



HAL
open science

Statistical modeling of event probabilities subject on a sports bet : Theory and applications to soccer, tennis and basketball

Paul Steffen

► **To cite this version:**

Paul Steffen. Statistical modeling of event probabilities subject on a sports bet : Theory and applications to soccer, tennis and basketball. Statistics [math.ST]. Université de Bordeaux, 2022. English. NNT : 2022BORD0210 . tel-03891393

HAL Id: tel-03891393

<https://theses.hal.science/tel-03891393>

Submitted on 9 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse présentée
pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE
LABORATOIRE DE L'INTÉGRATION DU MATÉRIAU AU SYSTÈME

Spécialité: MATHÉMATIQUES APPLIQUÉES

Par **Paul STEFFEN**

Statistical modeling of event probabilities subject on a
sports bet: Theory and applications to soccer, tennis and
basketball

Sous la direction de : **Léo GERVILLE-REACHE**

Soutenue le 1er juillet 2022

Membres du jury :

M. Julien MORLIER	Professeur des universités, Université de Bordeaux	Président du jury
M. Jean-Michel MARIN	Professeur des universités, Université de Montpellier	Rapporteur
M. Christophe LEY	Professeur associé, Université du Luxembourg	Rapporteur
M. Léo GERVILLE-REACHE	Maître de conférences, Université de Bordeaux	Directeur de thèse
Mme. Marie CHAVENT	Professeure des universités, Université de Bordeaux	Examineur
Mme. Brigitte GELEIN	Docteure, ENSAI	Examineur
M. Nicolas BISOFFI	Manager Data Science, Betcliv Group	Invité

A ma mère et ma grand-mère, pour l'éducation qu'elles m'ont procuré, ainsi que leur soutien et leur accompagnement.

A Pauline, ma compagne, pour son soutien quotidien et sans qui je n'aurais pas pu écrire cette thèse.

A ma famille, ma belle-famille et mes amis, leurs attentions et encouragements m'ont accompagné tout au long de ces années.

Acknowledgments

Je remercie chaleureusement toutes les personnes qui m'ont aidé pendant l'élaboration de ma thèse et notamment mon directeur, Léo Gerville-Réache, et mon manager au sein de Betcllic, Nicolas Bisoffi, pour leur intérêt et leur soutien, leur grande disponibilité et leurs nombreux conseils durant la réalisation de ma thèse.

Ce travail n'aurait pu être mené à bien sans la disponibilité et l'accueil chaleureux que m'ont témoignés Hassan Fathi et l'ensemble des personnes de l'équipe Data de Betcllic. Toujours très intéressés par mon sujet de recherche, ils m'ont accompagné durant la réalisation de ce travail.

Je remercie l'ensemble des membres du jury ayant accepté d'examiner et d'évaluer mon travail.

Je remercie Isabelle Bisoffi, pour l'aide qu'elle m'a apporté afin de proposer ce travail en anglais.

Ce travail n'aurait pas été possible sans le soutien de l'Université de Bordeaux et du Laboratoire IMS, qui m'ont permis, grâce à une allocation de recherches, de me consacrer sereinement à l'élaboration de ma thèse.

Abstract

Establishing the odds for a set of sports bets requires, amongst other things, establishing the probabilities for a set of characteristic events. If we take the example of a soccer game, the score at half time is an event. The final score is also an event (dependent on the score at half time). We can also bet on who scores the goal, on which team scores first... As the preliminary studies on the subject of sports predictions and analyses have shown ever since the mid-20th century, the more accurate, important and relevant the data fed into the model are, the more reliable the estimate of the probability for the occurrence of an event will be. With the recent development in the volume of data used, their accessibility as well as the technical means that allow for the processing of the data, previous sports event data that were up to now seldom used, have been gathered and collected from 6 different websites that specialize in the publication of data and information of sports results and statistics. Thus, a structured database concerning sporting events between the years 1991 and 2018 was built. Once the data are gathered, they are then cleaned, verified and formatted in order to be turned into a usable set of reliable data. Since the overall data come from different sources, it was necessary to join all the pieces of data together by using common indexes that were built on the syntactic proximity of the observations. Therefore the expected goals, the box-scores or the Elo points, which are all specialized metrics in the field of study, allow for a considerable improvement in the performance of the model. Faced with the problem of modeling the probability of a sporting event, supervised classification algorithms capable of predicting a probability distribution over a set of classes have been used, rather than displaying only the most probable class, for a given observation. Thus, one can have a certain level of confidence in the occurrence of all sporting events, and not be interested only in the most probable event:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} Pr(\mathbf{Y} = \mathbf{y}|\mathbf{X}) \quad \forall \mathbf{y} \in \mathbf{Y}$$

Furthermore, it is always this probability distribution that will be used to compare the models with each other with the help of appropriate evaluation metrics : where p_{ij} is the probability produced for the observation i of being in the class j , and y_{ij} is the variable

indicating whether or not the event has occurred:

$$Loss = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} f(y_{ij}, p_{ij})$$

where p_{ij} is the probability for observation i to be in class j and y_{ij} is the variable indicating the realization or not of the event. In order to minimize this loss function, representing the performance of the model, the features were selected and the hyper-parameters of the model were adjusted, following a division of the data into several samples, in order to simulate a use of the model in which the probabilities could be proposed before the beginning of each encounter. Following a comparison with other bookmakers, the proven quality of the results makes it possible for Betclie to suggest relevant odds pertaining to the outcome of sports encounters in tennis, basketball and soccer. The declination on finer events, such as the exact score, is also possible.

Keywords: Statistic – Probability – Data Science – Machine Learning – Sports betting – Soccer – Tennis - Basketball

Contents

Acknowledgments	ii
Abstract	iii
List of Tables	ix
List of Figures	xiii
1 Introduction	1
2 Data Collection	5
2.1 Data sources	7
2.2 Database overview	10
2.3 Web scraping	14
2.4 Schema integration	21
2.5 Conclusion	22
3 Some models on sport data	23
3.1 Match scores	24
3.2 Match outcomes	40

3.3	Conclusion	45
4	Background on some learning strategies	46
4.1	Probabilistic classification algorithms	46
4.1.1	Logistic regression	47
4.1.2	Naive Bayesian	50
4.1.3	K-Nearest Neighbors (KNN)	51
4.1.4	Support Vector Machines (SVM)	51
4.1.5	Decision Tree	54
4.1.6	Ensemble methods	56
4.1.7	Artificial Neural Networks (ANN)	60
4.2	Evaluation metrics	63
4.2.1	Accuracy	63
4.2.2	Brier score	64
4.2.3	Rank Probability Score (RPS)	64
4.2.4	Area Under Curve (AUC) of Receiver Operating Characteristic (ROC) curve	65
4.2.5	Logarithmic loss (Cross Entropy)	66
4.3	Hyperparameter tuning methods	67
4.3.1	Cross-Validation	68
4.3.2	Navigating the Hyperparameter Space	69
4.4	Conclusion	72
5	Feature Engineering & Selection	73

5.1	Feature Transformers	73
5.1.1	Numerical transformations	73
5.1.2	Categorical transformations	75
5.2	Feature Construction	76
5.2.1	Common features	78
5.2.2	Tennis specific features	84
5.2.3	Basketball specific features	90
5.2.4	Soccer specific features	94
5.3	Feature Selection	107
5.4	Conclusion	113
6	Experiments & Results	114
6.1	Benchmark	114
6.2	Data split	120
6.3	Hyperparameter tuning	121
6.4	Model results	132
6.5	Conclusion	144
7	Conclusion	147
	Bibliography	150
A	Appendix	162

List of Tables

5.1	One Hot Encoding	75
5.2	Ordinal Encoding	76
5.3	ATP point distribution since 2009	86
5.4	Non match-related ATP player-level features	87
5.5	Match-related ATP player-level features	88
5.6	Aggregate horizons for match-related features	89
5.7	Formulas of basketball team-level features	91
5.8	Formulas of basketball player-level features - 1/2	92
5.8	Formulas of basketball player-level features - 2/2	93
5.9	Soccer tactic encoding	95
5.10	Soccer xG model results	101
6.1	Bookmaker metrics on NBA	116
6.2	Bookmaker metrics on ATP	117
6.3	Bookmaker metrics on 5 main European soccer leagues	119
6.4	Partial Logistic Regression grid-search results on soccer data	123
6.5	Hyperparameter search space used during Bayesian optimization of XGBoost	129

6.6	Model metrics on ATP	135
6.7	Model metrics on NBA	139
6.8	Model metrics on 5 main European soccer leagues	143
6.9	2018 Champion's League probabilities of final scores	145
A.1	Basketball abbreviation table	163
A.2	Tennis X_i correlation	165
A.3	Basketball X_i correlation	167
A.4	Soccer X_i correlation	169

List of Figures

2.1	Tennis database diagram	11
2.2	Soccer database diagram	12
2.3	Basketball database diagram	13
2.4	Inter-Process Communication database diagram	13
2.5	<i>SoFIFA</i> player list web page	15
2.6	<i>Whoscored</i> match list web page	16
2.7	<i>Transfermarkt</i> club list web page	17
2.8	<i>2KMTC</i> player list web page	18
2.9	<i>Basketball-Reference</i> match list web page	19
2.10	<i>ATP</i> match list web page	20

4.1	Sigmoid function	48
4.2	Impact of the number of neighbors in a binary classification example using a K-Nearest Neighbors with two features	51
4.3	Binary classification example using Support Vector Machines with soft margin and two features	53
4.4	Multinomial classification example using a Decision Tree	56
4.5	Bagging process details	57
4.6	Boosting process details	59
4.7	Feedforward Neural Networks structure details	62
4.8	Receiver Operating Characteristic curve	66
4.9	Grid & Random search comparison from <i>Introduction to Deep Learning</i> (Varma and Das 2018)	70
4.10	Tree-Structured Parzen Estimator details from NeuPy (Dr. Thorben Jensen)	72
5.1	Win rate using observed and predicted Elo on tennis ATP from 2015 to 2018	83
5.2	xG ROC and calibration curves	102
5.3	xG heatmap	102
5.4	16×22 xT interpolated heatmap	105
5.5	VAEP of victory goal of MHSC-OGCN during the 2017/2018 season .	107
5.6	Filter method design	108
5.7	Wrapper method design	109
5.8	Boruta process	111

5.9	Probability Mass Function of a Binomial distribution with $n = 100$, $p = 0.5$ and decision areas with $\alpha = 0.005$	112
5.10	Embedded method design	113
6.1	Bookmakers' commercial margins	115
6.2	Evaluation of probabilities induced by <i>Bet365</i> tennis odds	118
6.3	Evaluation of probabilities induced by <i>Bet365</i> soccer odds	119
6.4	Temporal 3-fold cross-validation split	121
6.5	KNN performances depending on hyperparameters during Grid search on tennis data	124
6.6	SVM performances depending on hyperparameters during Grid search on soccer data	125
6.7	Decision Tree TPE performance history plot using basketball data . .	126
6.8	Random Forest optimization history using TPE on soccer data	127
6.9	Extra Trees performances depending on hyperparameters during Bayesian search on soccer data	128
6.10	Catboost performances depending on hyperparameters during Bayesian search on tennis data	130
6.11	Neural Networks optimization history using TPE on tennis data . . .	131
6.12	SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature, from "A Unified Approach to Interpreting Model Predictions"	132
6.13	Feature impact in logistic regression model applied to ATP data based on SHAP value	133

6.14	Explanation of the prediction of the 2018 Roland Garros final between R. Nadal and D. Thiem using logistic regression	134
6.15	Performance comparison between probabilities forecast and the estimation of the probabilities induced by the bookmakers' odds on tennis ATP	136
6.16	2017 NBA regular period	137
6.17	2017 NBA playoff period	137
6.18	Feature impact in Catboost model applied to NBA data based on SHAP value	137
6.19	2017 NBA final opposing Golden State to Cleveland	139
6.20	2018 NBA final opposing Golden State to Toronto	139
6.21	Explanation of the prediction of the NBA finals using Catboost . . .	139
6.22	Performance comparison between probabilities forecast and the estimation of the probabilities induced by the bookmakers' odds on NBA	140
6.23	Feature impact in Random Forest model applied to soccer data based on SHAP value	141
6.24	2017 Champion's League final opposing Real Madrid to Liverpool . .	142
6.25	2018 Champion's league final opposing Tottenham to Liverpool . . .	142
6.26	Explanation of the prediction of the Champion's League finals using Random Forest	142

6.27	Performance comparison between probabilities forecast and the estimation of the probabilities induced by the bookmakers' odds on the 5 main European leagues	144
A.1	Tennis features selected using <i>Boruta</i> algorithm	164
A.2	Basketball features selected using <i>Boruta</i> algorithm	166
A.3	Soccer features selected using <i>Boruta</i> algorithm	168
A.4	Code example: Basketball Elo ranking	170
A.5	Code example: Basketball teams metrics computation	171
A.6	Code example: Basketball player metrics computation 1/2	172
A.7	Code example: Basketball player metrics computation 2/2	173

1. Introduction

Now an inevitable and ever present aspect of the worlds of sports and games, betting is a practice that dates back to Ancient Greece. For a time, this practice took place in the form of Parimutuel betting, whereby all the stakes bet by the bettors were pooled into a common pool before being redistributed to the winners in proportion to their stake. This “mutual betting” system invented by Joseph Oller first appeared in France in 1867. That said, the most widespread form of betting is the fixed odds betting system.

In this context, betting on odds takes the form of a contract in which the bookmaker sets the odds, which can vary as much as the bookmaker wants them to in accordance with variations in the elements taken into account for their establishment. When the bettor confirms his bet, the odds are then definitely fixed. If the event that was bet on occurs, these fixed odds will determine the multiplier of the sum that was bet. Thus, the main task of the bookmaker is to calculate the odds of the bets he proposes. Based on statistical data, the history of encounters and all the relevant information available, these odds are mostly a transcription of the probability for the event to

occur. However, the odds also consider the bookmaker's margin and in a more subtle way, in order to control the financial risk, considering as well the spread of the bets.

To what extent can the use of data science help a bookmaker like Betclac, partner of this CIFRE thesis, in the elaboration of its sports betting odds ?

With the soaring in the volume of data, mainly due to a growing interest in this "new black gold", the ever expanding digital accessibility produced by a generalization of the use of sensors (such as the use of cameras or connected devices) and the ongoing improvement of our computational capacities, all the ingredients are gathered to offer ever more.

