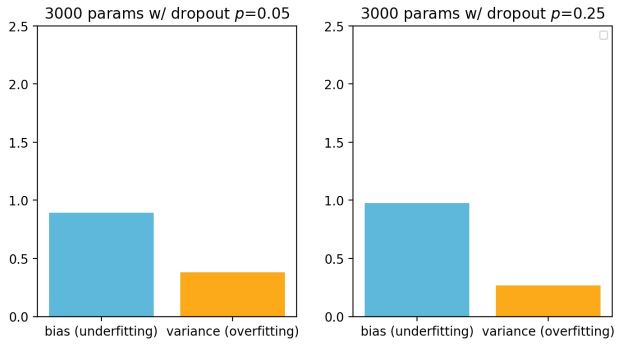


Figure 7.11: Bias-variance decomposition of trained non-BN and BN 3000 parameters Neural Networks.

We have performed several experiments with two values of p: 0.05 and 0.25. As it can be seen in Figure 7.10, this technique also accelerates the learning of the model, we can see how the accuracy increases enormously in the first 25 epochs. Furthermore, in this case we see improvements in the accuracy compared to the experiments with Weight decay, in which the learning was irregular and erratic, here it is much more regular, faster and also the model manages to reach a success rate of 50% in the test. The decrease in overfitting in the Drop-out experiments (Figure 7.11) compared to the BN experiments (Figure 7.7) is also significant.

Up to now, several techniques have been shown to correct for underfitting and overfitting in training. However, these exploratory experiments can still be improved as they have been performed independently and with many hyperparameters kept as defaults.

These experiments have been useful to get an idea of what impact these techniques can have on our model and how it reacts to the addition of new functionalities. In order to further optimise our model, we have now performed a search for the best hyperparameters for this dataset. Note that in these experiments we will also apply the BN, Drop-out and weight decay, in order to see how they interact with other hyperparameters.

7.2.4. Event Based Dataset: hyperparameter optimization

To perform this search for optimal hyperparameters, we have used the Random Search technique, which consists of defining a limited space of hyperparameter values and randomly choosing values from this space. This is a good method for discovering unintuitive hyperparameter combinations and values that with Grid Search would require more search time (Figure 7.12). With Grid Search it would take a long time to cover all values and we could be ignoring a value that minimises or maximises the objective function. With Random Search it is more likely to cover more values in the space.

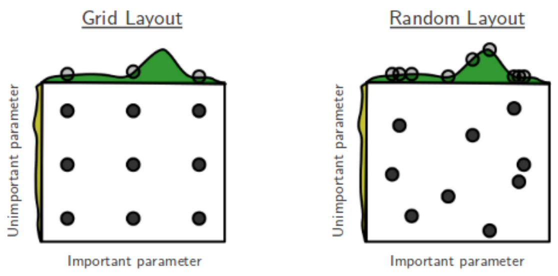


Figure 7.12: Difference between Grid Search and Random Search. The green distribution plot shows how a parameter maximize an objetive function.

In this first Random Search experiment we seek to optimise the hyperparameters (Figure 7.4) of the ANN using the functionalities we have mentioned so far: BN, weight decay and Drop-out. We cannot show, let alone explain, every single experiment done during

Hyperparameter	Search space	Data type	Constraints
Type of net	Vanilla ANN, BN ANN or Drop-out ANN	categorical	
Units per layer	[3-100]	integer	
Layers	2, 3, 4, 9	integer	non-including output layer
Drop-out p	(0, 0.2]	float	only in drop-out
Optimizer	ADAM or Stochastic Gradient Descend (SGD)	categorical	
Momentum	[0-1]	float	only in SGD
Weight decay	[0-1]	float	1/2 choices will be 0
Nesterov	True or False	Boolean	>0 momentum and 0 dampening
Dampening	[0, 0.5]	float	
Learning rate	0.00001, 0.0001, 0.001, 0.01, 0.1	float	
Betas	([0, 1.0], [0, 1.0])	tuple	only in Adam
Scaler	Normalizer, MinMax, MaxAbs, Standarize	categorical	
Batch size	16, 32, 64, 128	categorical	

Table 7.4: Hyperparameters to optimize with Random Search.

Random Search. However, we can summarise the execution of all the experiments by aggregating different statistics of the results obtained. Therefore, we are going to talk about the hyperparameters (Figure 7.4) that have the most impact on the performance of the model.

In Figure 7.13 we can see the hyperparameters with the best test accuracy. This plot

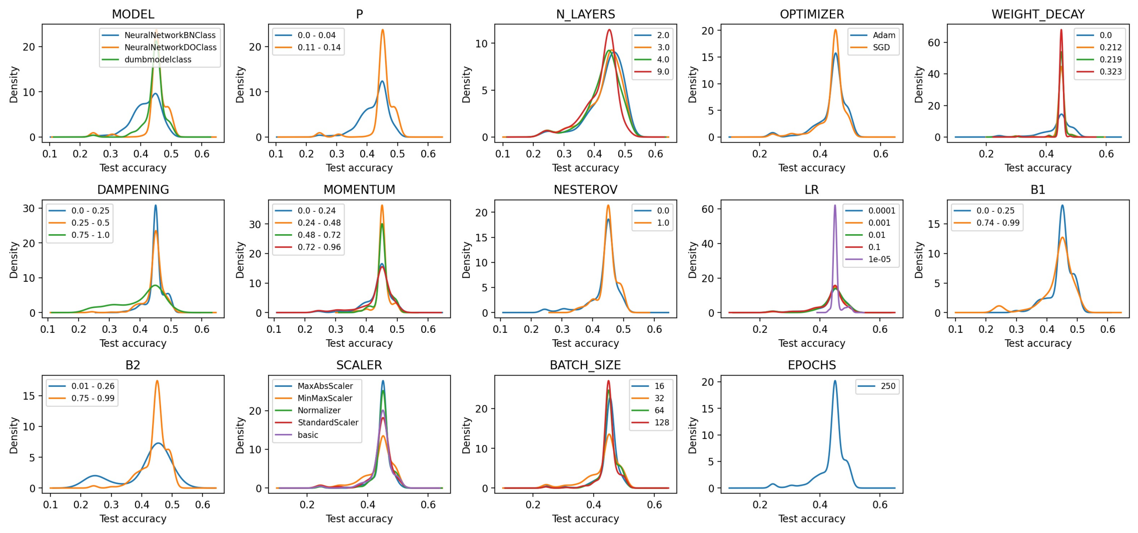


Figure 7.13: Density distribution of test accuracy in this Random Search for each hyperparameter and value.

shows the density distribution of accuracy for each hyper-parameter value, thus it takes into account all the experiments performed. We can see that practically all experiments are around 45% correct in the test. However, there are a few experiments that reach and pass the 50th percentile. It is the Drop-Out models that generally achieve the best results, although the Vanilla ANN (dumb model class) also obtains good results, far behind are the BN models. Of the Drop-out models, the best values for p seem to be clearly between 0.1 and 0.2. On the other hand the number of layers of the model is quite important and better results have been obtained with ANNs with few hidden layers (2 and 3).

Another important hyperparameter is the learning rate, which although all its values behave similarly with respect to the accuracy, it is true that a very small LR makes it more difficult for the ANN to learn, as we can see that for an LR of $1e-5$ practically all the experiments have the same accuracy. The weight decay does not obtain good results, except with values of practically 0. The rest of the hyperparameters do not seem to significantly condition the accuracy of the model.

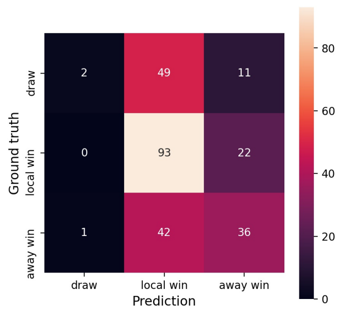
Model	Architecture	Optimizer	Lr.	Train err	Train acc	Test. err	Test acc	F1 draws	F1 home	F1 away
Drop-out	73-53-21-3	SGD	0.1	0.947	**0.546**	1.194	**0.512**	0.061	0.622	0.487
Drop-out	53-21-3	SGD	0.001	0.989	0.518	1.171	0.510	0.077	**0.630**	0.444
Drop-out	68-4-7-3	SGD	0.1	0.947	**0.546**	1.195	0.508	0.031	0.622	0.483
Drop-out	34-11-56	Adam	0.00001	0.992	0.512	**1.123**	0.508	0	0.619	**0.509**
Drop-out	42-18-18	Adam	0.001	**0.934**	0.544	1.201	0.508	**0.264**	0.621	0.428

Table 7.5: Top 5 best configurations on the Random Search.

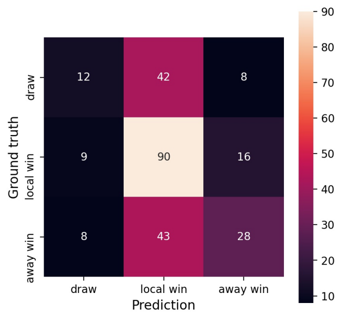
We have so far analysed how hyperparameters impact model performance and which values are optimal. We have also applied solutions to reduce underfitting and overfitting in order to get the best test results. If we check which are the best results we see that the best results have little overfitting, in fact the top 5 are trained with Drop-out (Table 7.5). However, there is one aspect that we have not yet analysed and that is, which matches the model gets right and which ones it misses, and what is the reason for this.

F1-score is a very appropriate metric to evaluate the results of a model trained with an unbalanced dataset, and as we have already mentioned in a previous chapter, our dataset is not balanced (Table 6.5). The F1-score is a metric that tells us the relationship between the Recall and Precision of a model. The accuracy of a model indicates whether what you have predicted is correct, i.e. the ratio of True Positives to all Positives (True + False). On the other hand, the Recall indicates whether you have predicted the samples of a class correctly, i.e. the ratio of True Positives among all the samples of that class (True Positives + False Negatives). F1-score is the harmonic mean of both metrics. In conclusion, F1-score indicates whether the model knows how to differentiate correctly between the different classes.

We can see in Table 7.5 that despite having a high accuracy in general (>50%) the F1-score in the draws is almost 0 or directly 0. Let's compare the difference between the model with the highest accuracy and the fifth with the highest accuracy, which is also the one with the highest F1-score. We can see in Figure 7.14 how the first model, despite having the best overall accuracy, is really bad at predicting draws. The distribution that the predictions follow does not match the actual values at all (Table **??**). Only 1% of the predictions are draws compared to 24% of the actual values. We can see that when it doubts between home win or draw, it always goes for the first one, because there is no case of home win that it has predicted as a draw. In the second model, we see that it considerably improves the draws and increases to 11% of the total predictions. However, this model is still far from 24% of draws.