Furthermore, a better utilization of the large volume of data available is made possible by the development of the field of sports analysis. This has been made known to the public in large part thanks to the movie "Moneyball" that relates how Paul DePodesta used sabermetrics to build a more competitive team for the Oakland Athletics. Thus, when used with a predictive classification model, whose role it is to classify observations into previously identified groups, a probability of occurrence for each event can be associated to the observations characterizing it, making it possible to obtain odds that are more precise than ever.

In France, as in many other countries around the world, the "king" of sports is soccer. Alongside tennis and basketball, these sports are very popular with bookmakers, whose users like to make predictions on who will win the upcoming championship or tournament. With hundreds of possible bets per match (such as the number of goals, sets or points scored during the match, or by team, or event at half-time)

punters can now bet on almost anything that could happen during a match. However, determining the outcome of the match is still the most important bet for bettors. Thus, the study of classifiers, binary for tennis and basketball and multi-class for soccer (where the match might end in a tie, with no winner or loser), is the main tool used to construct probabilities for the establishment of odds.

This section presents the outline of the rest of the report.

- **Chapter 2** presents the different data sources used. They are all accessible on sport websites and provide access to data of huge diversity and sufficient volume. Following the data extraction from these websites using web scraping, a relational database was created.
- **Chapter 3** presents some models for predicting the outcome of a match using soccer, basketball and tennis data. The outcome prediction of a sport match has interested sportsmen, bettors, but also statisticians for a long time. Since the middle of the 20th century, some have tried to find a solution to this problem. Whether the approach is direct, by determining who can be the winner of the match, or indirect, by predicting the match score, the approaches have been varied, and influenced by the trends and scientific and/or technical advances that we have known until today.
- **Chapter 4** presents the theoretical foundations of learning strategies used in this work. Inspired by previous works on the subject, the different classification algorithms and the methods used to optimize and evaluate the models are described.

-
- **Chapter 5** provides details on how that data was prepared and selected prior to the classification task. Fueled by the growing interest in “Sports Analytics”, the transformation of raw data into informative features greatly facilitates the task of the predictive model.
 - **Chapter 6** presents the application of the selected tools, with the aim of creating sport odds. According to an experiment very similar to real conditions of use of a bookmaker, a model is then selected, its results analyzed, before being compared to certainly one of the best current benchmark: the sport odds of a bookmaker.

2. Data Collection

As with any statistical modeling project, the key to success lies in the gathering and measuring of the data. The better this task is completed, the better and more accurate the solution to the problem will be.

At a time when the development of Big Data is exponentially rising, this is an obvious fact. Now that the Internet is an almost infinite source of data, a smart collection of these data could enrich analysis and modeling projects that could provide answers to all kinds of questions. Some key players in the sector have come to this realization and have decided to facilitate the access to this information. Google, for instance, launched “Google Dataset Search” in 2018. Its aim is to carry out structured “open data” searches. Kaggle and OpenML also provide free access to open datasets. Twitter even allows access to some of the data in its application via its APIs. More than 500 million tweets were sent every day in 2019, these could be used to reflect what is happening in the world and what people are talking about in real time.

The world of sports has also changed. Since the 1960’s, as the means and methods for collecting data have modernized, the answers that statisticians have been able to

contribute to the field of sports have evolved. Using mathematics and statistics to improve the objective knowledge of certain sports is increasingly widespread, making the quality and quantity of data required for this pursuit also increase.

While Moroney (1956) simply used the scores of 240 games that took place in England, Reep, Pollard, and Benjamin (1971) took into account the passes made between players during games, which were all recorded by hand over more than 2200 matches. Across the Atlantic, the sabermetrics¹ development in the 1960's followed by the APBRmetrics² development during the 1990's have also initiated the evolution and professionalization of practices in the field.

Today, companies have developed a whole economic activity around the topic. The display of real-time statistics during broadcasted sports events is now common in all forms of media, and right after a game, sport experts are supplied with information and statistics that enable them to discuss in depth the sporting performances of teams and players. At Opta, a British sports analytics company that provides data for 30 different sports, several professionals follow soccer matches and manually record every single event (around 2,000 per match). But some pieces of data, such as the distance traveled by a player, or the speed of the ball can only be calculated with the help of technology. In Germany, for example, the company ChyronHego has equipped a stadium with an electronic performance and optical tracking system. Thus, the quality of the data as well as their volume have greatly improved over the past 60 years, this in turn opens the door to new possibilities in the field of sports analysis

¹search for objective knowledge of baseball

²sabermetrics cousin concerning basketball

and outcome prediction.

Concerning the data used for this thesis project, this chapter focuses on presenting the sources, describing the database architecture and explaining how the data were retrieved.

2.1 Data sources

Some of the data needed to carry out this project were at first determined following a review of the state of the art for the subject at hand. In the end, some of the data also come from personal thoughts on the sports concerned, driven by the expertise of Betcltic. To obtain as much information as possible about the soccer, tennis and basketball matches concerned by this study, various websites were used to build up the database.

For soccer, 3 websites are chosen:

- *Whoscored* is a website that provides statistics on more than 500 international soccer leagues and competitions for free. Its database contains matches played ever since 1999 for some competitions making *WhoScored* able to offer detailed statistics on matches. For most of them, it is possible to know the teams playing, the time elapsed and the full-time score. Additional data have been available for the main championships since 2002, such as the line-up for each team, the substitutions that occurred and the main events of the game like goals, penalties, and cards. Finally, from the 2009/2010 season onwards, all the details of every match that took place in the main championships and competitions have been available. The stadium, the weather, the referee, or

the stadium attendance: all the events that took place during the match are observed. Thus, the geo-position, the minute of the match, the players involved and all the details characterizing passes, free kicks, tackles, shots or any other event of a match are known. All the information concerning the line-up of each team is available, such as its evolution during the match, the position of the players or who the captain is. Thus, many statistics about each player are calculable, such as his success percentages in one aspect of the game for instance.

- *Transfermarkt* is a website that provides the monetary value of soccer players. Founded by Matthias Seidel in 2000, this website has come to influence the transfer market in Europe. Often used as a benchmark by the media, clubs have even gone so far as to contact the teams behind this website to find out how the price of players was calculated. With major updates twice a year, during each transfer window, the value of the players is mainly determined manually by users, experts and administrators assisted by a calculation assistance system. For the 5 major European soccer leagues, the player value is collected for each player once per season. In addition, the ranking of each European league, based on the comparison of the performance of each club during European competitions, is available for the 2006/2007 season. This statistic makes it possible to compare the different European championships.
- *SoFIFA* is a website containing all the information available in the *EA Sports FIFA* video game franchise. With more than 20 years of expertise in the field, and thousands of scouts watching matches all over the world, *EA Sports*

determines scores in all aspects of a football player's skill set. Thus, beyond an overall rating and a potential rating, representing a player's overall level, the website provides ratings by position and for each aspect of the game (finishing skills, volleys etc... on the offensive side, or even marking, tackling, and sliding tackles on the defensive side). Globally, more than 30 aspects of a player's game, categorized into 7 areas, have been evaluated since the 2007/2008 season. The same is true for each team, which is given an overall score, as well as offensive, midfield, and defensive scores. Additionally, the line-up and tactics for a game are determined based on 20 aspects categorized into 5 areas.

For basketball, 2 websites were selected:

- *Basketball-Reference* is a website that belongs to the *Sports Reference* group, it offers statistics for sports such as baseball, basketball, American football, hockey and even soccer. Updated daily, this website is a reference in the field of sports statistics. It has been giving access to detailed data on all the NBA³ games, the evolution of the score and details of all the shots taken during the game (player, minute, score, geo-position), as well as most of the statistics generated in APBRmetry for each player since the 2000 season.
- *2KMTCentral* is a website that provides all the information available in the *EA Sports* video game franchise *NBA2K*. Like its soccer equivalent, this game has become a benchmark in the field, and the quality of the information it provides is difficult to dispute. Thus, a lot of information about NBA players has been specified as an overall rating, their size, their position, technical attributes

³National Basketball Association

as well as around forty ratings categorized into six game areas since the 2015 season.

Finally, all the information concerning tennis was collected on the same website.

- *ATP Tour* is the official website of the Association of Tennis Professionals. On this website, information on all the matches of more than 60 ATP tournaments around the world is accessible. All the scores, tournament details, week by week ATP rankings, player information such as backhand or backhand type and all the classical tennis statistics relating to the service or the return of players during a match ever since 1991 have been available. In addition, some statistics defining the previous performances of a player on a specific surface, or statistics about previous matches between two players are available.

2.2 Database overview

Accordingly, an architecture compatible with data analysis and modelling needed to be thought of. The choice was made to use a local database, this way each application necessary for the realization of this exercise could access the data as a user.

At the sight of the data described above, a relational database was initially used to store the data using tables, composed of rows and columns. Relationships between these tables are made by linking IDs. This database, presented using Figure 2.1, Figure 2.2 and Figure 2.3, is set up using *SQL Server 2016* version 13.0, and a copy of the data is made on *Amazon Simple Storage Service (Amazon S3)*, the object storage service of *Amazon Web Service (AWS)* to have easy access to data.

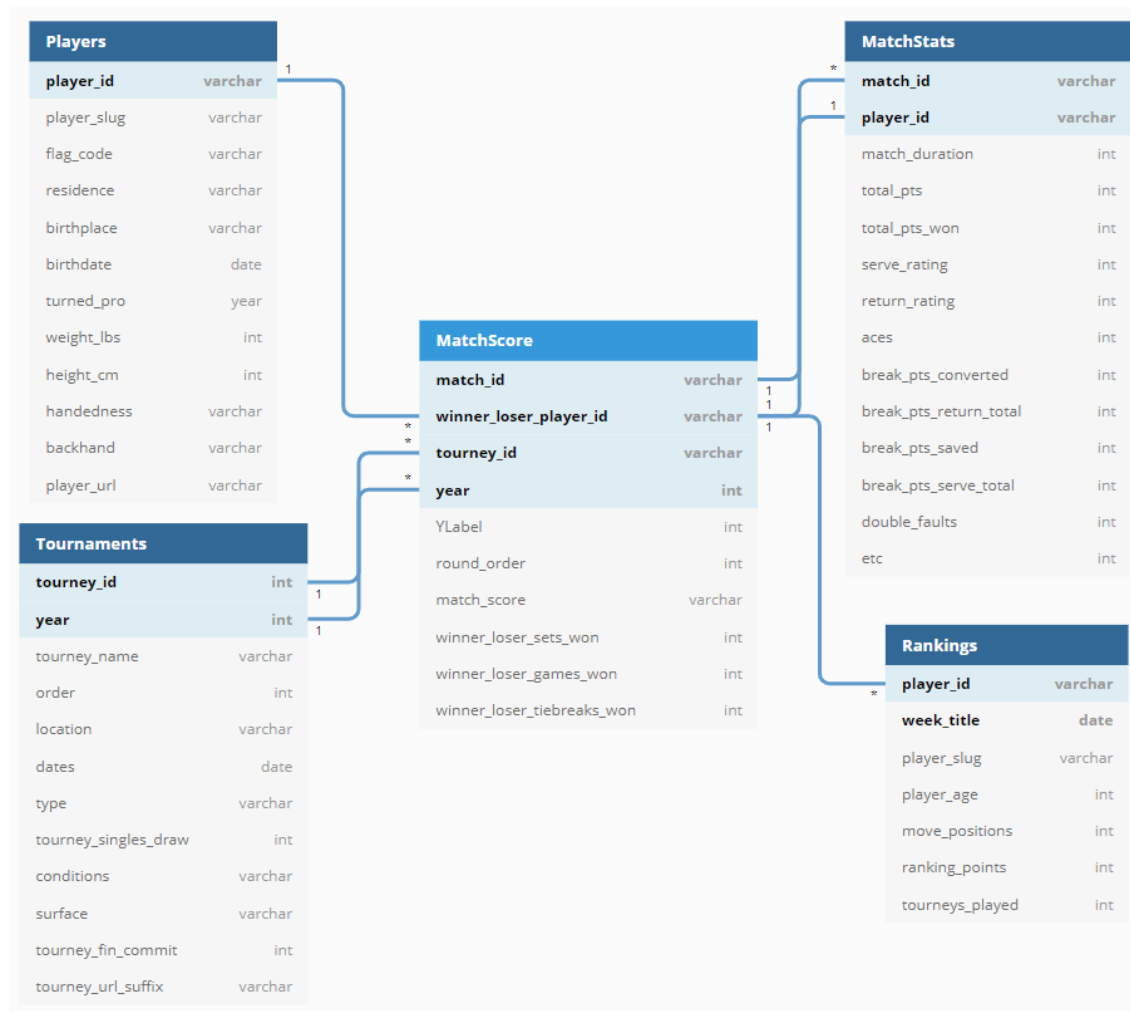


Figure 2.1: Tennis database diagram

Another database, presented using Figure 2.4, is used to transfer data, metadata and objects between the applications. To achieve this, a non-relational database (NoSQL⁴) was favored. Instead of the data being stored in tables, they are stored in a key-value format. A document-oriented database was chosen, within which the documents are stored in a JSON format⁵ and organized within groups of documents called collections. This database is set up using *MongoDB* 4.2.1.

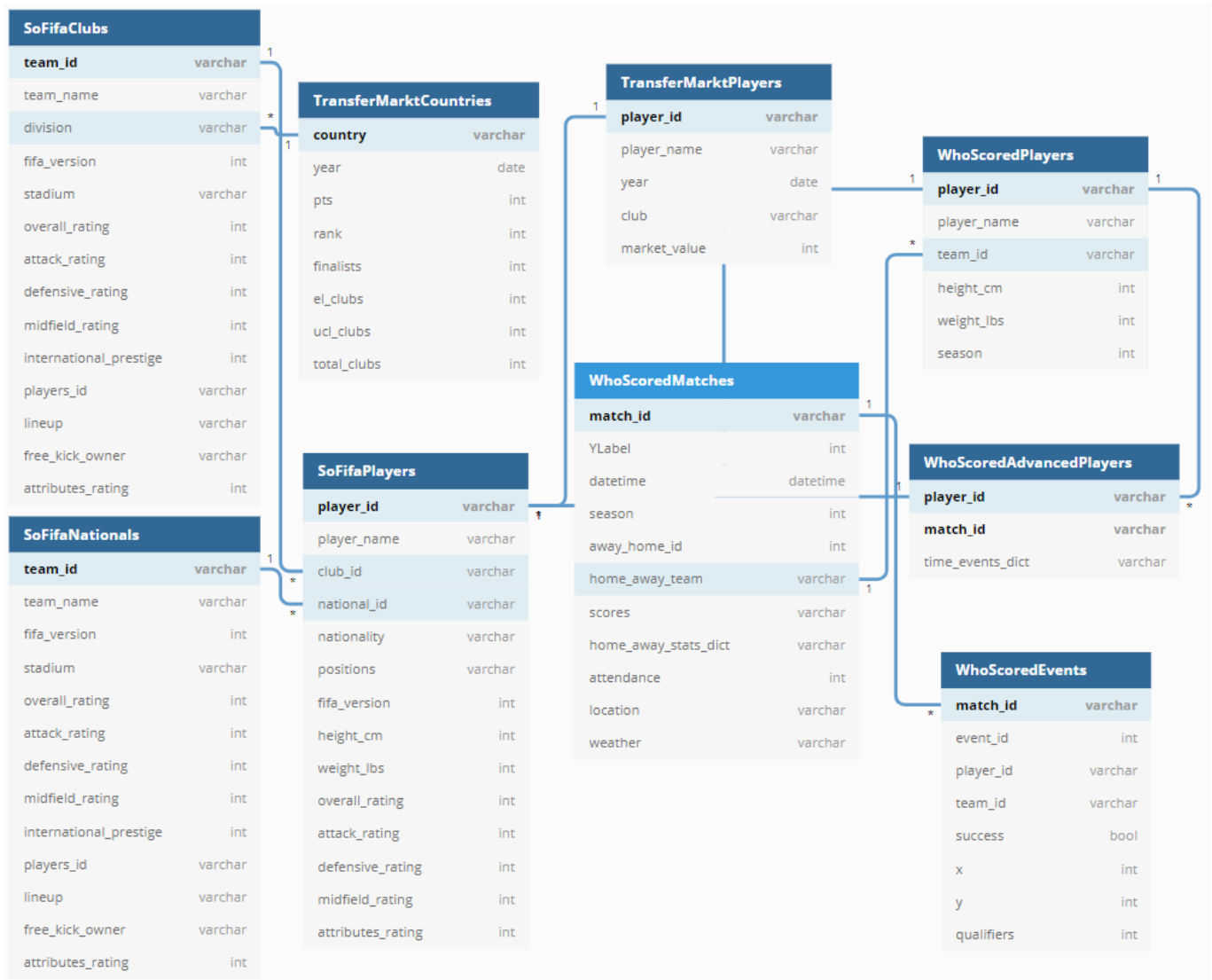


Figure 2.2: Soccer database diagram

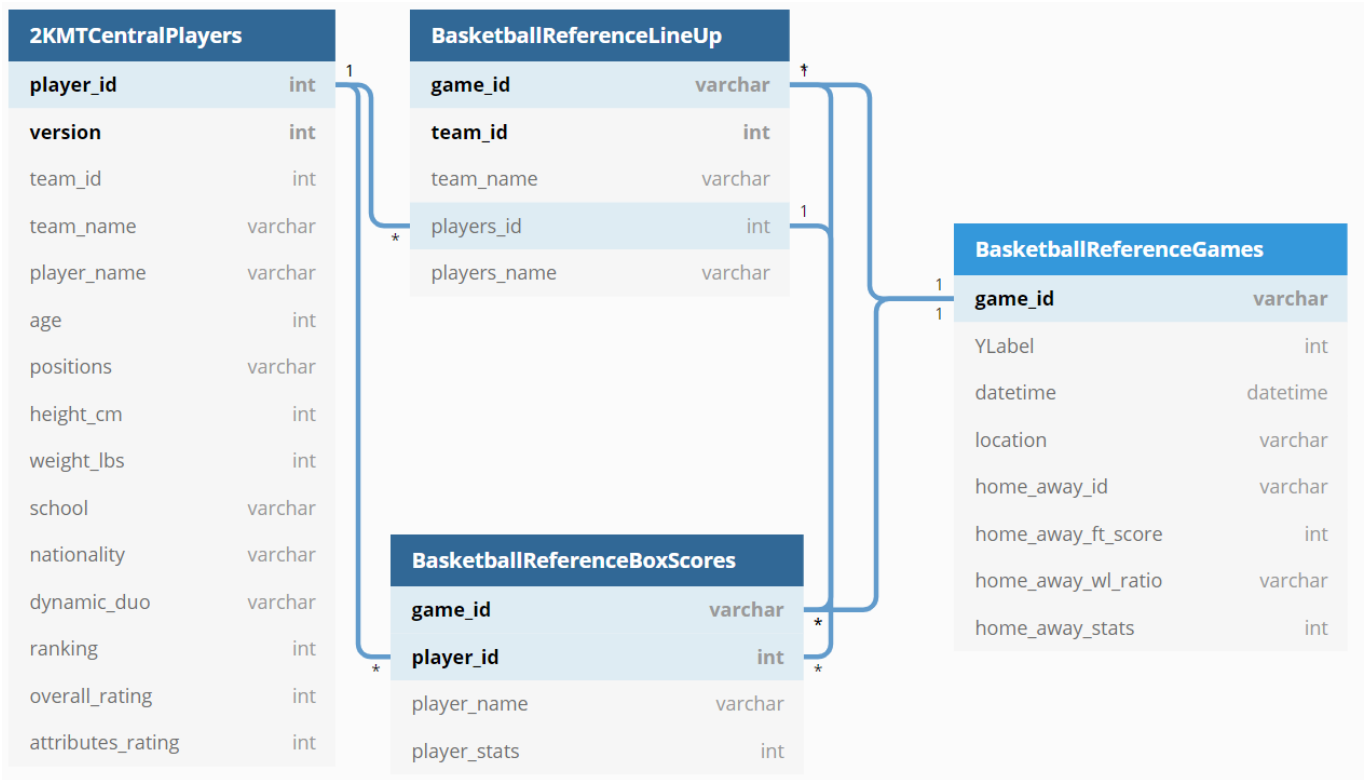


Figure 2.3: Basketball database diagram

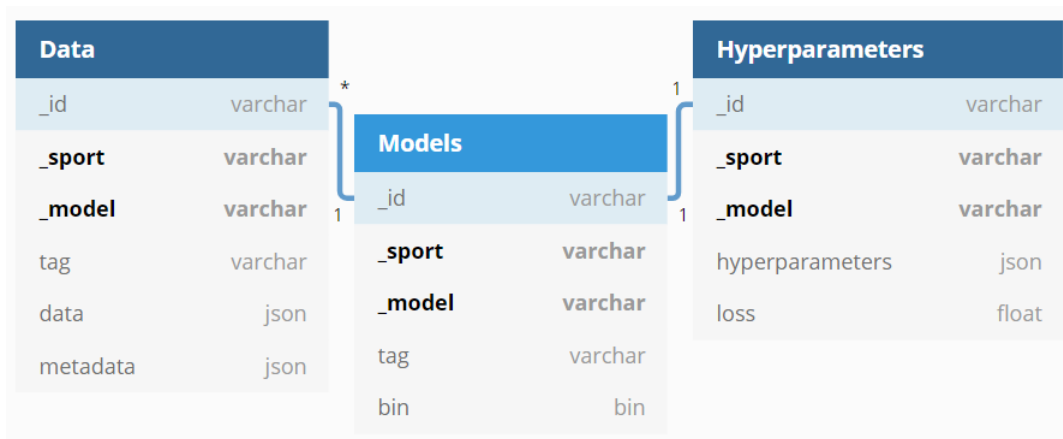


Figure 2.4: Inter-Process Communication database diagram

2.3 Web scraping

In order to complete the relational database described above, it was necessary to extract the content from the aforementioned websites. To achieve this, it was necessary to use different *Python* libraries like *Requests*, to send HTTP requests easily, *Selenium*, to automate tasks in a web browser, and *Beautiful Soup* to parse and extract the content from the explored web pages.

Since each website presents a different tree structure, the access strategies to the content were also different.

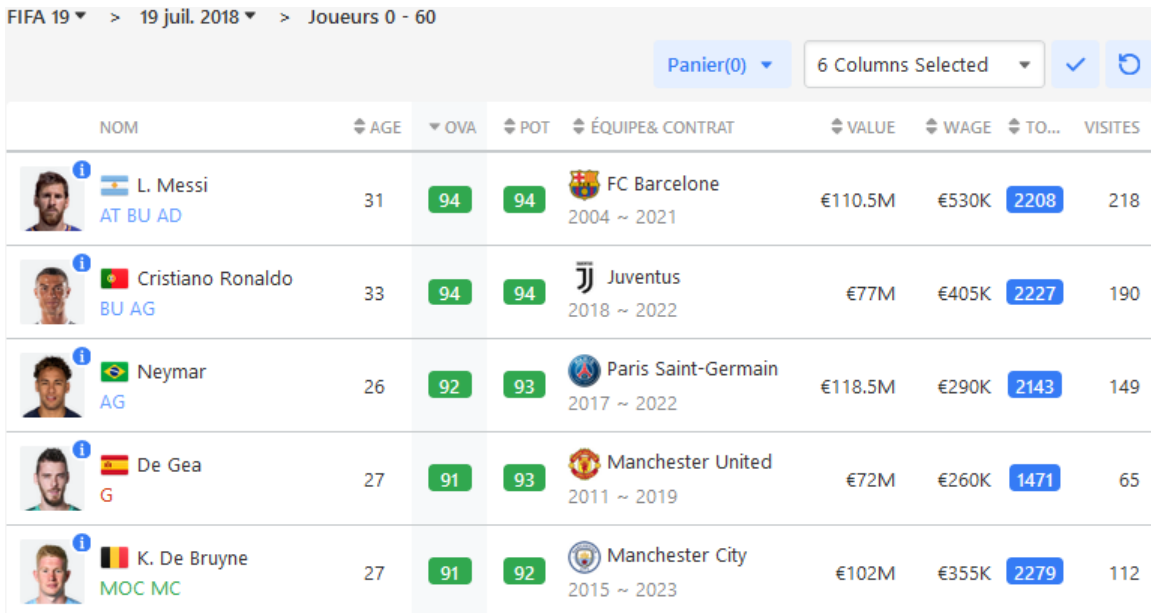
- *SoFIFA*: Whether it is the pages of players, clubs, or national teams, the process was the same. The URLs⁶ of the content of interest were accessible at the following addresses, including a list of players, as shown in Figure 2.5, or teams:

```
https://sofifa.com/CONTENT?type=TYPE&r=VERSION&set=true  
&offset=PLAYERS_COUNT
```

where **CONTENT** can be **players** or **teams**, **TYPE** can be **club** or **national** for the teams and **all** for the players, **VERSION** is a 6-digit code, in which the two first digits are the major version indicating the soccer season concerned (from 07 to 19), and the four last one are the minor version, indicating the update concerned (0001 has always been chosen in order to take the first version of each game that corresponds to the start of the season, which doesn't depend on the form of the players). Finally **PLAYERS_COUNT** is a number indicating the order of the last player / team accessible on the explored page.

⁶a Uniform Resource Locator or web address is a reference to a web resource

By iterating over all of the web pages, whose URLs are composed as described above, around forty URLs of national teams, more than six hundred URLs of clubs and more than sixteen thousand URLs of players are recovered for each version of the game.













NOM	AGE	OVA	POT	ÉQUIPE& CONTRAT	VALUE	WAGE	TO...	VISITES
 L. Messi AT BU AD	31	94	94	 FC Barcelone 2004 ~ 2021	€110.5M	€530K	2208	218
 Cristiano Ronaldo BU AG	33	94	94	 Juventus 2018 ~ 2022	€77M	€405K	2227	190
 Neymar AG	26	92	93	 Paris Saint-Germain 2017 ~ 2022	€118.5M	€290K	2143	149
 De Gea G	27	91	93	 Manchester United 2011 ~ 2019	€72M	€260K	1471	65
 K. De Bruyne MOC MC	27	91	92	 Manchester City 2015 ~ 2023	€102M	€355K	2279	112

Figure 2.5: SoFIFA player list web page

- *Whoscored*: The URLs of the content of interest are accessible at the following addresses, including a list of matches, as shown in Figure 2.6:

`https://whoscored.com/Regions/REGION/Tournaments/
TOURNAMENT/Seasons/SEASON/Stages/STAGE/Fixtures`

where **REGION** is a 2 or 3-digit code corresponding to the country (or geographical limitation concerning international tournaments), **TOURNAMENT** is a 1 or 2-digit code corresponding to the league/league, **SEASON** is a 4-digit

code corresponding to the season and **STAGE** is a 4 or 5-digit code for each league/tournament and season (with differentiate for group and final stages for European tournaments).

By iterating over all of the web pages, whose URLs are composed as described above, 21,556 matches for the English, Spanish, Italian, German, and French top leagues, the Champion's League and the Europa League from 2009/2010 season to 2018/2019 season are recovered.

LaLiga Fixture & Results

Sunday, Dec 1 2019 Dec 2019

▶	11:00	FT	Sevilla	1 : 0	Leganes	3	Match Report
▶	13:00	FT	Athletic Bilbao	2 : 0	Granada	3	Match Report
▶	15:00	FT	Espanyol	2 : 4	Osasuna 1	5	Match Report
▶	17:30	FT	Getafe	4 : 0	Levante		Match Report
▶	20:00	FT	Atletico Madrid	0 : 1	Barcelona	111	Match Report
Friday, Dec 6 2019							
	20:00	FT	Villarreal	0 : 0	Atletico Madrid	8	Match Report
Saturday, Dec 7 2019							
▶	12:00	FT	1 Real Madrid	2 : 0	Espanyol	30	Match Report
▶	15:00	FT	Granada	3 : 0	Deportivo Alaves 2	2	Match Report
▶	17:30	FT	1 Levante	2 : 4	Valencia	4	Match Report
▶	20:00	FT	Barcelona	5 : 2	Mallorca	80	Match Report

Figure 2.6: Whoscored match list web page

- *Transfermarkt*: The URLs of the content of interest are accessible at the following addresses, including a list of clubs, as shown in Figure 2.7:

<https://transfermarkt.com/LEAGUE/startseite/wettbewerb/>

LEAGUE_ID/plus/?season_=SEASON

where **LEAGUE** and **LEAGUE_ID** are a 2-tuple for each league in the following list: (premier-league, GB1), (ligue-1, FR1), (serie-a, IT1), (laliga,

ES1), (bundesliga, L1). **SEASON** is the corresponding season in the following format 20xx.