(a) 73-52-21-3 SGD model with 0.512 test accuracy



(b) 42-18-18-3 Adam model with 0.508 test accuracy and 0.264 F1 score in draws

Figure 7.14: Confusion matrix for two ANN with different accuracy in draws.

(%)	**Draws**	**Home wins**	**Away wins**
Ground truth	24	45	31
Predictions model 1	1	72	27
Predictions model 2	11	68	21

Table 7.6: Distribution in percentage (%) of each class in the test set labels and in the test predictions for models 1 (73-52-21-3) and 2 (42-18-18-3).

We can see that the models have problems when it comes to identifying draws (Recall), as they only predict at most 20% of these. The models under-predict draws, so solutions must be found to ensure that the model predicts draws better without penalising home and away wins.

For this reason, two solutions have been proposed. The first is related to simplifying the training for the model, and consists of applying a series of dimensionality reduction or feature selection techniques to the data. We will do a series of experiments applying Variance Threshold and ANOVA to select the best features, PCA to transform the data into a reduced dimensional space and we will do some experiments with a series of features selected manually after the analysis of the data (Table 6.15).

7.2.5. Event Based Dataset: dimensionality reduction

As we have mentioned, some approaches have been proposed to simplify the training of the model because of the dimensionality reduction. Now, the model will be able to focus in the most important features to find patterns in them, so the optimization may be more efficient. The experiments have also been performed with hyper-parameter optimization with Random Search (Table 7.4).

Figure 7.15 and Table 7.7 show how the model tends to make better predictions with inputs of larger dimensionalities as it provides more information to the model. However, the methods used to select the best features do not all obtain the same results. PCA obtains the best results in all the different dimensions to which the input data has been reduced. Especially with 15 dimensions we obtain the best accuracy of this experiment. Very close is the accuracy obtained training with the 20 features (all available features). We also mention the good result obtained with the manually selected features after the data analysis, which validates in a certain way the analysis made in a previous chapter about the importance of the variables. Finally, the worst results obtained with the method of selecting the features that exceed v variance threshold, Variance Threshold.

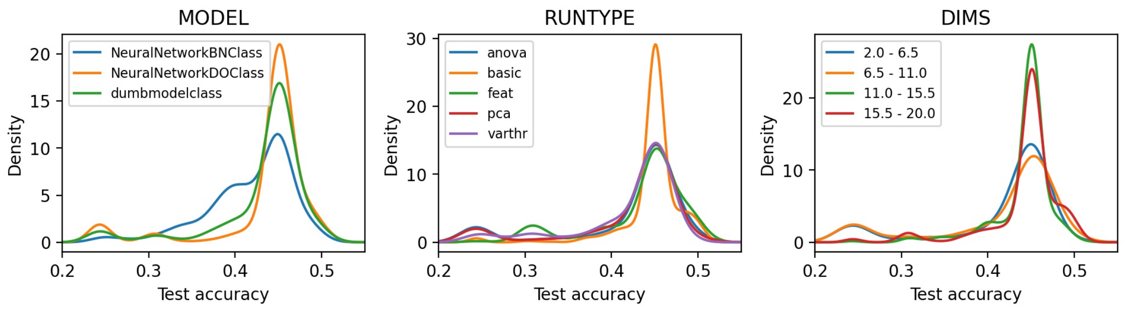


Figure 7.15: Test error distribution for each hyperparameter and value in feature selection experiments.

Method \ Dimensionality	5	10	12	15	20
All features	-	-	-	-	0.512
Manual selection	-	-	0.507	-	-
Variance Threshold	0.457	0.473	-	0.49	-
ANOVA	0.487	0.5	-	0.5	-
PCA	0.512	0.510	-	0.516	-

Table 7.7: Results of feature selection experiments. This table shows the higher accuracy reached for each method employed and new dimensional space of data.

The accuracy have not been significantly improved with the application of this techniques, and the model seems to perform better with high dimensionalities. For this reason, we will continue training the model with all the features of the data, although we will do some experiments applying the most successful methods, ANOVA and PCA.

In addition, in order to try to balance the distribution of predictions to that of the labels, we have carried out a series of experiments focused on finding the optimal weights assigned to each class in the error function.

7.2.6. Event Based Dataset: weighting loss-function

When calculating the error during forward-propagation, each class can be weighted with a weight $w(c)$, in order to penalise some classes more than others (Eq. 4.3). In this case, a higher weight is assigned to the draws so that it penalises the error in the draws more and learns more from the error it commits. This is intended to make the model more familiar with the draw and more sensitive to it. These weights have also been optimised with Random Search, optimising the other hyperparameters as well (Table 7.4).

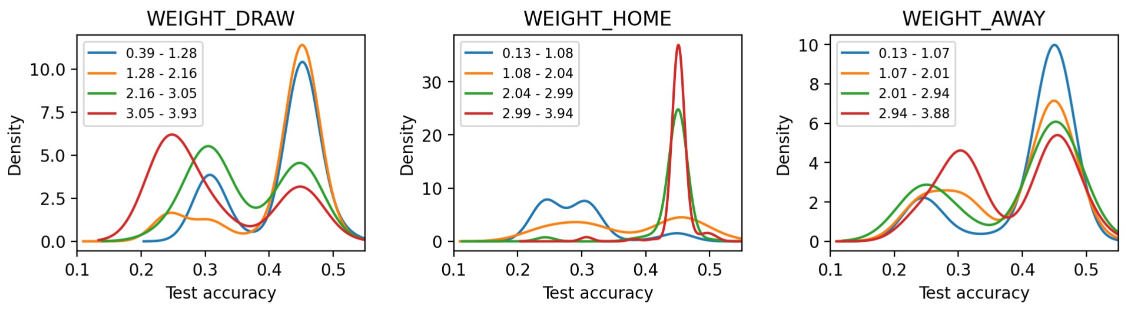


Figure 7.16: Test error distribution for each class loss-weight.

We can see in Figure 7.16 how these parameters have a great impact on the accuracy of the model. However, the relationship between them is complex because they interact with each other. When we assign a weight to a class, it not only influences that class but also the training of the other classes. Remember that the labels we pass to the CE loss (our error function) are One-Hot Encoded vectors and therefore in each match only the error in the target class is penalised even though the ANN gives as output a probability to each class. In other words, if the match is a home win, only the error in the output of the home win will be penalised, but the probability of a draw or away win will not be penalised and the ANN will not learn from it. With the default setting (weights of 1 for each class) all classes are penalised equally.

However, by having an unbalanced dataset, in practice, the error in the output of local wins will be penalised on many occasions, and conversely the output of away wins and draws on very few occasions. This causes the model to learn more in one direction than in others, i.e. it is biased by the large number of home wins. If you also add that draws are usually very tight matches and that, as there are fewer correlations of the features with the result, there is less separability in the dimensional space of the features, it makes it very difficult for the models to learn and guess the draws.

With the aim of rebalancing the error, different combinations of weights have been tested for each class. We see in Figure 7.16 that high values for the draw penalty cause a significant drop in the overall accuracy of the model (∼25%). This is because in this case we are forcing large changes in its weights when it commits a draw miss, biasing the model's learning towards over-predicting draws. Figure 7.17 shows the difference in gradients with different values of loss-weights. Therefore, the value of these hyperparameters also affects the performance of the model in terms of robustness and stability of learning. When we say that there are strong interactions between the three hyperparameters, it is not only the absolute value of the error weight that matters but the ratio or difference between the three (Figure 7.18). If we apply three small ratios but one class has half as much penalty as the others, it will be penalised half as much as the others and therefore the model will potentially predict fewer matches of that class.

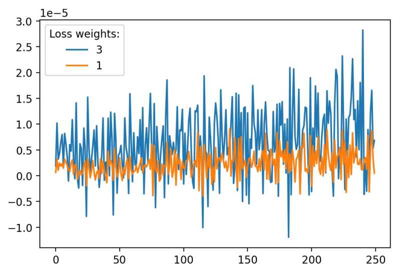
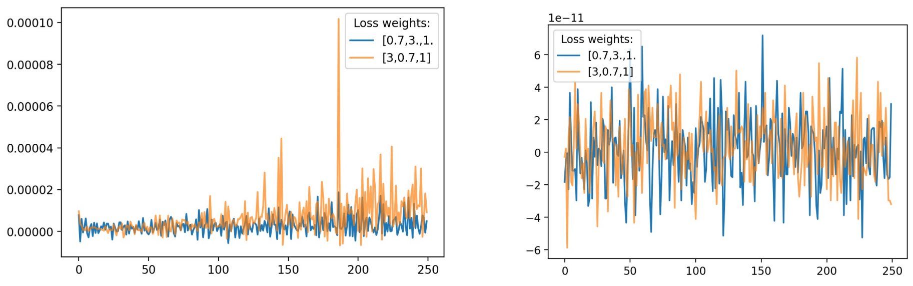


Figure 7.17: Input layer gradients during training with loss-weights of 1 and 3 for all the classes.



(a) Input layer gradients. **(b)** Output layer gradients .

Figure 7.18: Loss-weights focusing in home wins (blue) and draws (orange).

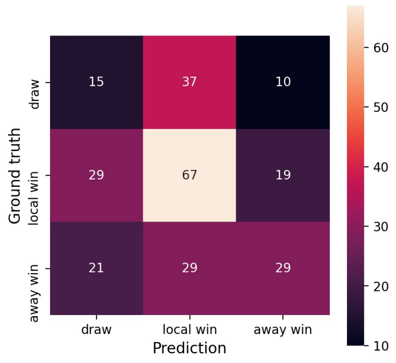


Figure 7.19: Confusion matrix of a model with the following loss weights: draw: 1.2, home win: 0.8 and away win: 1.0.

Finally, the weights that have given the best results in improving the prediction distribution have been those around 1. By using a slightly higher weight for draws than for home and away wins, the model also improves the prediction of draws (Table 7.8), but slightly reducing the overall accuracy of the model (Figure 7.19).

So far, we have trained ANN models that have learned some patterns and relationships from the data in order to make accurate predictions. We can confirm that comparing the results with the baseline accuracies (Table 7.1) that were up to 10% below our best trained model accuracy.