By iterating over all of the web pages, whose URLs are composed as described above, 49,865 monetary values for the English, Spanish, Italian, German, and French top leagues from 2005/2006 season to 2018/2019 season are recovered.







CLUBS - BUNDESLIGA 18/19					
Club	Squad ↓	ø age ↓	Foreigners ↓	Total market value ↓	ø market value ↓
 Bayern Munich	32	24,7	15	€835.55m	€26.11m
 Borussia Dortmund	34	23,4	19	€376.80m	€11.08m
 Bayer 04 Leverkusen	31	23,6	16	€359.35m	€11.59m
 RB Leipzig	34	21,6	21	€314.45m	€9.25m
 FC Schalke 04	39	24,1	24	€243.90m	€6.25m
 Borussia Mönchengladbach	30	24,2	17	€191.53m	€6.38m

Figure 2.7: *Transfermarkt* club list web page

- *2KMTCentral*: The URLs of the content of interest are accessible at the following addresses, including a list of players, as shown in Figure 2.8:

`https://2kmtcentral.com/VERSION/players/theme/current/
page/PAGE_COUNT`

where **VERSION** is a 2-digit code corresponding to the season and

PAGE_COUNT is the web page index.

By iterating over all of the web pages, whose URLs are composed as described above, around 400 players by season from 2014/2015 season to 2018/2019 season are recovered.

2K19 MyTEAM Players

NAME	OVR	POS	INS	OUT	PLY	ATH	DEF	REB	HEIGHT	PS4
LeBron James <small>'19 Current NBA / '19 Lakers</small>	91	SF PF	82	76	82	86	82	72	6'8"	95 100 44k
Russell Westbrook <small>'19 Current NBA / '19 Thunder</small>	90	PG	79	77	86	90	84	78	6'3"	41 64 26k
Anthony Davis <small>'19 Current NBA / '19 Pelicans</small>	90	PF C	86	77	50	80	84	81	6'10"	21 15 40k
Giannis Antetokounmpo <small>'19 Current NBA / '19 Bucks</small>	90	SF PF	76	67	73	83	80	82	6'11"	51 26 33k
Kawhi Leonard <small>'19 Current NBA / '19 Raptors</small>	90	SF PF	77	78	70	77	84	58	6'7"	32 31 25k

Figure 2.8: 2KMTC player list web page

- *Basketball-Reference*: The URLs of the content of interest are accessible at the following addresses, including a list of matches, as shown in Figure 2.9:

`https://basketball-reference.com/leagues/NBA_SEASON_games-MONTH.html`

where **SEASON** is the corresponding season in the following format 20xx and **MONTH** is the corresponding month of the game.

By iterating over all of the web pages, whose URLs are composed as described above, 24,531 matches of the NBA from 2000/2001 season to 2018/2019 season are recovered.

December Schedule [Share & more](#) ▼

Date	Start (ET)	Visitor/Neutral	PTS	Home/Neutral	PTS		Attend.	Notes
Thu, Dec 1, 2016	7:00p	Dallas Mavericks	87	Charlotte Hornets	97	Box Score	14,471	
Thu, Dec 1, 2016	7:30p	Milwaukee Bucks	111	Brooklyn Nets	93	Box Score	12,675	
Thu, Dec 1, 2016	8:00p	Los Angeles Clippers	113	Cleveland Cavaliers	94	Box Score	20,562	
Thu, Dec 1, 2016	8:00p	Orlando Magic	94	Memphis Grizzlies	95	Box Score	13,344	
Thu, Dec 1, 2016	9:00p	Miami Heat	111	Utah Jazz	110	Box Score	19,073	
Thu, Dec 1, 2016	10:30p	Houston Rockets	132	Golden State Warriors	127	Box Score	20T 19,596	
Fri, Dec 2, 2016	7:00p	Orlando Magic	105	Philadelphia 76ers	88	Box Score	13,711	
Fri, Dec 2, 2016	7:30p	Sacramento Kings	92	Boston Celtics	97	Box Score	18,624	

Figure 2.9: *Basketball-Reference* match list web page

- *ATP Tour*: First, web pages including a list of tournaments are explored to obtain tournaments and their IDs:

`https://atptour.com/en/scores/results-archive?year=YEAR`

where **YEAR** is the corresponding season in the following format 20xx.

Then, URLs tournament web pages, shown in Figure 2.10, are built with the following addresses:

`https://atptour.com/en/scores/archive/TOURNAMENT/`

`TOURNAMENT_ID/YEAR/results`

where **TOURNAMENT** and **TOURNAMENT_ID** are a 2-tuple for each tournament. Then, personal information on players is obtained with the following addresses:

`https://atptour.com/en/players/PLAYER/PLAYER_ID/overview`

where **PLAYER** and **PLAYER_ID** are a 2-tuple for each player. Finally,

rankings are obtained for each week with the following addresses:

`https://atptour.com/en/rankings/singles?rankDate=DATE`

`&rankRange=RANGE`

where **DATE** is the date, for each week, with the following format YYYY-MM-DD and **RANGE** is a range of 100 players with the following format X-X+100. By iterating over all of the web pages, whose URLs are composed as described above, 102,302 matches and 12,023 players of the ATP from 1991 season to 2018 season are recovered.

The screenshot shows the ATP match list web page for Roland Garros 2018. The page header includes the tournament name, location, dates, and prize money. Below the header, there are navigation tabs for Results, Draws, and Top Seeds. A filter section allows users to select Singles or Doubles, Rounds, Player, and Country. The main content area displays match results for the Finals and Semi-Finals.

Roland Garros				
Paris, France		2018.05.27 - 2018.06.10		
SGL 128 DBL 64		Clay	Prize Money €18,232,000	Total Financial Commitment €18,232,000
Results Draws Top Seeds				
Singles	Doubles	Rounds	Player	Country
Finals				
(1)	Rafael Nadal	Defeated	(7)	Dominic Thiem
			64 63 62	H2H
Semi-Finals				
(1)	Rafael Nadal	Defeated	(5)	Juan Martin del Potro
			64 61 62	H2H
(7)	Dominic Thiem	Defeated		Marco Cecchinato
			75 76 ¹⁰ 61	H2H

Figure 2.10: ATP match list web page

Once the URLs are retrieved, the HTML⁷ content of web pages are parsed to keep only the interesting and relevant content. After a preprocessing step (cleaning, editing, reducing and wrangling), data are stored in their corresponding tables within the relational database.

⁷HyperText Markup Language is the language used to define the meaning and the structure of web content

2.4 Schema integration

Because basketball and soccer data are derived from different data sources, the columns used to join the tables do not always perfectly match. For example, the soccer club of the city of Manchester, may be recorded as *Man. United* in one source and *Manchester United* in another. Then, a string similarity metric has been computed between text describing teams and players of each data sources, to approximate string matching.

Defined in 1965 by the Soviet mathematician Vladimir Levenshtein, the Levenshtein distance is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into another:

$$lev(a,b) = \begin{cases} \max(|a|, |b|) & \text{if } \min(|a|, |b|) = 0, \\ lev(a-1, b-1) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} lev(a-1, b) \\ lev(a, b-1) \\ lev(a-1, b-1) \end{cases} & \text{otherwise} \end{cases}$$

where a and b are two strings, $|a|$ is the cardinal of a (number of letters) and $a-1$ the string truncated by its first letter $a[0]$.

Using the *Python* library *TheFuzz*, a unique ID was created between each team of

each national league. Once the teams are associated, the players could be in each team. Then, the data scraped from *SoFIFA*, *Transfermarkt* and *Whoscored* could be joined, and so could those from *Basketball-Reference* and *2KMTCentral*.

2.5 Conclusion

Then, with a volume of more than 7 GiBs in *parquet*⁸ files and nearly 100 columns concerning tennis, more than 150 for basketball and over 300 columns for soccer, with data gathered between 1991 for the oldest entries and 2018, the built database seems to be able to offer an interesting quantity and quality of data for the realization of an efficient model. In addition, the simplicity of its structure allows easy access to the data, limiting the complexity of data query scripts and ensuring their efficiency.

⁸a columnar storage format, taking less size and faster than Comma-Separated Values for big volumes

3. Some models on sport data

Probabilistic forecast models of sports events have been developed through the application of different kinds of methodologies, some have spawned simply out of interest for the sport while some, like Borøy-Johnsen (2017), aim at beating the bookmakers.

Before the appearance of several methods related to machine learning, the initial method used econometric approaches. This chapter focuses on previous research related to quantitative models or analysis on the prediction of sports events. It presents the main studies on predicting the outcome or the final score of a match, as well as the studies that led to the building of relevant features for sport event models.

3.1 Match scores

The first models concerned with soccer were interested in determining the number of goals (scored and conceded) that would occur during a match using the Poisson distribution, or extensions thereof.

The Poisson distribution, to model football goals

Developed by the French mathematician Poisson (1837), the Poisson distribution models the number of successes occurring in a given time interval or a specified region of space. Moreover, this distribution supposes that the time intervals between successive events are independent of each other.

Then, a discrete random variable X is said to have a Poisson distribution, with parameter $\lambda > 0$, if it has a probability mass function given by:

$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}$$

where k is the number of successes ($k = 0, 1, 2, \dots$)

e is the Euler's number ($e = 2.71828$)

λ is the mean number of successes in the given time interval

or region of space

Using this distribution and its extensions, it can be possible to predict the probabilities associated with the outcomes of a football match using the number of goals scored and conceded in a match.

Moroney (1956) was the first to present a study on predicting the outcome of football matches. He considered the number of goals scored during a football match as a case of isolated events in a continuum of time. Then, he rejected the use of a Binomial Distribution, because the number of trials n in the binomial experiment is unknown: we can only count the number of times a goal occurs but it's impossible to count the number of times it did not. He therefore preferred using a Poisson Distribution, using the average of goals per team per match recorded on the 240 matches of the study as mean ($\lambda = 1.7$). According to the author, since this mean varies from trial to trial, due to weather factors or the team-matching, the initial formula was modified by incorporating the variance of goals per team per match, σ^2 , in the equation. The probability mass function of this modified Poisson distribution is given by:

$$P(X = k) = \left(\frac{c}{c+1}\right)^p \left(\frac{\frac{(p+k-1)!}{(p-1)!}}{k!(c+1)^k}\right)$$

where k is the number of successes ($k = 0, 1, 2, \dots$)

$$c = \frac{\bar{x}}{\sigma^2 - \bar{x}}$$

$$p = \bar{x} \cdot c$$

where X is the random variable counting the number of goals of a team during a match, $\bar{x} = 1.7$ and $\sigma^2 = 1.9$.

Using this modified Poisson distribution, he then obtained a predicted frequency of goals very close to the actual frequency for the 240 matches considered, using the following.

Afterwards, Reep, Pollard, and Benjamin (1971) preferred to call the modification of the Poisson used by Moroney (1956) a "compound Poisson". They also used a Negative Binomial distribution with English Football League First Division data from 1965 to 1969 for a study on 42 matches per season. But these approaches do not consider the quality of the team nor the quality of the opposition, in accordance with the remark of Reep and Benjamin (1968) that "chance does dominate the game", made after a study on passing and shooting areas of English Football League First Division from 1953 to 1967. Following this, by showing a significant positive correlation between forecasts made by experts at the beginning of the season, and the final league tables of the 1971-1972 English football season, Hill (1974) indicates that football results are not pure chance.

The Double Poisson distribution, to model team goals in a specific match

For Maher (1982), "over a whole season, skill rather than chance dominates the game". He considered the importance of possession in a football match, by considering that each possession has a probability p to be concluded by a goal. And even if p is small, the number of possession n in a match can be enormous. Considering p is constant and attacks are independent, he also used a Poisson distribution. Therefore, if team i is playing at home against team j , and the observed score is (x_{ij}, y_{ij}) , the final score

can be modeled with the two following independent Poisson:

$$X_{ij} \sim \text{Poisson}(\alpha_i \beta_j)$$

$$Y_{ij} \sim \text{Poisson}(\gamma_i \delta_j)$$

where α_i is the strength of team i 's attack when playing at home
 β_j is the weakness of team j 's defence when playing away
 γ_i is the weakness of team i 's defence when playing at home
 δ_j is the strength of team j 's attack when playing away

Since X_{ij} and Y_{ij} are assumed to be independent ("representing separate "games" at the two ends of the pitch"), $\boldsymbol{\alpha}^T = (\alpha_0, \dots, \alpha_n)$ and $\boldsymbol{\beta}^T = (\beta_0, \dots, \beta_n)$ can be estimated only for x , and $\boldsymbol{\delta}^T = (\delta_0, \dots, \delta_n)$ and $\boldsymbol{\gamma}^T = (\gamma_0, \dots, \gamma_n)$ only for y . Then, for the home teams' scores, the log likelihood function is:

$$\log L(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_i \sum_{j \neq i} - \left(\alpha_i \beta_j + x_{ij} \log(\alpha_i \beta_j) - \log(x_{ij}!) \right)$$

and so, because no analytical solution is possible, the maximum likelihood estimates $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ satisfy:

$$\hat{\alpha}_i = \frac{\sum_{j \neq i} x_{ij}}{\sum_{j \neq i} \hat{\beta}_j}$$

$$\hat{\beta}_j = \frac{\sum_{i \neq j} x_{ij}}{\sum_{i \neq j} \hat{\alpha}_i}$$

Using the Newton-Raphson method, Maher (1982) determined the maximum likelihood estimates, and applied a similar method for $\hat{\delta}$ and $\hat{\gamma}$ using y_{ij} for each team, using their previous performances. It was the first paper that had modeled football scores between specific teams, accounting for the differences in quality of the teams involved.

Using English Football League division data from 1973 to 1975, he showed that separate parameters for the quality of a team at home and away was not necessary, and preferred to keep only α and β , to describe the quality of the team's offense and the weakness of the team's defence, whether the team is playing at home or away. Moreover, he identified a home ground advantage, equal for all teams. By analyzing the frequencies of goal scores, he detected an underestimation to forecast one or two goals, an overestimation in predicting more than four goals, and, unlike to applications requiring Zero-Inflated Poisson because random event contains excess zero-count data in unit time, an overestimation in predicting 0 goals.

Later, with a particular interest in sports betting Dixon and Coles (1997) reused the independent Poisson model presented by Maher (1982) and added a home advantage:

$$X_{ij} \sim \text{Poisson}(\alpha_i \beta_j H)$$

$$Y_{ij} \sim \text{Poisson}(\alpha_j \beta_i)$$

where α_k and β_k are the offensive and defensive strengths of team k , and H the home ground advantage parameter.