(%)	**Draws**	**Home wins**	**Away wins**
Ground truth	24	45	31
Loss weights: 1.2 - 0.8 - 1.0	25	52	23
Loss weights: 1.1 - 0.9 - 1.0	6	54	40
Loss weights: 1.2 - 0.7 - 1.1	24	34	42

Table 7.8: Distribution in percentage (%) of each class in the test set for different loss weight values.

Morever, we have done some experiments to apply new approaches and find optimal configurations in order to enhance the model performance. We have applied solutions to reduce overfitting and underfitting, solutions to improve the performance of the model so that it learns faster and more stably in training, and we have also applied solutions to make the model make better and more varied predictions. We have improved the results from the first experiments where the models were 48% correct, whereas now we have reached 51.6% correct. We have also improved the accuracy of draws, where the first models had an F1-score of approximately 0, i.e. almost ignoring draws. We have now achieved an F1-score on draws of 0.26 and have managed to fit the distribution of predictions to the distribution of labels. However, we believe that we can obtain even better results, both in terms of accuracy and class separability.

Given the limitations offered by this dataset in terms of number of records to train and test, we have created a new dataset (6.3.2) with new features and 10 times more records than the one used so far. With more data and therefore variability, we expect the model to be able to learn new patterns and relationships in the data.

7.2.7. Historical Dataset: testing different dataset versions

The purpose of the following experiments is to improve the results of our previous experiments by training our model with a larger dataset and other information. As mentioned in a previous chapter, the Historical Dataset contains data about the short and long term form of the two teams playing a match. With this, it is intended that the model has not only information from the last n matches but also from months and years ago. In the previous model we could have the case of a team, like Chelsea or Bayern Munchen, coming from a bad streak and the model would automatically give the winner to the rival team as it only looks at the last n matches. Now the model will also take into consideration the last months and years of both teams.

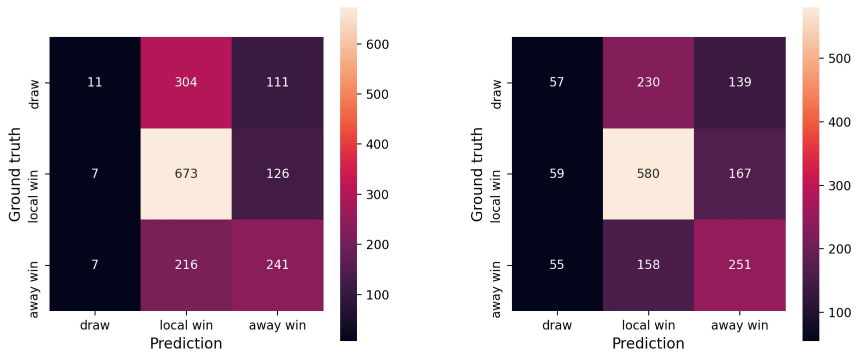
As also mentioned above, we have several versions of this dataset, which we will compare. For this purpose, as in the experiments with the Event-Based dataset, we will perform a Random Search to find the optimal hyperparameters of the models trained with these datasets. The hyperparameters to be optimised are the same as in the previous experiments (Figure 7.4). In addition, the weights of each class will also be optimised in the model error calculation. Now, on the other hand, the trainings will last 100 epochs instead of 250 as with the previous dataset. It is because now the dataset is bigger and it is not necessary to iterate over it so many times, and also because it avoids the execution time to be extremely long.

As we can see in Table 7.9 the datasets with long-term information have a significantly higher accuracy. In fact, the two Longterm datasets achieve an accuracy that we have not achieved so far. With the Longterm Complete dataset (accuracy of 54.5%) even the accuracy of the bookmaker's odds benchmark is reached (Figure 7.2). The left graph in

Dims. / Method	Historical Complete				Hist.Longterm Basic				Hist. Longterm Complete			
	3	**8**	**15**	**23**	**5**	**15**	**25**	**55**	**10**	**25**	**50**	**140**
All features	-	-	-	0.495	-	-	-	0.521	-	-	-	**0.545**
ANOVA	0.488	0.486	0.492	-	0.502	0.511	0.516	-	0.512	0.526	0.526	-
PCA	0.491	0.49	0.495	-	0.501	0.503	0.512	-	0.512	0.514	0.515	-

Table 7.9: Comparison of the accuracy of the Random Search experiments over the three experiments. This table shows the higher accuracy reached for each method employed with each new dimensional space of data and each Dataset.

Figure 7.20 shows the confusion matrix of the model with 54.5% accuracy.



(a) Model A with the highest accuracy. (F1-score Draws = 0.05)

(b) Model B with high accuracy and high F1-score in draws. (F1-score Draws = 0.2)

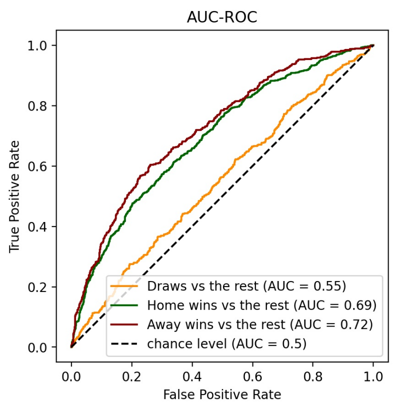
Figure 7.20: Confusion matrix for two models trained with Historical Longterm Complete dataset.

In the figure above we also show the confusion matrix of another model trained on the same data (Historical Longterm Complete) but which manages to predict accurately more draws. In the AUC-ROC graph (Figure 7.21) we can better see the difference between the two models when differentiating between classes. We see that the difference is smaller than what is observed in the confusion matrix or in the F1-Score, in fact the AUC index is practically the same in all three classes. This is because the AUC-ROC is calculated with the probabilities output of the model.

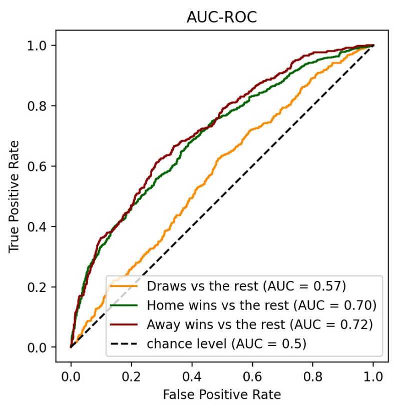
Div.	**Season**	**Match**	**Res.**	Model A				Model B			
				Proposal	**pred D**	**pred H**	**pred A**	**Proposal**	**pred D**	**pred H**	**pred A**
FRA	2019-20	Reims - Nice	0-0	D	0.35	0.34	0.31	A	0.33	0.25	0.42
GER	2008-09	M'gladbach - Wolfsburg	0-0	H	0.34	0.37	0.29	D	0.36	0.3	0.34
GER	2014-15	Frankfurt - Hoffenheim	3-0	H	0.34	0.35	0.3	D	0.36	0.3	0.34
ITA	2018-19	Juventus - Sampdoria	1-1	H	0.23	0.6	0.17	H	0.24	0.62	0.14
ITA	2020-21	Parma - Fiorentina	0-0	H	0.28	0.41	0.31	D	0.36	0.32	0.32
SPA	2011-12	Villareal - Barcelona	0-0	A	0.24	0.22	0.54	A	0.33	0.24	0.43
SPA	2007-08	Real Madrid - Levante	3-0	H	0.25	0.61	0.14	H	0.24	0.62	0.14

Table 7.10: Examples of outputs of model a) and b) with each actual match outcome, each model proposal and probabilities for each class.

Table 7.10 shows several examples of the difference in predictions between the model with the highest accuracy (Model A) and the one with F1-score Draw = 0.2 (Model B). As we can see the predictions are not very different except for certain matches like Parma vs Fiorentina. This is the reason why the ROC curve of both models is not so different. However F1-score shows a bigger difference between both models because it is based on the accuracy. The accuracy is calculated in a Boolean way (hit or miss, 1 or 0) so the



(a) Model A with the highest accuracy. (F1-score Draws = 0.05)



(b) Model B with high accuracy and high F1-score in draws. (F1-score Draws = 0.20)

Figure 7.21: AUC-ROC for two models trained with Historical Longterm Complete dataset.

intermediate information (the intermediate greys) that show the probabilities are lost. As we can see in the M'gladbach vs Wolfsburg in the second model the accuracy is very low, while in the first model it is very low, being the error made insignificant, however the accuracy penalises you in the same way. As we know that the probabilities given by an ANN are not totally accurate (as a probability), we could define a Confidence Interval that marks the margin of error of our predictions. In this way we would see if within the Confidence Interval the model would get the outcome of the match right or not.

Match	**Res.**	**15_H**	**15_A**	**60_H**	**60_A**	**1825_H**	**1825_A**	Historical Complete				Historical Longterm Complete			
								Proposal	**D**	**H**	**A**	**Proposal**	**D**	**H**	**A**
Everton - Man Utd	1-2	0	0.5	1.86	1.16	1.39	1.79	H	0.21	0.62	0.17	A	0.33	0.24	0.43
Elche - Real Madrid	0-0	3	1.5	1.86	2.25	1.03	2.43	A	0.26	0.36	0.37	A	0.34	0.23	0.43
Bremen - Hertha	2-0	3	1.5	2.5	1.12	1.17	1.32	H	0.14	0.77	0.09	H	0.34	0.42	0.23
Valladolid - Barça	0-2	2	2.3	1.37	1.55	1.08	2.28	H	0.27	0.38	0.35	A	0.34	0.21	0.45

Table 7.11: Examples of how lags influence output probabilities. 15_x,60_x and 1825_x are the points ganined for the last n days for each team, and D, H and A are the probabilities given in the output by each model.

On the other hand, the addition of longterm features has provided the model with valuable information that it did not have until now, namely the history and status of a club. With the lags of matches over the last year or even several years we give the model information about the success a team has had in the long term. Now the model has to learn how to relate the short term variables (several weeks or months) to the long term variables (many months or years). If we refer to the accuracy of the models with both types of data we see that without longterm the model achieves an accuracy of 49.5% while with longterm variables we reach 54.5%.

Table 7.11 shows some examples of how the predictions of a model trained only with short term variables (Historical Complete) and another one that uses both types of variables (Historical Longterm Complete) change. Until now, the models only received as input the variables 15_x and 60_x and relied on them to predict. We see that, in this way, matches such as Valladolid vs Barcelona would be won by the home team because both teams have a similar recent streak, and probably give the victory to Valladolid because of the home factor. Instead, we see that by adding long term streak information the model knows that if it is even in the short term, the historically better team is more likely to win. Even so, we see that the model gives a 33% probability of a draw, which makes perfect sense. We see in other matches such as Elche vs Real Madrid or Werden Bremen vs Hertha that even if the final prediction does not change, the probabilities are more consistent with the lag of 1825_x.

We have already mentioned that our model only penalises the error committed in the target class (i.e. if the match is an away win, only the output of the away win is penalised). What we have not yet mentioned is that, in addition, all outcomes are penalised equally regardless of the goal difference, i.e. it is equally penalised to make a mistake in a 2-1 as in a 5-0. In order to force the model to learn more efficiently, and to penalise its most serious errors, we have proposed a modification of the label that is passed to the error function.

7.2.8. Historical Dataset: weighting error depending on the goal difference

The new label would be the result of multiplying the One-Hot Encoded vector of the original label by the goal difference of the match. In case of a draw (the difference is 0) the original label is left. It is true that, in this way, home and away wins will be penalised (when there is more than one goal difference) more than draws, which will harm the predictions of draws. For this reason, it has also been proposed to scale with a MinMax function between 0 and 1 the predictions of each class. In this way, draws will always have a penalty of 1, while home and away wins will be scaled (between 0 and 1) according to the goal difference.

Finally, as there are many matches with low goal difference, which could make the error generally very low and the model learn slowly, it has been decided to apply a multiplicative factor f. This factor multiplies all scaled columns to increase the error by a factor of f. This hyperparameter f must be optimised like any other hyperparameter. Being n the number of matches of the training set, c the target class, being three the number of classes, t the matrix $(n \times 3)$ with the original One-Hot Encoded of the labels of the training set, t' the new matrix of the labels of each match $(ntimes3)$, d the vector of goal difference in the matches of the training set $(n \times 1)$ and f the multiplier factor (scalar), the transformation applied is the following one:

$$t'_c = \underset{c}{minmax}(d \cdot t) \cdot f \tag{7.4}$$

Factor \ Test metrics	Hist.Longterm Basic			Hist. Longterm Complete		
	error	**accuracy**	**f1-score**$_{draws}$	**error**	**accuracy**	**f1-score**$_{draws}$
no factor f (original)	0.98	0.521	0.2	0.98	0.545	0.15
label $\cdot$ goal diff. (no f)	1.9	0.52	0.02	1.88	0.523	0.02
$f = 2$	0.17	0.519	0.3	0.14	0.514	0.29
$f = 5$	0.34	0.516	0.31	0.27	0.516	0.29
$f = 10$	0.67	0.519	0.3	0.55	0.52	0.31

Table 7.12: Results of factor experiments. It shows the model with the minimum test error and maximum test accuracy and also the mean of the f1-score of each factor f. The first two experiments are the original label format and the label multiplied by the goal difference of the match (with no scaling or factor).

In Table 7.12 we can see the results of the Random Search optimisation of models with different factors f. We see that the factor has a considerable impact on the prediction of draws. By applying a factor of 2.5 or 10 the average f1-score of the draws increases by 50% compared to the original form of the labels. As we predicted before, if we don't scale the labels once multiplied by the goal difference, we will have very large labels for matches that end in goals, especially compared to the draw labels (which are always 1), and therefore the error function will give a huge error for those matches, biasing the learning and hurting the draw predictions (which will not learn anything in comparison). As we can see, this has come true and the f1-score for the draws is only 0.02, and the error is the highest of all experiments (~ 1.9).

It is interesting to see how the results are very similar with the two datasets, which makes sense since this modification only affects the labels and the calculation of the error and not the features. The accuracy on the other hand does change more significantly, in the Basic dataset the accuracy decreases very little from the original version to that of the factors, however, in the Complete dataset we see that there is a decrease of up to 3% in the accuracy. Despite this, the decrease in accuracy is compensated by more balanced and distributed predictions.

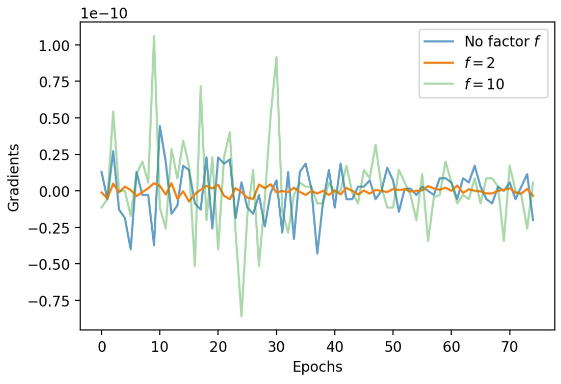
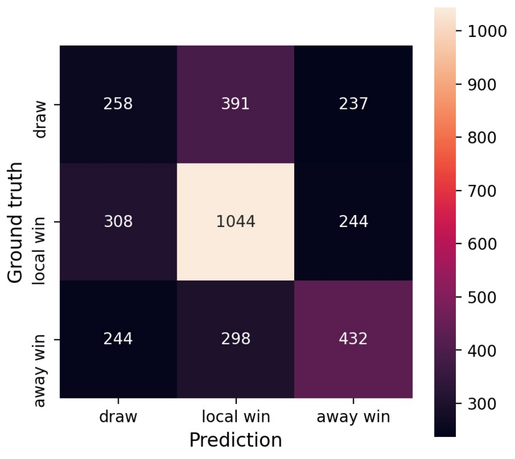
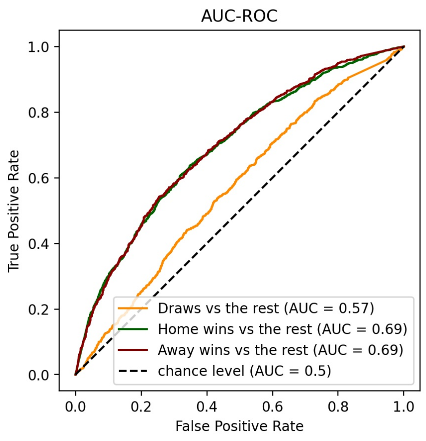


Figure 7.22: Gradients during ANN training for 3 different values of factor f.



(a) Confusion matrix



(b) AUC-ROC

Figure 7.23: Best model with $f = 10$ with accuracy of 0.519 trained with Historical Longterm Complete dataset.

As we have explained, the value of the factor f strongly influences the value of the error. The two values are directly proportional; for large f there will be a larger error than for small f. The magnitude of the error has an impact on the update of the ANN weights during back-propagation. Figure 7.22 shows the gradients of a Hidden layer for different values of f. As can be seen, choosing a value of f that is too small can cause the error to be very small and therefore the gradients to be negligible. This problem propagates throughout the ANN to the previous layers and is known as the Vanishing Gradient problem. The consequence of this is a model that will learn much more slowly as the gradients that will reach the neurons when updating will be practically 0.

Figure 7.23 shows the model with the best F1-score in the draws (0.3). It is a model trained on the Historical Longterm Complete dataset, with a factor $f = 10$. The ANN has 3000 parameters in 4 layers, it has been trained with Adam as optimiser and regularised with Drop-out with $p = 0.07$, it has also used weights in the loss-function (draws = 1.1, home = 0.9 and away = 1.0), and the training has been 100 epochs with a Batch size of 32 samples. All features have been used for training (140).

If we analyse some examples of the predictions in the test of our original model and a model trained with $f = 10$ (Table 7.13) we can see that the differences are significant. As we have said, increasing the factor implies increasing the error committed by the model and therefore forcing the model to make larger changes to correct these errors. In these examples we see how with a high factor the model's decisions are more robust, there is more difference between the different probabilities it draws in the output, i.e. it is more certain of the forecast it makes. At the same time, by penalising more in the draws, we see that it balances the predictions better, following a distribution more similar to that of the labels.

Match	**Res.**	**15$_H$**	**15$_A$**	**60$_H$**	**60$_A$**	**1825$_H$**	**1825$_A$**	NO Factor (original)				Factor $f = 10$			
								Proposal	**D**	**H**	**A**	**Proposal**	**D**	**H**	**A**
Blacburn - Arsenal	1-0	1.5	0.5	2	1.33	1.43	2.02	H	0.31	0.36	0.33	D	0.37	0.3	0.33
Racing - Zaragoza	0-0	0.5	2	0.88	1.5	1.26	1.24	H	0.30	0.36	0.34	D	0.38	0.3	0.32
Milan - Roma	0-1	0	3	1.33	1.88	2.15	1.87	H	0.28	0.54	0.19	D	0.367	0.368	0.265
Betis - Real Madrid	0-3	0.5	3	0.33	2.67	1.18	2.4	A	0.25	0.12	0.63	A	0.24	0.04	0.72

Table 7.13: Examples of how lags influence output probabilities. 15_x,60_x and 1825_x are the points ganined for the last n days for each team, and D, H and A are the probabilities given in the output by each model.

Goal difference	**NO Factor**	**Only goal difference**	$f = 1$	$f = 5$	$f = 10$
0	0.13	0	0.29	0.28	0.29
1	0.57	0.64	0.49	0.49	0.49
2	0.68	0.7	0.58	0.59	0.59
3	0.77	0.81	0.72	0.72	0.72
4	0.86	0.84	0.75	0.74	0.73
5	0.95	0.98	0.98	0.96	0.98
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1

Table 7.14: Test accuracy comparison between a model with no factor and high f1-score for draws (0.2) and models with different f values. All the models have been trained with the same hyper-parameters and data.

In Table 7.14 we actually see the impact of the factors according to the goal difference of the match. We see that there is no significant difference between the different values of f. In the draws it doubles the accuracy of the original model (without factor), although this improvement in the draws is especially penalised in the matches with 1, 2 and 4 goals difference where it has 10% less accuracy. It is also interesting to see how in all the variants practically all the results of the 5 or more goals difference scores are correct.

At this point, we have finished with the experiments based on ANN. These have been the main experiments of this work, however, we believe that it is interesting to perform experiments with another Machine Learning algorithm to verify that we are obtaining the best possible results with ANN. Therefore, we will now present a series of experiments performed with the LightGBM algorithm.

7.3 LightGBM Based Experiments

LightGBM is an open-source tree-based ML algorithm based on Gradient boosting that is currently widely used for data-intensive problems. LightGBM is increasingly used ahead of other Gradient Boosting algorithms for its speed and performance on classification and regression problems. In fact, it has recently achieved excellent results on tabular data, even ahead of ANN and other more classical ML algorithms. For these reasons and because it was more familiar to us, we have chosen this algorithm for our experiments.