Because this independent model is bad at predicting low-scoring matches (equal to 1

goal or less for each team), they added a dependence parameter ρ :

$$\Pr(X_{ij} = x, Y_{ij} = y) = \tau_{\lambda, \mu}(x, y) \frac{\lambda^x \exp(-\lambda)}{x!} \frac{\mu^y \exp(-\mu)}{y!}$$

$$\text{where } \lambda = \alpha_i \beta_j H \quad \mu = \alpha_j \beta_i$$

$$\tau_{\lambda, \mu}(x, y) = \begin{cases} 1 - \lambda\mu\rho & \text{if } x = y = 0 \\ 1 + \lambda\rho & \text{if } x = 0, y = 1 \\ 1 + \mu\rho & \text{if } x = 1, y = 0 \\ 1 - \rho & \text{if } x = y = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$\text{and } \max(-1/\lambda, -1/\mu) \leq \rho \leq \min(1/\lambda\mu, 1)$$

with the dependence parameter ρ , equals to 0 for independence.

Dixon and Coles (1997), suggesting that a team's performance was dynamic and varied between periods, reduced the contribution of older data, to obtain offensive and defensive team strengths more up to date with recent match performances. Then, they used a 'pseudolikelihood' for each time point:

$$L_t(\alpha_i, \beta_i, \sigma, H; i = 1, \dots, n) = \prod_{k \in A_t} \{ \tau_{\lambda_k, \mu_k}(x_k, y_k) \exp(-\lambda_k) \lambda_k^{x_k} \exp(-\mu_k) \mu_k^{y_k} \}^{\phi(t-t_k)}$$

$$\text{with } \phi(t) = \exp(-\xi t)$$

where t_k is the time when the match k was played, $A_t = \{k : t_k < t\}$, and ϕ is a non-increasing function of time.

For this study, Dixon and Coles (1997) used scores from 6629 full-time league and cup matches from the season that took place between 1992 and 1995.

Rue and Salvesen (2000) used a modified independent Poisson model, as proposed by Dixon and Coles (1997), and included a psychological effect of underestimation of the weaker team by the stronger team. A measure of the difference in strength between the two teams is then proposed using the following expression:

$$\Delta_{ij} = \frac{(\alpha_i + \beta_i - \alpha_j - \beta_j)}{2}$$

This last one is weighted by a small constant γ giving the magnitude of the psychological effect. The difference in strength should not be too important, because the teams opposed during a match are in the same league, it is reasonable to expect $\gamma > 0$ (the opposite effect, $\gamma < 0$, which suppose that a team is so superior compared to the other one, that the latter develops an inferiority complex, is not expected when teams are in the same league). Rue and Salvesen (2000) did not use a home ground advantage, but also corrected low-scoring matches like Dixon and Coles (1997) with a dependence parameter $\sigma = -0.1$, and truncated the Poisson law after five goals, arguing that the number of goals beyond this threshold provided no further information about the offensive and defense strengths of a team. In addition, they suggested that not all information was included in the final score, and added an ϵ parameter, defining how much the league average should contribute to the prediction.

This is expressed as follows:

$$\begin{aligned} \Pr(X_{ij} = x, Y_{ij} = y) = & (1 - \epsilon) \Pr'(X_{ij} = x, Y_{ij} = y \mid \lambda, \mu) \\ & + \epsilon \Pr'(X_{ij} = x, Y_{ij} = y \mid \exp(c^x), \exp(c^y)) \end{aligned}$$

where $\exp(c^x)$ and $\exp(c^y)$ in the non-informative part, are the average goal intensities.

In addition Rue and Salvesen (2000) allowed for the offensive and defensive variables to vary with time, by overweighting the most recent results, using Brownian motion:

$$\alpha_k^{t'} = \alpha_k^t + \left\{ B_{\alpha,k} \left(\frac{t'}{\tau} \right) - B_{\alpha,k} \left(\frac{t}{\tau} \right) \right\} \frac{\sigma_{\alpha,k}}{\sqrt{1 - \gamma(1 - \gamma/2)}}$$

where $t' \geq t$ are two following time points, $\sigma_{\alpha,k}^2$ is the prior variance for α_k for team k . $\{B(t), t \geq 0\}$ is standard Brownian motion starting at level 0 and τ is a loss of memory rate parameter, common to all teams.

Finally Rue and Salvesen (2000) used a Bayesian network, using Bayesian methods to update the estimates after each match and Markov Chain Monte Carlo techniques to draw inferences from the network, with Premier League and Division 1 data from 1997 to 1998. Later on, Crowder et al. (2002) still with a model based on Dixon and Coles (1997), replaced the Markov Chain Monte Carlo procedure usage, considered as too slow, with an approximation, considering the model as a non-Gaussian state space model with time-varying offensive and defensive strengths.

To conclude, Angelini and Angelis (2017) used a Poisson AutoRegression with exogenous covariates (PARX) to take into account the results of the previous games

and improve the estimation of Dixon and Coles (1997), applying this technique to English football Premier League data from 2013 to 2016.

The Bivariate Poisson distribution and the stacked Bayesian regression model, to model score difference in a specific match

The Poisson distribution has also been extended to the bivariate case. Then, two discrete random variables X and Y following a Bivariate Poisson distribution, with parameters $\lambda_0, \lambda_1, \lambda_2$, have a joint probability mass function given by:

$$P(X = x, Y = y) = e^{-(\lambda_1 + \lambda_2 + \lambda_0)} \frac{\lambda_1^x}{x!} \frac{\lambda_2^y}{y!} \sum_{i=0}^{\min(x,y)} \binom{x}{i} \binom{y}{i} i! \left(\frac{\lambda_0}{\lambda_1 \lambda_2}\right)^i$$

$$\text{where } X \sim \text{Poisson}(\lambda_1 + \lambda_0)$$

$$Y \sim \text{Poisson}(\lambda_2 + \lambda_0)$$

By relaxing the independence assumption between scores, Maher (1982) considered the difference in the number of goals as two dependent parts:

$$Z_{ij} = X_{ij} - Y_{ij}$$

$$\text{with } X_{ij} = U_{ij} + W_{ij} \quad \text{and} \quad Y_{ij} = V_{ij} + W_{ij}$$

where U_{ij} , V_{ij} , and W_{ij} are independent Poisson with means of $(\mu_{ij} - \eta_{ij})$, $(\lambda_{ij} - \eta_{ij})$, and η_{ij} respectively, with η_{ij} being the co-variance between X_{ij} and Y_{ij} . After testing a range of values, Maher (1982) used $\eta_{ij} = 0.2$, and got considerably better results. Despite this improvement, the underestimation of draw matches is persistent.

Later on, Lee (1997), using complete scores of all the 380 games played in the 1995-1996 season in English Premier League, and Karlis and Ntzoufras (1997), using data from the 24 championships of different European countries, demonstrated the (relatively low) correlation between the number of goals scored by the two opponents.

Based on the model of Maher (1982), Karlis and Ntzoufras (2003) preferred to add the correlation factor directly in the distribution:

$$(X_{ij}, Y_{ij}) \sim \text{BivariatePoisson}(\lambda_i, \lambda_j, \sigma)$$

$$\text{with } \log(\lambda_i) = \mu + H + \alpha_i + \beta_j \quad \text{and} \quad \log(\lambda_j) = \mu + \alpha_j + \beta_i$$

where λ_i and λ_j represent respectively the expected number of goals scored for the home and away teams, and σ is the correlation factor. μ is a constant parameter representing the average number of goals scored per team when two teams have similar strengths, H is the home team effect parameter, and α_k and β_k are the offensive and defensive abilities of team k . The difference of goals scored during a match can then be expressed using the variable $X_{ij} - Y_{ij}$, which implies the match result.

To fix the issue of underestimating draw games, Karlis and Ntzoufras (2003) purposed a diagonal inflated model, generalizing the multivariate zero-inflated model of Li et al. (1999) on all draw results. Then, they included additional parameters to inflate low-scoring draws and deflate the other probabilities. For that paper, Karlis and Ntzoufras (2003) used Serie A data from 1991 to 1992.

Koopman and Lit (2012) used a similar model, and defined the offensive and defensive

performances of a team as a stochastic function of time:

$$(X_{ij}, Y_{ij}) \sim \text{BivariatePoisson}(\lambda_i, \lambda_j, \sigma)$$

$$\text{with } \lambda_{i,ijt} = \exp(H + \alpha_{it} + \beta_{jt}) \quad \text{and} \quad \lambda_{j,ijt} = \exp(\alpha_{jt} + \beta_{it})$$

$$\alpha_{kt} = \mu_{\alpha,k} + \phi_{\alpha,k}\alpha_{k,t-1} + \eta_{\alpha,kt} \quad \text{and} \quad \beta_{kt} = \mu_{\beta,k} + \phi_{\beta,k}\beta_{k,t-1} + \eta_{\beta,kt}$$

where, for a team k , $\mu_{\alpha,k}$ and $\mu_{\beta,k}$ are unknown constants, $\phi_{\alpha,k}$ and $\phi_{\beta,k}$ are autoregressive coefficients, which control the intensity to change over time since the composition and the performance of the teams will change, and $\eta_{\alpha,kt}$ and $\eta_{\beta,kt}$ are normally distributed independent error terms. α_{kt} and β_{kt} are determined using the maximum likelihood estimator.

Also using score differences to deduce match results, Lam (2018) proposed a pioneering model, based on stacked Bayesian regressions, by training it with match data in the NBA 2013/2014 regular season and predicting the points scored by each team over 1,230 matches in the following 2014/2015 season of NBA. First, he inferred player's ability from player's previous performance using exponential smoothing, arguing that a player's performance and his true ability are different because sometimes, players make good use of their talents but sometimes, they do not. After, because the simple fact of concatenating the ability vectors of all the players of a team will produce a vector dimension that is too high, and using a dimensionality reduction methods will not encode the domain knowledge from the sport itself, he preferred training a Bayesian regression to estimate, for each player position, an estimator that represents the player's contribution to his team. Then, the team strength was expressed as

the concatenation of those estimations rather than that of all the abilities of every player. Now, that a quantitative comparison of two opposing teams is made, a second regression task is used, to determine the differential points of the next game.

The use of copulas as a solution to the goal dependency of each team

Mchale and Scarf (2011) used also a discrete bivariate distribution in their discussion over the impact of competitiveness on the dependence between the number of goals scored by the 2 opposing teams. Since most of the work used data from national championships, the level of the teams was balanced and, by definition, the matches more competitive. The dependence between the goals scored by the teams was therefore less important. Mchale and Scarf (2011) preferred to use data from 6101 international soccer matches between 1993 and 2004, in order to obtain a larger panel of matches, including less competitive ones. But faced with the impossibility of using a negative correlation in the studies cited above (such as Karlis and Ntzoufras (2003)) and the obligation to use marginal Poisson distributions, they proposed to use copulas, in order to generate different bivariate dependent discrete distributions and to predict the scores of each team.

According to the Sklar's theorem, the joint distribution function F of any pair of random variables (Y_1, Y_2) can be expressed as follows:

$$F(x, y) = C(F_1(x), F_2(y)), \quad (x, y) \in R^2$$

where the copula C is a multivariate cumulative distribution function, defined on

$[0, 1]^2$, and F_1 and F_2 are uniform marginal probability distribution, defined on $[0, 1]$ Mchale and Scarf (2011) used a Frank copula, which has the particularity of being symmetrical on its lower and upper tails, and 2 Negative Binomial distributions or 2 Poisson distributions to model the bivariate random variable of the number of goals during a soccer match. Thus, the probabilities of the results of a soccer match can be obtained using the following relation:

$$P(X_{ij} = x, Y_{ij} = y) = C_{\theta}(F_1(x), F_2(y)) - C_{\theta}(F_1(x-1), F_2(y)) \\ - C_{\theta}(F_1(x), F_2(y-1)) + C_{\theta}(F_1(x-1), F_2(y-1))$$

$$\text{with } C_{\theta}(u, v) = -\kappa^{-1} \times \log \left(1 - \frac{(1 - e^{-\kappa u})(1 - e^{-\kappa v})}{(1 - e^{-\kappa})} \right)$$

where κ is a real number, used as a dependency parameter in the Frank copula.

Subsequently, Wurp et al. (2019) used a set of 5 copulas (Frank, Gumbel, Joe, Gaussian, Clayton and its 90-rotated version), and marginal Poisson distributions, which he proposed to penalize. With the help of 320 matches taking place during the soccer World Cups between 2002 and 2018, he showed the interest of considering the dependence of the goals scored by the teams with the help of copulas, in the prediction of the score of a soccer match.

A point-based approach, to model tennis scores

Specifically designed around the rules of tennis, this type of model assumes that the probability of winning a point is fixed throughout the match for each server. Then,

it is possible to calculate the winning probabilities for a game, a set, a match, or a tournament by summing up all the ways of winning. Then, Newton and Keller (2005) defined the following probabilities, using data from the 2002 U.S. Open and Wimbledon tournaments:

Probability of winning a game	$p_A^G = (p_A^R)^4 [1 + 4q_A^R + 10(q_A^R)^2] + 20(p_A^R q_A^R)^3 (p_A^R)^2 [1 - 2p_A^R q_A^R]^{-1}$
Probability of winning a tie-break	$p_A^T = \sum_{j=0}^5 p_A^T(7, j) + p_A^T(6, 6) p_A^R q_B^R [1 - p_A^R p_B^R - q_A^R q_B^R]^{-1}$
Probability of winning a set	$p_A^S = \sum_{j=0}^4 p_A^S(6, j) + p_A^S(7, 5) + p_A^S(6, 6) p_A^T$
Probability of winning a match	$p_A^M = \begin{cases} (p_A^S)^2 + 2(p_A^S)^2 p_B^S & \text{if it is a two out of three set format} \\ (p_A^S)^3 + 3(p_A^S)^3 p_B^S + 6(p_A^S)^3 (p_B^S)^2 & \text{if it is a three out of five set format} \end{cases}$

where p_A^R , the probability that player A wins a rally when he serves against player B is obtained using empirical data. $q_A^* = 1 - p_A^*$, $p_A^*(i, j)$ are then calculated using

recursion formulas and similarly for player B . As Newton and Keller (2005) did, because the probability p_A^R that A wins a rally on serve depends upon the opponent B as well as upon A , if data are not available for A serving to B , data for A playing against players similar to B can be used.

For example, Champagne and Gerville-Réache (2015) used these formulas to simulate the season of a tennis player, in order to observe if the French tennis ranking method is sensitive to the number of matches a player plays.

But this hypothesis does not take into account the "first game effect", i.e. that the first game of a match is the hardest one to break, highlighted by Magnus and Klaassen (1999) and the "hot-hand" phenomenon i.e. that the chances of winning a point or a game increase when the player wins the previous one, or the opposite "back-to-the-wall" effect. Then, the point independence hypothesis can be relaxed, and it can be more realistic to use the following probability that player A wins a point on serve as:

$$\hat{p}_A^R = p_A^R + \delta p_{AB}^R(i, j)$$

where p_A^R is constant through the match, $p_{AB}^R(i, j)$ is player A 's probability of winning a point on serve against player B , when the score is i points for player A and j points for player B , and δ is a small weight.

Later, Barnett and Clarke (2005) proposed to use an opponent-adjustment, and estimated an advantage or disadvantage for each player, comparing him to the

average tour player:

$$f_A = a_A b_A + (1 - a_A) c_A$$

$$\text{and } g_A = a_{av} d_A + (1 - a_{av}) e_A$$

where f_A is the percentage of points won on serve for player A , g_A is the percentage of points won on return for player A , a_A is the percentage of first serves in play for player A , b_A is the percentage of points won on first serves given that the first serve is in for player A , c_A is the percentage of points won on second serves for player A , d_A is the percentage of points won on returns of first serves for player A , e_A is the percentage of points won on returns of second serves for player A and a_{av} is the first serve percentage for ATP tour averages. Then, by combining player statistics, they proposed the two following formulas:

$$f_{AB} = f_t + (f_A - f_{av}) - (g_B - g_{av})$$

$$\text{and } g_{BA} = g_t + (g_B - g_{av}) - (f_A - f_{av})$$

where f_{AB} is the combined percentage of points won on serves for player A against player B , g_{BA} is the combined percentage of points won on returns for player B against player A , t denotes the specific tournament averages and $f_{AB} + g_{BA} = 1$.

3.2 Match outcomes

In parallel to the development of the previously exposed models, estimating the number of goals per team per match for soccer or the number of points per player for tennis, an approach that directly estimates the result of a match has been developed, using discrete choice models.

The Ordered Probit model

Initially used in bio-statistics (see Aitchison and Silvey (1957)), before finding applications in social sciences (see McKelvey and Zavoina (1975)), the ordered probit model generalizes the probit model (see Bliss (1934)) to the case of more than two outcomes of an ordinal dependent variable.

Then, defining Y as the dependent ordinal variable with m categories, X as the vector of independent variables and Y^* as the latent dependent variable, the ordered probit model is characterized by the following equations:

$$\Pr(Y = j | X) = \begin{cases} \Phi(\mu_0 - X^T \beta) & \text{if } j = 0, \\ \Phi(\mu_j - X^T \beta) - \Phi(\mu_{j-1} - X^T \beta) & \text{if } 0 < j < m, \\ 1 - \Phi(\mu_{m-1} - X^T \beta) & \text{if } j = m \end{cases}$$

where β is the vector of regression coefficients, typically estimated by maximum likelihood, and $\forall k \in [0, m]$, μ_k are the unknown threshold parameters to be estimated with β . Φ is the cumulative distribution function of the standard normal distribution.

Then, Koning (2000) used an ordered probit model with three categories (a first one for the win, a second one for the draw and a last one for a the loss), on match results from Dutch professional soccer league data from 1955 to 1999:

$$Y_{ij}^* = \alpha_i - \alpha_j + h_{ij} + \eta_{ij}$$

$$Y = \begin{cases} -1 & \text{if } Y_{ij}^* \leq \mu'_0, \\ 0 & \text{if } \mu'_0 < Y_{ij}^* \leq \mu'_1, \\ 1 & \text{if } Y_{ij}^* > \mu'_1 \end{cases}$$

where the latent variable Y_{ij}^* is a random walk determining the outcome of the game, α_i measures the strength of team i , and is independent of both the opponent and the venue of the game, and is assumed to be constant throughout the season. h_{ij} is the home ground advantage of team i over team j which is assumed to be normally distributed with mean h and η_{ij} is a mean 0 random variable that captures other determinants of the results. The observed outcome Y_{ij} is equal to 1 if the home team wins, 0 if the outcome is a draw and to -1 if the away team wins the game.

Capable of achieving a forecasting performance, this model has proved to be convincing by its simplicity compared to scores forecasting models and was used by Kuypers (2000), Forrest and Simmons (2000), Goddard and Asimakopoulos (2004) and Graham and Stott (2008) among others, with some minor changes on the construction of the latent variable or on the values that Y_{ij} can take.

The Logistic model

Klaassen and Magnus (2003) predicted the outcomes of Wimbledon matches from 1992 to 1995, using a simple logit model, as follows:

$$\Pr(A > B) = \frac{\exp(\lambda(R_A - R_B))}{1 + \exp(\lambda(R_A - R_B))}$$

$$\text{with } R_k = 8 - \log_2(RANK_k) \quad \text{for } k \in \{A, B\}$$

where $RANK_k$ is the ATP ranking of player k and player A is the highest ranking player. The comparison $A > B$ can be read as "player A beats player B " and R_A is the "expected round" of player A . The log-transformation of the ATP ranking is preferred to its raw value because quality in tennis is considered to be like a pyramid (the difference between the two top players is larger than that between two players ranked 101st and 102nd). Finally, λ is estimated by a maximum of likelihood.

The Bradley-Terry model

Already studied by Zermelo (1929), the model as presented by Bradley and Terry (1952) has found its first applications in ranking documents by relevance, reflecting that if a document is more relevant than another for a specific user's query, it should be displayed earlier in the result list. The initial Bradley-Terry model is used to model paired comparisons for a binary outcome:

$$\Pr(i > j) = \frac{\alpha_i}{\alpha_i + \alpha_j}$$

where α_i and α_j represent respectively the skill of team i and j and the comparison $i > j$ can be read as " i beats j ".

Having no specification in its most natural version, this paired comparison model can be applied to any sport. For example, Mchale and Morton (2011) used it to estimate the abilities of tennis players from a likelihood of games won and lost, using an exponential decay function to weight more heavily more recent matches. This likelihood was also stratified by surface to obtain surface-specific abilities.

Then, several extensions on this model have been proposed to be more accurate for soccer cases for example, such as Rao and Kupper (1967) or Davidson (1970) to handle ties, Agresti and Kateri (2002) to include home advantage, Huang, Weng, and Lin (2006) to use player rankings or Critchlow and Fligner (1991) to introduce covariates. But Cattelan, Varin, and Firth (2013) used this model on football and basketball data for the first time, using Italian Serie A football league data from 2008 to 2009 and NBA regular season data from 2009 to 2010. They used a dynamic Bradley-Terry model, in which they defined $\alpha_i(t)$ as the ability of the home team i at time t , which evolves in time following the exponentially moving average process using only previous matches played at home:

$$\alpha_i(t) = \lambda_1 \mu_i(t) + (1 - \lambda_1) \alpha_i(t^{-1})$$

$$\text{with } \mu_i(t) = \beta_1 r_i(t^{-1})$$

where $\mu_i(t)$ denotes the mean home ability of team i based only on the result of the nearest previous match played at home by i , $\lambda_1 \in [0, 1]$ is the home-specific smoothing

parameter, β_1 is a home-specific parameter and $r_i(t^{-1})$ is a variable measuring the result of team i in the previous match played at home at time t^{-1} . As the home team i has played K matches at home before the match played at time t , it can be possible to reformulate previous equations by back-substitution and obtain:

$$\alpha_i(t) = \beta_1 \left[\lambda_1 \sum_{k=0}^{K-1} (1 - \lambda_1)^k r_i(t^{-(k-1)}) + (1 - \lambda_1)^K \bar{r}_i \right]$$

Then, each team's home ability is defined with the entire previous history of home matches, and the ability to play away is similarly modeled. Finally, Cattelan, Varin, and Firth (2013) estimated the outcome of each match as follows:

$$\Pr(Y_k \leq y_k | Y_{k-1} = y_{k-1}, \dots, Y_1 = y_1) = \frac{\exp(\delta_{y_k} + \alpha_i(t) - \alpha_j(t))}{1 + \exp(\delta_{y_k} + \alpha_i(t) - \alpha_j(t))}$$

where $y_k \in \{0, 1, 2\}$ denotes the outcome of the match (2 for home team victory, 1 for draw and 0 for away team victory) and δ_{y_k} are cut point parameters, where $\delta_0 < \delta_1 < \delta_2$. Despite good results, Cattelan, Varin, and Firth (2013) stated that their model only used information about the final result of previous matches, and that using more detailed information about previous matches may result in a more accurate fitting and better forecasts.

The rise of Machine Learning

Beckler, Wang, and Papamichael (2008) applied methods like linear regression or logistic regression, but also ones more related to Machine Learning like Support Vector Machines or Artificial Neural Networks on NBA data from 1991 to 1997, using

team-centric and player-centric features. Later, Torres (2013), also used NBA data focused on team-centric features from 2006 to 2012 with the same classifiers, and obtained better results with linear classifiers.

For soccer, Constantinou, Fenton, and Neil (2012) used their pi-football (probabilistic intelligence football) model on English Premier league data from 1993 to 2010. This model is a Bayesian network model, in which they updated an objective forecast based on teams' strengths (determined by the team's points) with a "subjective proximity", based on form, psychology and fatigue of the two teams, by means of experts' advice.

3.3 Conclusion

As previously shown with Moroney (1956), scientific expertise has been used in sports for many years. However, before the term "sports analytics" was used, the practice of applying mathematical and statistical principles to sports was mainly found in statistics papers or in applied sciences like econometrics. This may explain why, in view of the works described above, there are few specific academic journals or conferences on the field.

But whether it is with the help of the score, or by wishing to directly predict the result of an encounter, a common feature emerges from the works previously presented: over time, with an increasing volume and accessibility of data and the evolution of technical means to use this data, the quantity of data used in sports event probability forecast models increased hand in hand with the performances of these models.

4. Background on some learning strategies

By proposing a probability distribution over a set of classes, rather than only outputting the most likely class that the observation should belong to, the probabilistic classifiers have been identified as the best way to address our problem. But the simple use of these algorithms is not sufficient to obtain high quality results. This chapter will then present and explain algorithms and methods used to prepare and classify data.

4.1 Probabilistic classification algorithms

“Supervised learning” as defined by *DeepAI*, a company that gathers all the news and research related to Artificial Intelligence, is “a class of systems and algorithms that determines a predictive model using data points with known outcomes, in which the model is learned by training through an appropriate learning algorithm . . . that typically works through some optimization routine to minimize a loss or error

function." In contrast, "unsupervised learning" does not require known outcomes.

Classification is a supervised learning task in which the observations are classified in a set of finite labels. This set can comprise two groups (binary classification) or more (multiclass classification).

4.1.1 Logistic regression

Logistic regression is a Generalized Linear Model model used for classification in which the possible outcomes of an observation are modeled using a logistic function. Initially defined for binary classification by Berkson (1944), this model is the equivalent of linear regression for the classification case, in which we transform the linear relation by a sigmoid function (Figure 4.1):

$$\hat{y}(w, X) = f(w_0 + w_1X_1 + \dots + w_pX_p)$$
$$\text{with } f(x) = \frac{1}{1 + e^{-x}} \quad \forall x \in \mathbb{R}$$

where \hat{y} is the predicted output, the sigmoid function f is the cumulative distribution function of the logistic distribution of location 0 and scale 1, p is the number of features, w_0 is the intercept, $X = (X_1, \dots, X_p)$ are the features and $w = (w_1, \dots, w_p)$ the weights.

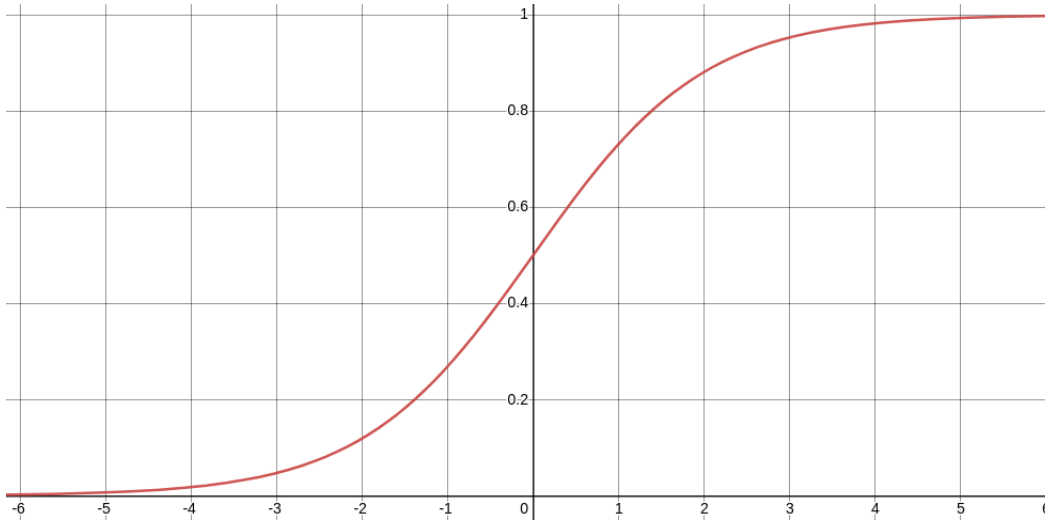


Figure 4.1: Sigmoid function

Contrary to a linear regression where the residual sum of squares is used, in logistic regression, assuming that target $y_i \in \{-1, 1\}$, the weights w are fit using Maximum Likelihood Estimation. Then, the following optimization problem is solved:

$$\min_w \sum_{i=1}^n \log \left(1 + e^{-y_i f(w_0 + \sum_{j=1}^p w_j x_{ij})} \right)$$

by means of an algorithm such as coordinate descent, which successively minimizes along coordinate directions, updating one parameter at a time, or gradient descent for example, updating all parameters at once, to find the minimum of the function.