This tree-based algorithm obtains state-of-the-art results on tabular data because it is able to handle both numerical and categorical data. In traditional machine learning models, categorical variables are typically converted into numerical variables through techniques such as one-hot coding. This can lead to high-dimensional and sparse feature spaces, which can have a negative impact on model performance and slow down training times. However, LightGBM uses a unique algorithm that can directly handle categorical features by creating a histogram-based split instead of using one-hot coding. This results in more efficient and accurate models that can handle high-dimensional and sparse feature spaces with ease.

Hyperparameter optimisation is performed using the Optuna framework. This framework optimises by creating combinations of hyperparameters, which are evaluated using an objective function that measures the performance of the model. Optuna uses this evaluation to adjust the sampling strategy and focus on the hyperparameter combinations that look most promising. As more tests are performed, Optuna refines its search until it finds the combination of hyperparameters that maximises the model's performance. The search strategy it uses is called Bayesian search.

In Bayesian search, a probability distribution is used to model the search space of the hyperparameters. As more tests are performed, the probability distribution is updated with the results of previous tests, allowing the search to focus on the most promising areas of the search space.

Optuna's Bayesian search strategy is based on the Tree-structured Parzen Estimator (TPE) algorithm [Akiba et al., 2019], which is an efficient and scalable method for hyperparameter optimisation. TPE uses two probability models to model the promising and unpromising regions of the hyperparameter search space, and focuses on the most promising region to select the next hyperparameter combination to be tested.

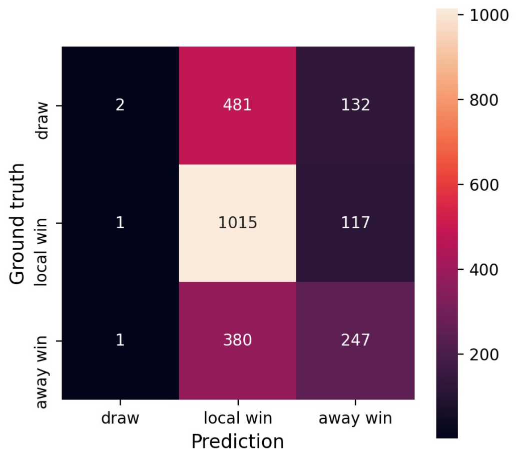
In these experiments, analogous to ANN experiments, we seek to minimise the test error. Furthermore, the experiments have been performed with the two datasets that have given the best results with ANNs (Historical Longterm Basic and Historical Longterm Complete), and we will perform experiments both with the complete dataset and applying ANOVA and PCA.

Table 7.15 shows the results of the LightGBM experiments. As we can see LightGBM achieves a really good accuracy with both datasets, but contrary to ANNs, it performs better with the Historical Longterm Basic dataset. Moreover, it seems that selecting the best features according to ANOVA improves the performance of the model compared

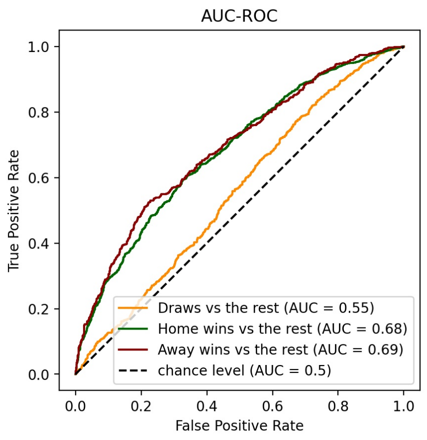
Model \ Test metrics	Hist.Longterm Basic		Hist. Longterm Complete	
	error	**accuracy**	**error**	**accuracy**
Basic LightGBM	0.99	0.527	0.99	0.525
ANOVA 25 dims	0.997	0.524	0.989	0.525
ANOVA 50 dims	0.992	0.532	0.987	0.529
PCA 25 dims	1.02	0.506	0.1	0.513
PCA 50 dims	1.01	0.512	0.995	0.519

Table 7.15: LightGBM experiments resume.

to using the full dataset. However, LightGBM models do not model draws well, unlike ANNs (Figure 7.24), or at least not on their own (without adding any extra functionality). In fact, the AUC score of the LightGBMs is the lowest we have obtained so far.



(a) Confusion matrix



(b) AUC-ROC

Figure 7.24: LightGBM model with highest test accuracy.

A very interesting feature of LightGBM is that it is more explanatory with its decisions (predictions) than ANNs. In fact, the LightGBM library we use offers a method that shows you the most important and influential variables for the model when making predictions. We see in Figure 7.25 that the most important variables, as we suspected, are the long term variables, especially the average points obtained in the last 6 months, 1 year and 5 years. It is also important to mention how the home team variables have more influence on the LightGBM predictions, most likely due to the distribution of results in favour of home wins. The short-term variables are considerably less important for the model, with only the away team's goals in the last 2 months being among the 20 most important variables. Finally, the variables of goals and points scored by the home team in the last 10 matches between the two teams (derby features) also influence the decision making of the model.

We have seen that LightGBM can obtain really good results in accuracy, simply by optimising hyperparameters and applying an appropriate feature selection. The big limitation we have obtained in the results is a poor performance in draw prediction, as in the first experiments performed with ANN. However, what cannot be doubted is the great potential of LIghtGBM for tabular data.

Now that we have performed experiments with two different Machine Learning algo-

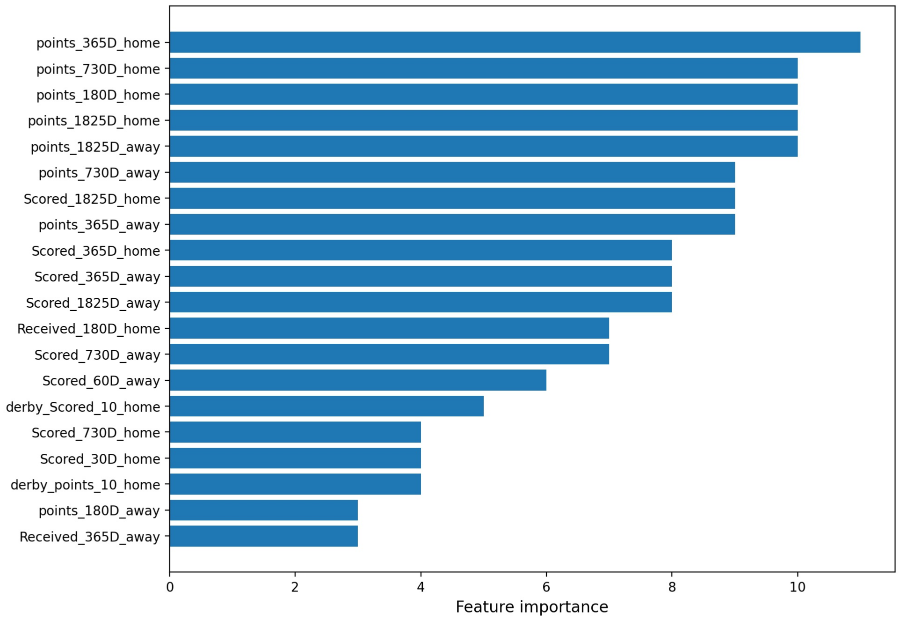


Figure 7.25: Feature importance in LightGBM.

rithms, we will analyse and summarise the overall results of this set of experiments. We will also compare our results with our baseline, the bookmakers' odds.

7.4 Conclusions and benchmark comparison

In this section we will recap the results obtained so far, both with ANNs and LightGBM. In order to draw solid and definitive conclusions, we will analyse the results in detail, seeing where our models are accurate and where they are not. So far we have analysed the results on a fairly general level, except for a few example games shown, now we will analyse the performance of our models at different levels. Finally we will compare our results and predictions with those of the benchmark.

First of all, from the beginning of the experiments we were training models that learn patterns and relationship on the data in order to make predictions. If we compare the accuracy of our models with the one of the two baselines (Figure 7.1), we realize how our model has learned. e can see that there is a significant difference between the accuracy of 42.1% taking only the best team in the last month and the accuracy of 45.6% of a model that only predicts home wins, and the model with the highest accuracy achieved, 54.5%. Specially, comparing with the first baseline (Figure 7.1), that only takes one variable, it can be drawn that our models learn patterns and relationships when they are trained with multiple and proper features.

Now we focus on the comparison between the benchmark and the trained models. Table 7.16 shows the best results for each type of model: ANN with the Event Based dataset, ANN with Historical Short term dataset, two ANN models with Historical Long term dataset, one with uncalibrated predictions (low accuracy in draws) and one with calibrated ones, and finally the LightGBM model. We can see that we can only achieve the accuracy of the bookmakers' odds with one model, the ANN model without calibration.

The same table also shows the respective accuracy scores for each league that appears in the dataset. We can see that although similar results are obtained in each league across

League Model	Overall	ENG	SPA	GER	ITA	FRA
Benchmark (odds)	0.545	0.54	0.54	0.52	0.55	0.5
ANN (event based)	0.5	0.53	0.5	0.52	0.47	0.46
ANN (hist short term)	0.52	0.55	0.54	0.51	0.5	0.48
ANN (uncalibrated)	0.545	0.59	0.52	0.54	0.54	0.55
ANN (calibrated)	0.521	0.54	0.53	0.49	0.53	0.52
LightGBM	0.532	0.57	0.53	0.57	0.53	0.47

Table 7.16: Accuracy in every national league for every algorithm. We show two models for ANN, one uncalibrated (with no balanced predictions) and other calibrated (with balanced predictions).

the different models, there are some exceptions. We can see that in general the English Premier League is the league where we obtain the best results, and we manage to beat the accuracy of the Baseline in up to three models, especially in the ANN uncalibrated where we outperform the bookmakers' accuracy by 5%. The German and French leagues also managed to outperform the Baseline in two models each, also with up to 5% more accuracy. On the other hand, the Spanish and Italian leagues, although they do not have bad results, are not able to beat the accuracy of the Baseline. Although the French league does manage to beat the Baseline with two models, it is the most complicated to predict due to the equality and irregularity of its teams, which is demonstrated by the accuracy in the other three models.