In order to be able to respond to a larger number of cases, logistic regression, as defined above for binary classification, has been extended to multiclass problems by using a One-vs-Rest (OvR) extension or a multinomial extension. While the first one transforms the classification problem into multiple binary problems by training

a separate model for each class and assuming that each classification problem is independent, the second one changes the loss function used by the algorithm into a cross-entropy loss which sums up the losses in each class and thus supports natively multi-class. To discourage learning a more complex or flexible model and prevents overfitting (the fact that the model is capturing the noise), the regularization process is also used. Then, it is possible to impose a ℓ_2 -penalty on the size of the coefficients in the optimization problem using the Ridge method. By preventing the weights from getting too large, the model is made less complex and has a lesser chance of overfitting. Then, the optimization problem becomes:

$$\min_w \sum_{i=1}^n \log \left(1 + e^{-y_i f(w_0 + \sum_{j=1}^p w_j x_{ij})} \right) + \frac{\lambda}{2} \sum_{j=1}^p w_j^2$$

where the complexity parameter $\lambda \geq 0$ controls the amount of shrinkage: the larger this parameter is, the greater the amount of shrinkage. An alternative is to impose a ℓ_1 -penalty using the Lasso method, which tends to prefer solutions with fewer non-zero coefficients by reducing the number of features:

$$\min_w \sum_{i=1}^n \log \left(1 + e^{-y_i f(w_0 + \sum_{j=1}^p w_j x_{ij})} \right) + \lambda \sum_{j=1}^p |w_j|$$

Finally, the Elastic-Net is a linear regression model trained with both ℓ_1 and ℓ_2 regularizations, whose weights depend on the $\rho \in [0, 1]$ coefficient:

$$\min_w \sum_{i=1}^n \log \left(1 + e^{-y_i f(w_0 + \sum_{j=1}^p w_j x_{ij})} \right) + \rho \sum_{j=1}^p |w_j| + \frac{1-\rho}{2} \sum_{j=1}^p w_j^2$$

4.1.2 Naive Bayesian

Based on the work of Bayes (1763), this algorithm assumes that all the features are completely independent of one another, given the target y . It is the reason why this algorithm is called "naive". For a classification case, the equation used is the Bayes' formula:

$$P(y|X) = \frac{P(y)P(X|y)}{P(X)}$$

where $P(A|B)$ is the conditional probability that event A occurs given that event B is true, with $P(B) \neq 0$. $P(y)$ and $P(X)$ are respectively the prior probability of the target and the prior probability of the predictors. $P(y|X)$ is the "posterior probability" of y and $P(X_k|y)$ for $k = 1, \dots, p$ is the "likelihood" of X_k .

Using the naive conditional independence assumption that $P(X_j|y, X_{-j}) = P(X_j|y)$, the relation can be simplified to:

$$P(y|X) = \frac{P(y) \prod_{j=0}^p P(X_j|y)}{P(X)}$$

Since $P(X)$ is constant given the input, the following classification rule can be used:

$$\hat{y} = \arg \max_y P(y) \prod_{j=0}^p P(X_j|y)$$

Then, $P(y)$ and $P(X_i|y)$ can be estimated using the *Maximum A Posterior* method, which consists in maximising the likelihood, weighted by the prior, that the model produced the data that were actually observed.

4.1.3 K-Nearest Neighbors (KNN)

First, developed by Fix and Hodges (1989), before being expanded by Altman (1992), this non-parametric model simply predicts the label of an observation using a vote between the k closest observed neighborhoods, with uniform weights or inversely proportional to the distance. It is then necessary to find an optimal k , as shown in Figure 4.2, usually using cross-validation, and a good distance metric, like the Euclidean distance, the cosine similarity or the Minkowski distance, for example.

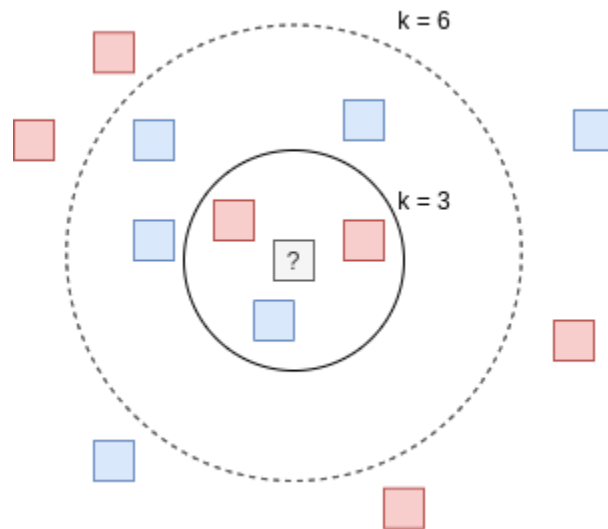


Figure 4.2: Impact of the number of neighbors in a binary classification example using a K-Nearest Neighbors with two features

4.1.4 Support Vector Machines (SVM)

Developed at *Bell labs* by Cortes and Vapnik (1995), *SVM* are a family of machine learning algorithms used to solve classification, regression or anomaly detection problems. Their goal is to separate the data into classes using an optimal hyperplane, so that the distance between the different groups of data and the boundaries that

separate them is maximal. The closest data to the border are the "support vectors" and the distance between the "support vectors" of different labels is known as "margin" (see Figure 4.3 for a visual interpretation).

Then, the maximization of the margin is an optimization problem that can be solved as follows:

$$\begin{aligned} & \min_{w, \zeta} \frac{1}{2} w^T w + C \sum_{i=1}^n \zeta_i \\ & \text{subject to } \begin{cases} y_i (w^T \phi(x_i)) \geq 1 - \zeta_i, \\ \zeta_i \geq 0 \end{cases} \end{aligned}$$

where $\mathbf{x}_i \in \mathbb{R}^p$ are the training vectors, $\mathbf{y} \in \{1, -1\}^n$ are the correct labels, ϕ is the identity function and the predictions are given by $\text{sign}(\mathbf{w}^T \phi(X))$. Since the hyperplane cannot separate perfectly within the correct classes, some samples are allowed to be at a distance ζ_i from the correct margin boundary and the penalty term C controls the strength of this penalty. This is then called a "soft margin".

Just like the *Logistic Regression*, the *SVM* is natively used for binary classification. But some extensions make it possible to use this algorithm for a multi-class problem, like the One-vs-Rest strategy presented previously or the One-vs-One approach, which fit $n_{classes} * (n_{classes} - 1) / 2$ classifiers.

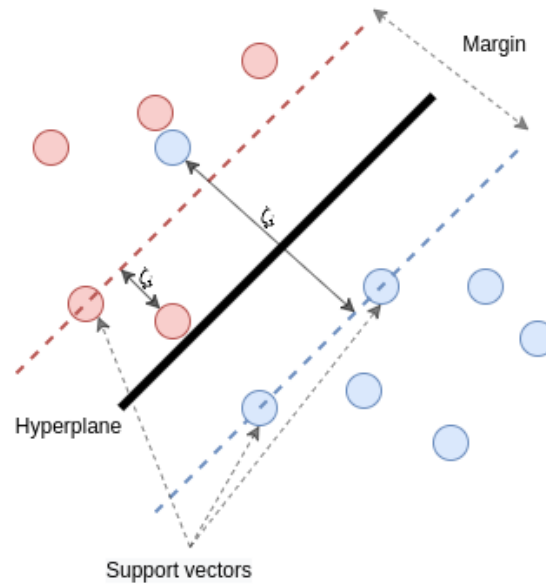


Figure 4.3: Binary classification example using Support Vector Machines with soft margin and two features

Finally, the *SVM* can be extended to non linear problems, using the kernel trick, which preprocesses the training data X by a map $K : \mathbb{X} \mapsto \mathbb{F}$ into a higher dimensional space \mathbb{F} . Then, several functions allow for the application of this technique, such as the following ones:

$$\text{Polynomial kernel: } K(\mathbf{x}, \mathbf{x}') = (\alpha \mathbf{x}^T \mathbf{x}' + \lambda)^d$$

$$\text{Gaussian kernel: } K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

$$\text{Laplacian kernel: } K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{\sigma}\right)$$

$$\text{Radial Basis Function (RBF) kernel: } K(\mathbf{x}, \mathbf{x}') = (\alpha \|\mathbf{x} - \mathbf{x}'\|^2 + \lambda)$$

4.1.5 Decision Tree

Even if it does not perform well in general, this non-parametric model can be very performant when used with stacking methods. Despite the numerous implementations it has undergone since its first use during the 1960's, it is relatively simple and easy to understand. Its functioning is relatively similar to that of the human mind: it attempts to split data into different parts, depending on the answers given to questions that are based on the available features.

Decision trees are built with two kinds of elements: nodes and branches. At each node, depending on the values taken by the feature, one of the features of our data is evaluated to either split the observations in the training process or to follow a certain path when making a prediction. Following this node, branches represent the possible outcomes or actions. We can then differentiate three kinds of nodes:

- the root node, which is the first node, which evaluates the variable that best splits the data.
- the intermediate nodes, which are the nodes where variables are evaluated but which are not the final nodes where predictions are made.
- the leaf nodes, which are the final nodes where the predictions are made.

First, the decision tree calculates the impurity of the dataset using a metric like the

Gini index or the Entropy, expressed below:

$$\text{Gini} = 1 - \sum_{i=1}^n p^2(c_i)$$

$$\text{Entropy} = \sum_{i=1}^n -p(c_i) \times \log_2(p(c_i))$$

where $p(c_i)$ is the probability of class c_i in a node.

Then, the dataset is split, and the impurity metric is calculated for each branch. This metric is added proportionally, to get a total impurity measure for the split. The algorithm keeps the best variable/threshold combination to build a child node and repeats the same process on every branch. If the impurity measure of a branch is greater than 0, then it needs further splitting, or else the branch is a leaf node. During its training process, a decision tree tries out different splits for each variable. For a discrete feature, all its possible values are evaluated, and for continuous features, the mean of each two consecutive values, ordered from lowest to highest, are used as possible thresholds (see Figure 4.4 for a visual interpretation).

To compare different splits, a decision tree uses the information gain, evaluated as follows:

$$IG(D_p, f) = I(D_p) - \frac{n_{left}}{n} I(D_{left}) - \frac{n_{right}}{n} I(D_{right})$$

where f is the feature concerned by the split, D_p is the parent node dataset and D_{left} and D_{right} are the child nodes dataset. I is the impurity criterion (Gini index or Entropy). n is the total number of samples, and n_{left} and n_{right} are the number of samples at child nodes.

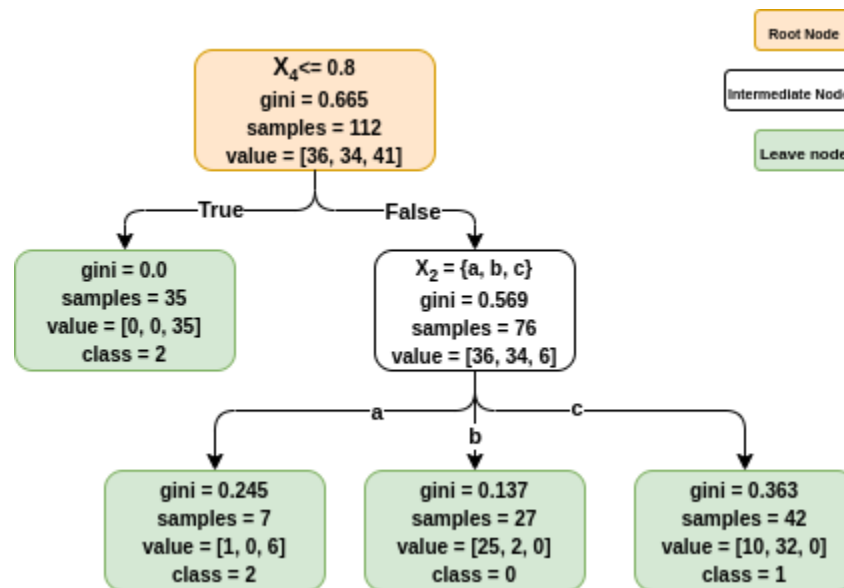


Figure 4.4: Multinomial classification example using a Decision Tree

4.1.6 Ensemble methods

The idea behind ensemble methods is to train a combination of $k > 1$ weak learners, using the same learning method (contrary to stacking, which trains learners with different learning techniques), to create a strong learner and obtain a better performance. Indeed, this combination of learners helps decrease variance, which indicates how much the model can adjust to the change in data, then control for over-fitting. It may also produce more reliable forecasts.

Bootstrap aggregating

In Bootstrap aggregating (or bagging) algorithms, the k learners are trained in parallel and each model is built independently using k training datasets, produced by random

sampling with replacement from the original observations. The prediction is obtained by averaging the responses of the k learners (or majority vote for classification). This procedure, visually detailed in Figure 4.5, reduces the variance of the prediction and may solve the over-fitting problem.

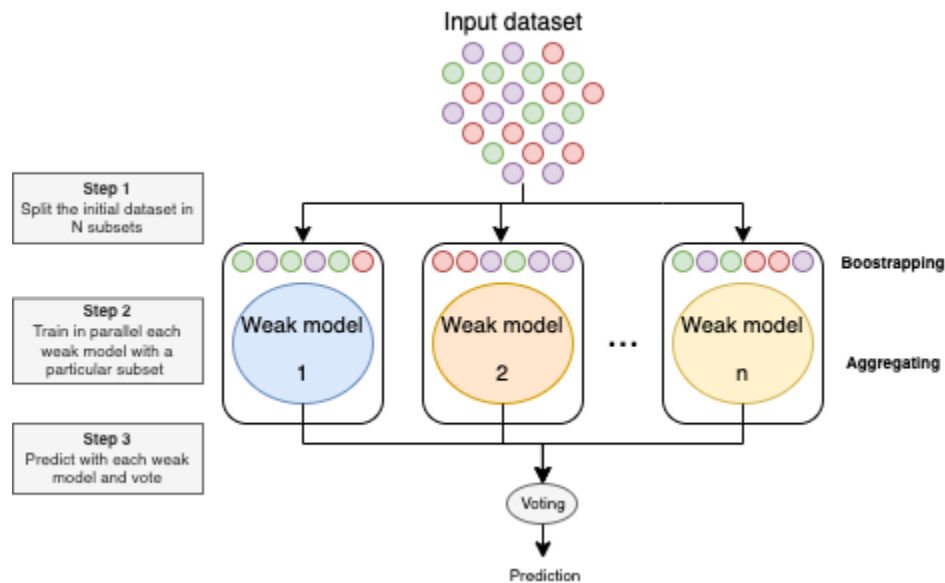


Figure 4.5: Bagging process details

Random Forest Inspired by the work of Ho (1995), the *Random Forest* algorithm has undergone many improvements before becoming the well-known version proposed by Breiman (2001). It differs from the classical bagging procedure by also selecting a random subset of the original set of features at each node of the decision trees used as weak learner. This "feature sampling" makes learning quicker and if one of the features is too strong a predictor for the target output, it avoids a strong correlation between trees.