Another interesting aspect to analyse is the accuracy obtained depending on the stage of the season in which the matches are played. The accuracy throughout the season tends to be fairly consistent across the models, with differences of up to 5-7% between the different models. However, there are exceptions, especially in months when the competition resumes after months of stoppage, such as in July, August or September, when a new season starts after a 3 month break. This seriously affects the predictions in general, but especially those of the models without long-term data (Event Based and Historical Short Term). We can see that the accuracy of these models improves as the competition progresses and several rounds have already been played.

Month Model	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Benchmark (odds)	0.53	0.53	0.52	0.53	0.55	0.56	0.55	0.52	0.55	0.55	0.52	0.52
ANN (event based)	0.5	0.52	0.51	0.53	0.52	0.51	0.47	0.38	0.42	0.45	0.51	0.52
ANN (hist short term)	0.51	0.47	0.48	0.53	0.52	0.33	0.5	0.44	0.47	0.53	0.49	0.5
ANN (uncalibrated)	0.6	0.53	0.54	0.54	0.51	0.5	0.75	0.43	0.54	0.5	0.62	0.59
ANN (calibrated)	0.54	0.52	0.49	0.5	0.52	0.63	0.71	0.57	0.46	0.57	0.67	0.57
LightGBM	0.52	0.5	0.53	0.59	0.54	0.47	0.5	0.53	0.49	0.54	0.5	0.54

Table 7.17: Accuracy in month of the year for every algorithm. We show two models for ANN, one uncalibrated (with no balanced predictions) and other calibrated (with balanced predictions).

As would be expected, our models predict matches between opponents of different levels more accurately than between opponents of the same level where the matches are usually close and both teams arrive with very similar statistics. In Figure 7.26 we have classified the teams into different clusters based on points scored over the last 5 and one year.

The classification has been done with a clustering algorithm called K-means with $k = 3$ clusters. The K-means algorithm is a clustering method that groups a set of data into K clusters. The process starts by choosing K random centroids, one for each cluster. Then, each data point is assigned to the cluster whose centroid is closest. After all data points have been assigned, the centroid of each cluster is recalculated and the assignment process is repeated until the centroids no longer change significantly or a maximum number of iterations is reached. The final result is a set of K clusters with centroids representing the data clusters. The choice of $k = 3$ has been practical more than anything else, as we seek to show inter- and intra-cluster accuracy on a graph, and selecting a higher k would give too many combinations.

In our case, in green (cluster 2) we have the best teams, in orange the teams that usually play in Europe but without winning titles and in blue the teams in the middle and lower part of the table.

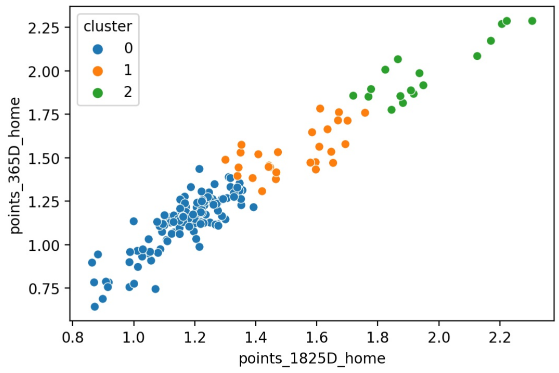


Figure 7.26: Clustering of the teams based on their points in the last 1 and 3 years.

If we take the predictions given by a model with F1-score in draws of 0.2 and calculate the confusion matrices and the accuracy of the matches that have been played between the different clusters (including between teams in the same cluster), we observe (Figure 7.27) that the difference in accuracy from one type of match to another is large. In matches with a large difference in level (cluster 2 vs cluster 0) the model is close to 80% accuracy, although it is true that the model is overconfident in its predictions and ignores anything other than the favourite team winning. As we have seen before, the long-term variables that represent the historical level of a team weigh heavily in the model. On the other hand, the model also overtrusts the home team when the two teams are evenly matched (cluster 0 vs 0 or 1 vs 1). We see that the types of matches in which the model fails the most are those between two very good teams (37% of accuracy), in which there are a large number of draws, almost equal to home wins. In short, we get better results in combinations in which there are fewer draws, especially in which there are more home wins.

Given these accuracies in each cluster, we can conclude that it is really difficult with the data we had to be able to better model matches between teams of similar level. However, we can compare our results with the predictions made by bookmakers, which have really powerful data and tools to make them. With this comparison we will see if our predictions are good and are at the same level or better than those of the bookmakers or, on the other hand, they are far from their results.

In order to make the comparison (Figure 7.28), we have taken the highest prediction of our model for each match (either draw, home or away win) and compared it with the same prediction from the bookmakers. In addition, we show in different colours those predictions that have been correct and those that have not. Finally, a line is displayed separating the predictions which are higher than the odds at below and the opposite at

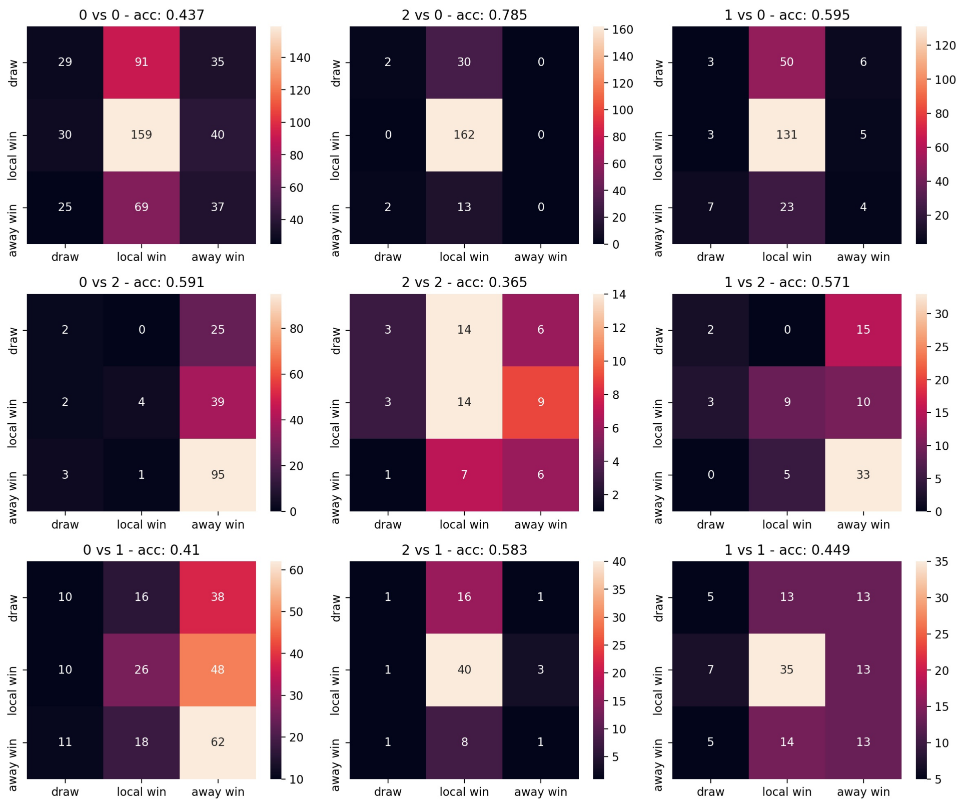


Figure 7.27: Test accuracy and confusion matrix of every type of match.

above. Therefore, if we get the outcome right and the point is below the line it means that our model has given more probability to that outcome than the betting odds, and therefore our model will have made a better prediction.

In Figure 7.28 two models are shown, the top one shows the predictions of the ANN model with the best accuracy and below the model with the best accuracy in draws. We can see that the distribution of the points in the bottom graph is more closely aligned with the blue line, i.e. both predictions are more similar (correlated 1 to 1). In the graph above we can see how the model makes predictions with much lower probabilities than the other model.

(%)	bet > pred.	pred. > bet
Accurate	30	11
Wrong	38	21

(a) Model with highest accuracy.

(%)	bet > pred.	pred. > bet
Accurate	26	16
Wrong	32	26

(b) Model with highest F1-score in draws.

Table 7.18: Difference and accuracy of bets and model predictions.

(mean)	bet > pred.	pred. > bet
Accurate	0.51	0.418
Wrong	0.482	0.428

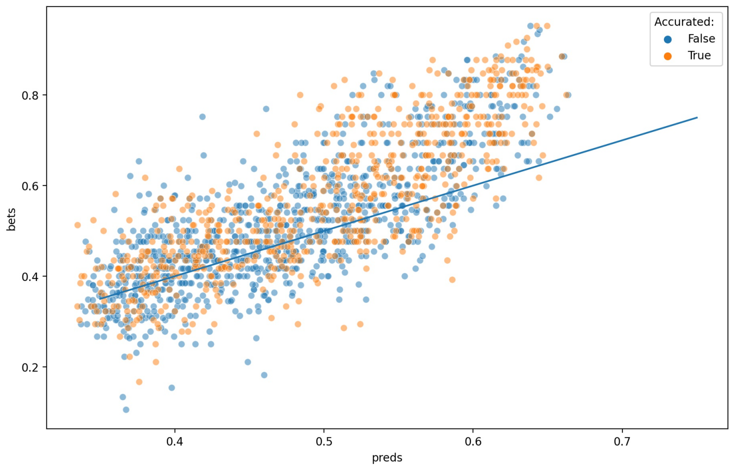
(a) Model probabilities.

(mean)	bet > pred.	pred. > bet
Accurate	0.64	0.36
Wrong	0.59	0.37

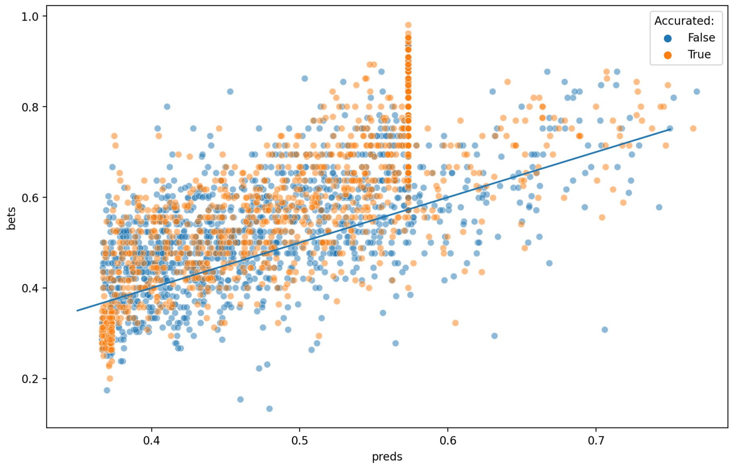
(b) Betting probabilities.

Table 7.19: Mean probability of bets and model predictions.

The Table 7.18 shows the percentages of matches that are accurate and incorrect in relation to the difference between the odds and our predicted results. The predictions of



(a) Best vs predictions of the model with highest accuracy (model with 54.5% accuracy).



(b) Best vs predictions of the model with highest f1-score in draws (model with $f = 10$).

Figure 7.28: Comparison of bets and predictions. Coloured in orange the well predicted matches and in blue the wrong predicted ones.

the model with a larger f1-score of draws are shown. We can see that in the model with the best accuracy in the draws we predict 16% of the matches with a probability higher than the betting odds, 5% more than the model with the best accuracy. For these 16% of the matches, our model predicts the results better and also gets the outcome right. Moreover, the average probability that our model gives for the correct result in these matches is only 41.7% compared to the average of 36% given by the bookmakers (Figure 7.19). In other words, we also beat the bookmakers' predictions in tight matches, where they are not convinced of what is going to happen and set a conservative odds.

However, we have already mentioned several times throughout this paper that using the model output as probabilities is not entirely accurate and that they contain a margin of error. We show in Figure 7.20 what the accuracy would be if we consider the probabilities within a 95% Confidence Interval.

The Table 7.20 shows the range of accuracy of each model within each confidence interval. We see that the calibrated ANN, although it can reach 65% accuracy at a CI of 90%, is also the most sensitive model and can also fall to a minimum of 38.3% accuracy. This is because this model, in order to calibrate the probabilities (balancing the predic-

CI (%) / Model	acc.	acc. 95%	acc. 90%	acc. 99%
Baseline (odds)	0.545	0.504 - 0.552	0.473 - 0.590	0.523 - 0.545
ANN (event based)	0.5	0.402 - 0.541	0.390 - 0.570	0.490 - 0.510
ANN (hist short term)	0.52	0.410 - 0.551	0.421 - 0.580	0.516 - 0.525
ANN (uncalibrated)	0.545	0.500 - 0.590	0.460 - 0.625	0.534 - 0.552
ANN (calibrated)	0.521	0.440 - 0.594	0.383 - 0.653	0.510 - 0.540
LightGBM	0.532	0.500 - 0.552	0.467 - 0.584	0.526 - 0.536

Table 7.20: Best possible accuracy within different Confidence Intervals.

tions so that there are draws), puts them together and makes less aggressive predictions. This shows that this model, despite being the best at predicting draws, is also the least reliable because it is often on the edge of being right or wrong. We see, then, that similar to the baseline is the accuracy of the uncalibrated ANN, which although slightly worse at the lower level of the CI, it is 4% better than the baseline in the upper range. This suggests that the uncalibrated ANN is quite robust in its predictions and also comes very close quite often to getting the result right. The LightGBM model, although lower in accuracy, also proves to be a fairly robust model.

With this final comparative analysis of the different models with the baseline we conclude the experiments. We have seen during this part how our models can reach the same accuracy and error as the bookmakers' odds, how our models can in some cases predict in a more robust and convincing way (with a higher probability) the outcome of a match, and how we can calibrate our predictions and see how reliable they are. We have also seen in which type of matches our models do better (leagues, months, level of the match...) and in which cases you can take advantage of these predictions and in which ones there is still margin for improvement.

CHAPTER 8
Conclusions

In this chapter we will discuss the final conclusions reached after the work has been completed. In contrast to the conclusions of the results, here we will focus on more general conclusions.

Getting back to the objectives set at the beginning of the work, we can say that we have managed to create a predictive model of football results based on Machine Learning. To say this, we rely on the comparison made between our predictive models and the benchmark used, the bookmakers' odds. We have obtained an overall accuracy equal to that obtained by bookmakers. Bookmakers have been doing this for years and have very well prepared working teams for this, as well as using more complete data than we had for this work. In addition, our model makes better predictions in the most difficult matches to predict, the draws. That is why we believe that the results obtained are satisfactory if we compare them with this Baseline.

In addition, we have tested two different classification Machine Learning algorithms and compared them with each other. We have seen how neural networks are algorithms that need precise configurations to obtain good results, as well as a proper feature engineering and an iterative process of model error analysis to find the way to calibrate the prediction distributions, especially in a problem like this one where the target classes are not evenly distributed. On the other hand, we have performed experiments with the Gradient boosting algorithm LightGBM, which has surprised us as it is possible to achieve really good results simply by optimising hyperparameters.

On the other hand, it is complicated to compare our work with the related works presented, since, as we have already mentioned, the fact of using totally different data or approaches (binary classification) complicates the comparison. However, we consider it appropriate to highlight, for example, the similar success obtained with the work carried out by Tax and Joustra in which they predict the outcome (draw, home or away win) from thirteen seasons of the Dutch league also applying MLP and dimensionality reduction methods [Tax and Joustra, 2015]. Other works related to outcome prediction are more difficult to compare with ours, as they use a completely different approach or data, both in the data used and in the type of matches to be predicted.

Our ANN-based model shows how data based on general statistics from previous matches can be used to make bookmaker-level predictions and improve the accuracy of the most difficult matches to predict, draws, which have so far been one of the biggest problems in result prediction. In our results we demonstrate something that previous works have not paid much attention to, the importance of not only using recent results, but also old matches to give a complete match context to the model. In addition, we have proposed solutions, not presented so far in works on score prediction, to combat the problem of the unbalanced dataset that causes problems when predicting draws. With these solutions we have been able to predict, in many cases, close matches with greater accuracy and confidence.

Related to this, as a further objective, we have been able to draw from the exploratory and outcome analyses conclusions about certain hidden or unintuitive relationships that exist between certain statistics of previous matches and the outcome of a football match. These relationships have helped us to propose solutions to enhance the predictive performance of our models. We have already mentioned that counter-intuitively results or statistics, such as corners, from 1 or 5 years ago have a stronger relationship with the result of a current match than the result of the previous week's match. We have also seen that previous results between two teams do not usually have a decisive influence on the outcome of a match and therefore our models do not give them much importance.

In addition to the conclusions about the results obtained, we will discuss in the following section some findings about the relationship of our work with the courses taken during the degree.

8.1 Relationship between the work carried out and the studies undertaken

This work has required the application of many different fields that I have studied during my degree in Computer Engineering. In this section we will explain what knowledge I have acquired during the degree I have applied in this work as well as the new knowledge I have acquired during its realization.

On the one hand, I have been able to apply the knowledge acquired in software engineering during the implementation of the whole infrastructure of the experiments. Thanks to this I have been able to implement different types of neural networks by re-using code through class inheritance, which is a property of Object Oriented Programming. Moreover, thanks to the acquired knowledge related to Data Structures and Databases I have been able to implement the code of the experiments, both the execution of the experiment and the writing of logs and results, in a efficient way in terms of computational cost.

Furthermore, related to the Data Science knowledge, it has been very useful to have been taught during the degree Data Science concepts such as overtraining, the bias of a model, the optimisation of a Machine Learning model or statistical concepts such as the different continuous and discrete distributions or the statistical analysis of ANOVA. It has also been very useful to learn machine learning algorithms such as Multi-Layer Perceptron or K-Means..

In addition, during my academic exchange I had the opportunity to acquire a lot of valuable knowledge that I have been able to apply in this work. First of all, during that time I learned the theoretical and mathematical foundations behind neural networks and other Data Science related algorithms such as PCA. In addition, I learned to use Data Science tools (Pandas, Numpy, Matplotlib, Scikit-Learn, etc.) in a practical way that has helped me to speed up data processing and visualisation tasks. This work has helped me to reinforce and deepen the knowledge I acquired both theoretically and practically in Data Science and Software Engineering.

Apart from what I learnt during my degree, during this work I have developed new technical, methodological and work planning skills. On the technical side, I have learned to use some frameworks such as Optuna from scratch. Meanwhile, in the more methodological part I have learned everything related to conducting a Data Science research work, from researching related work and the State-Of-The-Art to defining a methodology for experiments and analysis. I have also learnt to deal with situations where I was stuck and did not know how to proceed.

As in any learning process, problems and difficulties have arisen and I have had to face them and look for solutions. These aspects will be discussed in the next section.

8.2 Difficulties and limitations during my thesis

The difficulties and limitations that have arisen during the work have been basically two: difficulty in finding a suitable database to carry out the work and difficulty in automating the experiments.

Regarding the first point, we must be aware of the important limitation to obtain good results due to the fact of having such limited datasets, whether due to the amount of data (Event Based dataset) or due to the information contained in it (Historical dataset). This is why many hours have been invested not only in implementing the transformation of the original databases into trainable data, but also in thinking and designing the characteristics of the dataset based on the original data.

The other difficulty has to do with the development of the experiment infrastructure and its use. Moreover, the design and implementation of such infrastructure has not been trivial as it had a high level of complexity and it was not easy to carry out tests to check the correct functioning of the code. In addition, the design of the infrastructure had to be reconsidered several times, either because of important changes in the functionalities because the new implemented functionalities required it, e.g. new training logs. It has also been necessary to make major changes in the implementation due to the limitation of the virtual machine's memory, which on one occasion led to changing the way in which the logs were collected and the outputs of each model were saved.

Additionally, the machine on which the experiments were performed had problems during the experiments, being unavailable for certain periods of time, which affected the continuous development of the experiments. However, these problems were finally solved and the experiments could continue as normal.

Finally, as it is common when you are introduced to a new subject, difficulties arose related to theoretical concepts that were new to us. Especially in the application of techniques that we had not used so far, such as the statistical analysis of ANOVA for feature selection or the application of Batch Normalization to a neural network.

Since this is a work to be carried out within a limited time span, some solutions were proposed in the course of the project, but could not be implemented. In the following section we will discuss these aspects.

8.3 Future work

The lack of time meant that it was not possible to develop many of the ideas and new research approaches that had been considered during the course of this work.

One of these new routes of research was to change the type of problem set to predict outcomes from a classification problem to a regression problem. The idea was to approximate the exact outcome of the match with a predictive model (these could be the same as the ones used in the paper, ANN and LightGBM). In fact, some exploratory tests have been carried out but they have not been conclusive and it has been decided not to include them in the report. However, the results we were able to obtain from these tests suggested that this approach is quite promising.

We also considered the possibility of considering the data and the problem as a time series prediction problem, in which the statistics of each match would be data points of a time series of statistics of each team and when predicting the outcome of a match, the time series of both teams would be taken into account. This idea has not been developed further due to the limited time we had and the complexity it has.

As next steps, we could develop a new software infrastructure that would allow the automation of predictions with current matches. Through web scrapping it could be possible to collect data from current matches and add them to our training dataset in order to predict upcoming matches.

In parallel to this, we would continue to improve the predictive model, adding more data (leagues, previous seasons, lower divisions, etc) and also rethinking the problem of the non-uniform distribution of results in the dataset in order to make better quality predictions.

Bibliography

[Akiba et al., 2019] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

[Bialkowski et al., 2014] Bialkowski, A., Lucey, P., Carr, P., Yue, Y., and Matthews, I. (2014). Win at home and draw away: Automatic formation analysis highlighting the differences in home and away team behaviors. In *Proceedings of 8th annual MIT sloan sports analytics conference*, pages 1–7. Citeseer.

[Bunker and Thabtah, 2019] Bunker, R. P. and Thabtah, F. (2019). A machine learning framework for sport result prediction. *Applied computing and informatics*, 15(1):27–33.

[Bush et al., 2015] Bush, M., Barnes, C., Archer, D. T., Hogg, B., and Bradley, P. S. (2015). Evolution of match performance parameters for various playing positions in the english premier league. *Human movement science*, 39:1–11.

[Fernandez-Navarro et al., 2016] Fernandez-Navarro, J., Fradua, L., Zubillaga, A., Ford, P. R., and McRobert, A. P. (2016). Attacking and defensive styles of play in soccer: analysis of spanish and english elite teams. *Journal of sports sciences*, 34(24):2195–2204.

[Gama et al., 2014] Gama, J., Passos, P., Davids, K., Relvas, H., Ribeiro, J., Vaz, V., and Dias, G. (2014). Network analysis and intra-team activity in attacking phases of professional football. *International Journal of Performance Analysis in Sport*, 14(3):692–708.

[Goddard, 2006] Goddard, J. (2006). Who wins the football? *Significance*, 3(1):16–19.

[Godin et al., 2014] Godin, F., Zuallaert, J., Vandersmissen, B., De Neve, W., and Van de Walle, R. (2014). Beating the bookmakers: leveraging statistics and twitter microposts for predicting soccer results. In *KDD Workshop on large-scale sports analytics*, pages 2–14. ACM New York, NY, USA.

[Gonçalves et al., 2014] Gonçalves, B. V., Figueira, B. E., Maçãs, V., and Sampaio, J. (2014). Effect of player position on movement behaviour, physical and physiological performances during an 11-a-side football game. *Journal of sports sciences*, 32(2):191–199.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Guan and Wang, 2022] Guan, S. and Wang, X. (2022). Optimization analysis of football match prediction model based on neural network. *Neural Computing and Applications*, 34(4):2525–2541.

[Guo et al., 2017] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks.

[Heuer and Rubner, 2009] Heuer, A. and Rubner, O. (2009). Fitness, chance, and myths: an objective view on soccer results. *The European Physical Journal B*, 67(3):445–458.

[Jain et al., 2021] Jain, S., Tiwari, E., and Sardar, P. (2021). Soccer result prediction using deep learning and neural networks. In *Intelligent Data Communication Technologies and Internet of Things*, pages 697–707. Springer.

[Koopman and Lit, 2015] Koopman, S. J. and Lit, R. (2015). A dynamic bivariate poisson model for analysing and forecasting match results in the english premier league. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 178(1):167–186.

[Neal et al., 2018] Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I. (2018). A modern take on the bias-variance tradeoff in neural networks.

[Nyquist and Pettersson, 2017] Nyquist, R. and Pettersson, D. (2017). Football match prediction using deep learning. Master's thesis.

[Pappalardo and Cintia, 2018] Pappalardo, L. and Cintia, P. (2018). Quantifying the relation between performance and success in soccer. *Advances in Complex Systems*, 21(03n04):1750014.

[Pappalardo et al., 2019] Pappalardo, L., Cintia, P., Rossi, A., Massucco, E., Ferragina, P., Pedreschi, D., and Giannotti, F. (2019). A public data set of spatio-temporal match events in soccer competitions. *Scientific data*, 6(1):1–15.

[Rahman et al., 2020] Rahman, M. et al. (2020). A deep learning framework for football match prediction. *SN Applied Sciences*, 2(2):1–12.

[Reep and Benjamin, 1968] Reep, C. and Benjamin, B. (1968). Skill and chance in association football. *Journal of the Royal Statistical Society. Series A (General)*, 131(4):581–585.

[Rein and Memmert, 2016] Rein, R. and Memmert, D. (2016). Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. *SpringerPlus*, 5(1):1–13.

[Rudrapal et al., 2020] Rudrapal, D., Boro, S., Srivastava, J., and Singh, S. (2020). A deep learning approach to predict football match result. In *Computational Intelligence in Data Mining*, pages 93–99. Springer.

[Shlens, 2014] Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

[Silva et al., 2014] Silva, P., Travassos, B., Vilar, L., Aguiar, P., Davids, K., Araújo, D., and Garganta, J. (2014). Numerical relations and skill level constrain co-adaptive behaviors of agents in sports teams. *PloS one*, 9(9):e107112.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[Tax and Joustra, 2015] Tax, N. and Joustra, Y. (2015). Predicting the dutch football competition using public data: A machine learning approach. *Transactions on knowledge and data engineering*, 10(10):1–13.

[Ulmer et al., 2013] Ulmer, B., Fernandez, M., and Peterson, M. (2013). Predicting soccer match results in the english premier league. *Doctoral dissertation, Doctoral dissertation, Ph. D. dissertation, Stanford*.

[Wang et al., 2015] Wang, Q., Zhu, H., Hu, W., Shen, Z., and Yao, Y. (2015). Discerning tactical patterns for professional soccer teams: an enhanced topic model with applications. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2197–2206.

[Yu and Wang, 2015] Yu, Y. and Wang, X. (2015). World cup 2014 in the twitter world: A big data analysis of sentiments in us sports fans' tweets. *Computers in Human Behavior*, 48:392–400.

[Yucesoy and Barabási, 2016] Yucesoy, B. and Barabási, A.-L. (2016). Untangling performance from success. *EPJ Data Science*, 5(1):1–10.

APPENDIX A
Terminology

ANN	Artificial Neural Network
API	Application Programming Interface
BP	Back-propagation
CE	Cross-Entropy
CV	Cross-Validation
EM	Expectation Maximization
GB	Gradient Boosting
LSTM	Long-Short Term Memory
MLP	Multi-layer Perceptron
NLP	Natural Language Processing
PCA	Principal Component Analysis
PCF	Probability Cumulative Function
PDF	Probability Density Function
RBF-SVM	Radial Basis Function - Support Vector Machine
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descend

SNA	Social Network Analysis
SVD	Singular Value Decomposition
SVM	Support Vector Machine

APPENDIX B

Sustainable development goals

Relevance of the work to the Sustainable Development Goals (SDGs).

Sustainable Development Goals	**High**	**Medium**	**Low**	**Not applicable**
ODS 1. **End of poverty.**				X
ODS 2. **Zero hunger.**				X
ODS 3. **Health and welfare.**		X		
ODS 4. **Quality education.**		X		
ODS 5. **Gender equality.**		X		
ODS 6. **Clean water and sanitation.**				X
ODS 7. **Affordable and clean energy.**				X
ODS 8. **Decent work and economic growth.**	X			
ODS 9. **Industry, innovation and infrastructure.**	X			
ODS 10. **Reducing inequality.**	X			
ODS 11. **Sustainable cities and communities.**			X	
ODS 12. **Responsible production and consumption.**			X	
ODS 13. **Climate action.**				X
ODS 14. **Underwater life.**				X
ODS 15. **Life of terrestrial ecosystems.**				X
ODS 16. **Peace, justice and solid institutions.**				X
ODS 17. **Partnerships to achieve objectives.**			X	

Consideration of the relationship of the TFG/TFM with the SDGs and with the most relevant SDG(s).

Sport analytics can play a significant role in supporting the United Nations' Sustainable Development Goals (SDGs). Through the collection, analysis, and interpretation of data, sport analytics can help identify opportunities to promote social and environmental sustainability within the world of sport. For instance, data analysis can be used to track and improve the sustainability of sporting events, from reducing waste to increasing the use of renewable energy. Additionally, sport analytics can be used to promote inclusivity and gender equality in sports by analyzing participation rates and identifying areas where marginalized groups may be underrepresented.

The relationship between football match prediction using machine learning methods and the SDGs is not a direct one. However, there are potential indirect relationships that could exist between the two. By analyzing data such as player statistics, team performance history, and other relevant factors, football forecasting can help promote responsible gambling practices and prevent match-fixing, both of which can have negative social and economic consequences. Additionally, forecasting can help inform decisions related to sustainable development by analyzing the environmental impact of football matches and identifying opportunities to reduce waste, energy use, and carbon emissions associated with large-scale sporting events.

In terms of quality education, machine learning methods used in football match prediction could help promote the use of data-driven decision-making in the sports industry, leading to the development of more sophisticated data analysis techniques and greater access to training and education opportunities in data science.

The use of machine learning in football match prediction has the potential to drive innovation in the sports industry by creating new opportunities for data analysis, modeling, and prediction. This, in turn, could lead to the development of new technologies and tools that can improve the quality of infrastructure in the industry, such as better tracking systems or more sophisticated broadcasting techniques.

Finally, the use of machine learning in football match prediction could lead to more responsible consumption patterns. For instance, by accurately predicting match outcomes, bettors may be less likely to engage in risky or impulsive betting practices, leading to more responsible gambling behavior. Additionally, by analyzing data related to the environmental impact of football matches, machine learning techniques could be used to identify opportunities for reducing waste, energy consumption, and carbon emissions associated with large-scale sporting events, promoting more sustainable consumption patterns.

On the other hand, there are potential concerns related to the responsible production of machine learning algorithms used in football match prediction. For example, the production of these algorithms may require significant resources and energy, and their use may lead to privacy concerns or biases in decision-making.

Overall, the relationship between football match prediction using machine learning methods and the SDGs is complex and indirect. While accurate match predictions could lead to economic growth, greater access to education, and more equitable sports industry practices, there are no guarantees that this will be the case. However, the potential for these outcomes highlights the importance of continuing to explore the use of data science in the sports industry.