Extremely Randomized Trees *Extremely Randomized Trees* or *Extra Trees* were invented by Geurts, Ernst, and Wehenkel (2006). It is simply a *Random Forest* in

which each tree is trained with all the samples (and not with a bootstrap sample) and the top-down splitting in the tree learner is randomized (and not based on an information gain).

Boosting

According to Hastie, Tibshirani, and Friedman (2009), *Boosting is one of the most powerful learning ideas introduced in the last twenty years*. In boosting algorithms, the k learners are trained sequentially, using the k training datasets, produced by random sampling with replacement in which some observations are overweighted. Indeed, each learner is trained on data that consider the previous learners' success. Then, after each training step, the weights are redistributed to increase the weights of mispredicted data and emphasize the most difficult cases. This way, subsequent learners will focus on them. Finally, the prediction is obtained by using a weighted average (or weighted vote for classification) based on the performance of each learner on the training data. Moreover, some of the boosting techniques include an extra-condition to keep or discard a learner depending on his performance. This procedure, visually detailed in Figure 4.6, can reduce the bias but increase the over-fit.

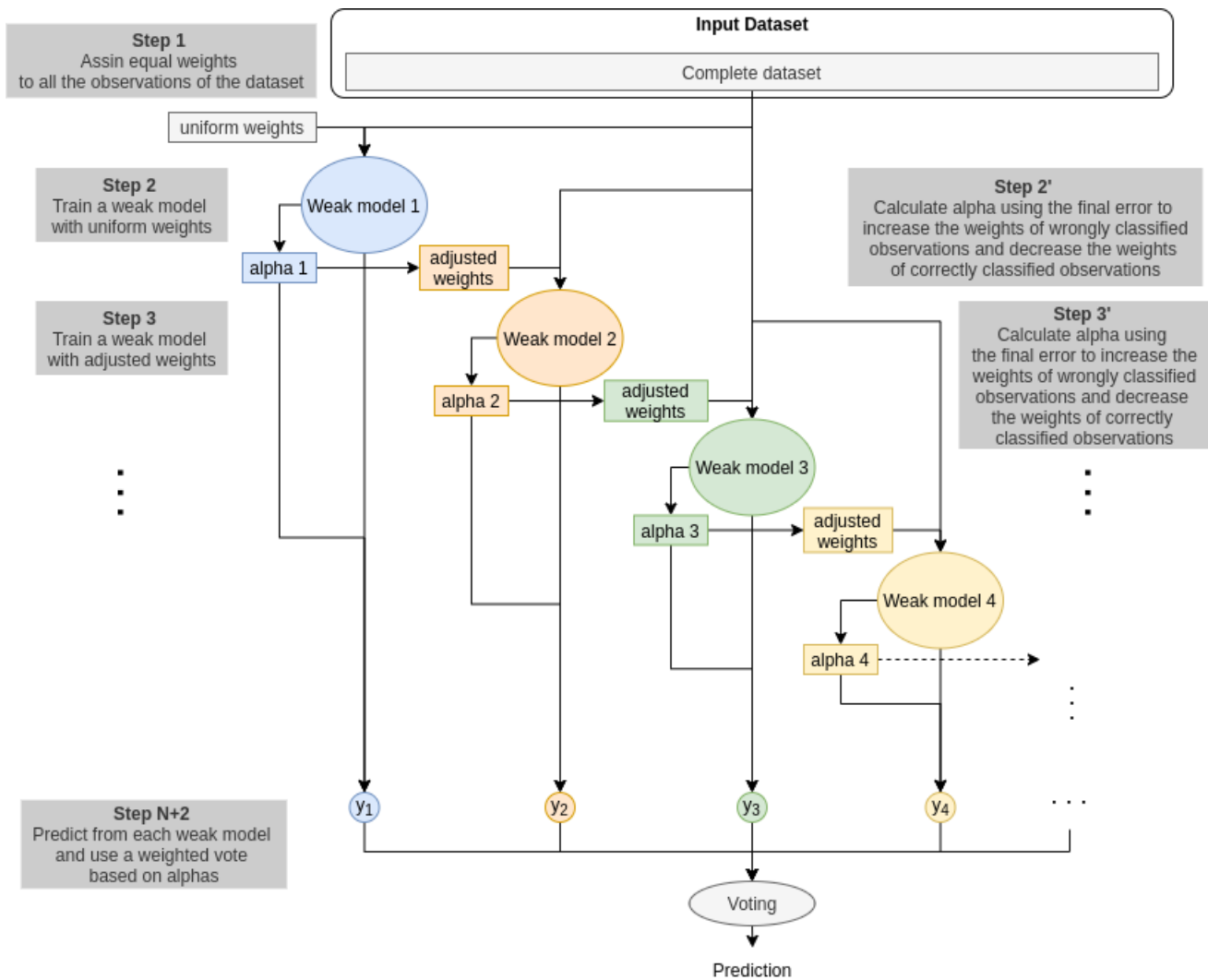


Figure 4.6: Boosting process details

Gradient Boosting Machines (GBM) Developed by Friedman (2001), *GBM* prefer using the gradients in the loss function, which can be specified, rather than using high weight data points, based on misprediction. By this specification of the cost function, this algorithm is very generic and can be adapted to various applications.

Moreover, contrary to other boosting algorithms, the prediction of gradient boosting is unweighted. Highly popular in Machine Learning competitions, numerous libraries provide efficient implementations of this algorithm, such as *XGBoost*, *LightGBM* or *CatBoost*.

4.1.7 Artificial Neural Networks (ANN)

ANN is the component of artificial intelligence that aims to simulate how a human brain functions: it uses interconnected neuron nodes like a web. Nodes, also called "processing units", are the location where computations happen. It combines input from the data with weights, that amplify or dampen this input, depending on the significance of inputs with regards to the learning task. These input-weight products are summed up, this sum is then passed through an activation function, to determine whether the signal should progress further through the network to affect the ultimate outcome and to what extent. Then, if the signal passes, the neuron has been "activated". The simplest kind of ANN is the single-layer perceptron. Inspired by earlier work by Warren McCulloch and Walter Pitts and developed in the 1950's and 1960's by the scientist Rosenblatt (1958), this algorithm has somewhat evolved since its first uses for binary classification, using only the step function as an activation function. Indeed, it is possible to build a more complex model, by connecting several layers, composed of several neurons. The equation for a neuron can be written as follows:

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

where a_j^l is the neuron output of the j^{th} neuron in the l^{th} layer, σ is the activation function (usually step, sigmoid, hyperbolic tangent or Rectified Linear Unit function), w_{jk}^l is the weight of the j^{th} neuron in the l^{th} layer over all neurons k in the $(l-1)^{\text{th}}$ layer, and b_j^l is the bias of the j^{th} neuron in the l^{th} layer.

There are several layer structures, depending on the learning task. According to Goodfellow, Bengio, and Courville (2016), *the modern feedforward network is the culmination of centuries of progress on the general function approximation task*. For our classification problem, this kind of structure is an appropriate layer organization. In this structure, the information moves forward from the input nodes through the hidden nodes and toward the output nodes. As shown in Figure 4.7, each neuron is fully connected to all neurons in the subsequent layer. When training this type of algorithm, the goal is to find weights and biases that minimize a loss function and compare forecasts of the model with observed targets. This can be done using two phases: forward-propagation and backward-propagation. As previously described, during the forward-propagation phase an output is produced after passing information through the entire network.

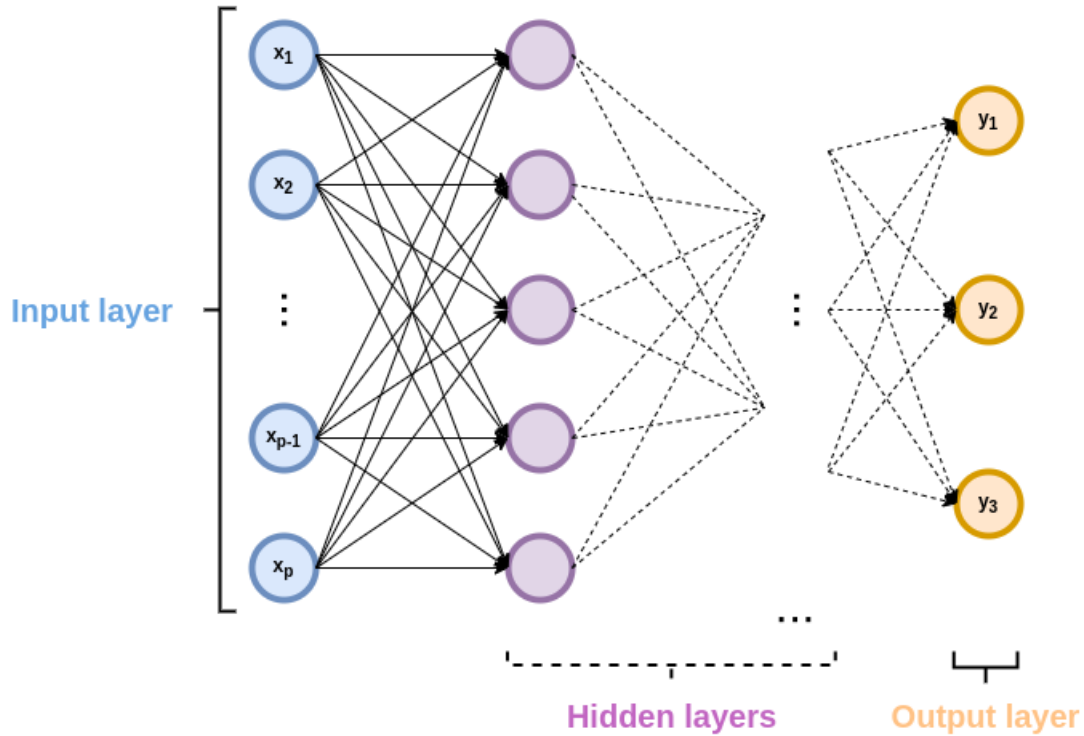


Figure 4.7: Feedforward Neural Networks structure details

During the backward-propagation, the loss associated with this backward-propagation step is propagated backward through the network, calculating an error gradient for each neuron in the hidden layer. These gradients are used to adjust neurons' weights to minimize the loss function, using methods like *gradient descent*. Then, each weight is updated as follows:

$$w_{jk}^l := w_{jk}^l - \alpha \frac{\partial C}{\partial w_{jk}^l}$$

where w_{jk}^l is the weight of the j^{th} neuron in the l^{th} layer, α is the learning rate, which controls the speed at which the model updates parameters, and $C(X, w)$ is the loss function. Some additional extensions, like the use of regularization or dropout

(deleting some hidden neurons) to avoid overfitting, can also be used to improve the way neural networks learn.

4.2 Evaluation metrics

According to Brownlee (2020), "*a classifier is only as good as the metric used to evaluate it*". Evaluating the predictive performance of models is certainly the most important thing in learning tasks. It makes a comparison of the models with all other possible and can be an indicator of the future performance of a model. Choosing a wrong metric can lead to a poor choice of model and mislead the expectations of the performance of the model. To best adapt the metric to its use cases, it is common to use sample weights to highlight the most interesting cases. Because the goal of this thesis is to produce a probabilistic forecast, only metrics using probabilities of labels rather than direct labels will be used. Except for the classification accuracy, which is easily interpretable and gives an idea of the efficiency of the model to non-experts.

4.2.1 Accuracy

It is simply the ratio of the number of correct predictions to the total number of samples:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

Useful when there are equal numbers of samples belonging to each class, this metric can become misleading when the cost of misclassification of the minor class samples is very high. This metric lies between 0 and 1, with higher scores being better.

4.2.2 Brier score

This loss function - for which a lower score indicates a more accurate model - is a Mean Squared Error (MSE) between the predicted probabilities and the expected values. Then, it summarizes the magnitude of the error in the probability forecasts, and the predictions, that are further away from the expected probability, are more penalized:

$$\text{Brier Score} = \frac{1}{N} \sum_{i=1}^N (y_{ij} - p_{ij})^2$$

where p_{ij} is the predicted probability for the observation i to be in class j and y_{ij} is the true outcome, which is equal to 1 if it is the true class and 0 otherwise. In case of imbalanced classes, it can be more convenient to prefer the Brier Skill Score, because it compares the Brier score with the "reference forecast", which refers to a naive prediction, like always predicting that home teams will win in soccer games:

$$\text{Brier Skill Score} = 1 - \frac{\text{Brier Score}}{\text{Brier Score}_{ref}}$$

4.2.3 Rank Probability Score (RPS)

Introduced by Epstein (1969), the *RPS* is identified by Murphy (1970) as "*a particularly appropriate scoring rule to evaluate probability forecasts of ordered variables*". Used by Constantinou and Fenton (2012) to evaluate their model to forecast soccer game outcomes, this metric is the multiclass version of the Brier score.

$$\text{RPS} = \frac{1}{K-1} \sum_{j=1}^K \left(\sum_{i=1}^j (p_i - y_i) \right)^2$$

where p_{ij} is the predicted probability for the observation i to be in class j and y_{ij} is the true outcome, which is equal to 1 if it is the true class and 0 otherwise.

This metric is then appropriate for soccer game outcomes for example.

4.2.4 Area Under Curve (AUC) of Receiver Operating Characteristic (ROC) curve

ROC is a graph showing the performance of the classification model at all the classification thresholds (the probability threshold, generally defined at 0.5, above which the classification model predicts the occurrence of the concerned class). This curve operates as a trade-off between the *True Positive Rate* (or *Sensitivity*), which is the ratio between the samples that are correctly identified as this label and the total number of samples that match this label, for each label and regardless of the classification, and the *False Positive Rate* (or $1 - \textit{Specificity}$), which is, for each label, the ratio between the samples that are not correctly identified as this label and the total number of samples that do not match this label, regardless of the classification. As shown in Figure 4.8, classifiers with curves closer to the top-left corner indicate a better performance (a maximal *TPR* for a minimal *FPR*) and random classifiers are expected to give a diagonal linear curve, indicating a *TPR* equals to the *FPR*). *AUC* summarizes the performance of the classifier associated to a *ROC*. It is a measure of a classifier's ability to distinguish classes, which can sometimes better score a classifier that underperforms in a specific region, but in practice, is a good measure of the predictive quality of a model.

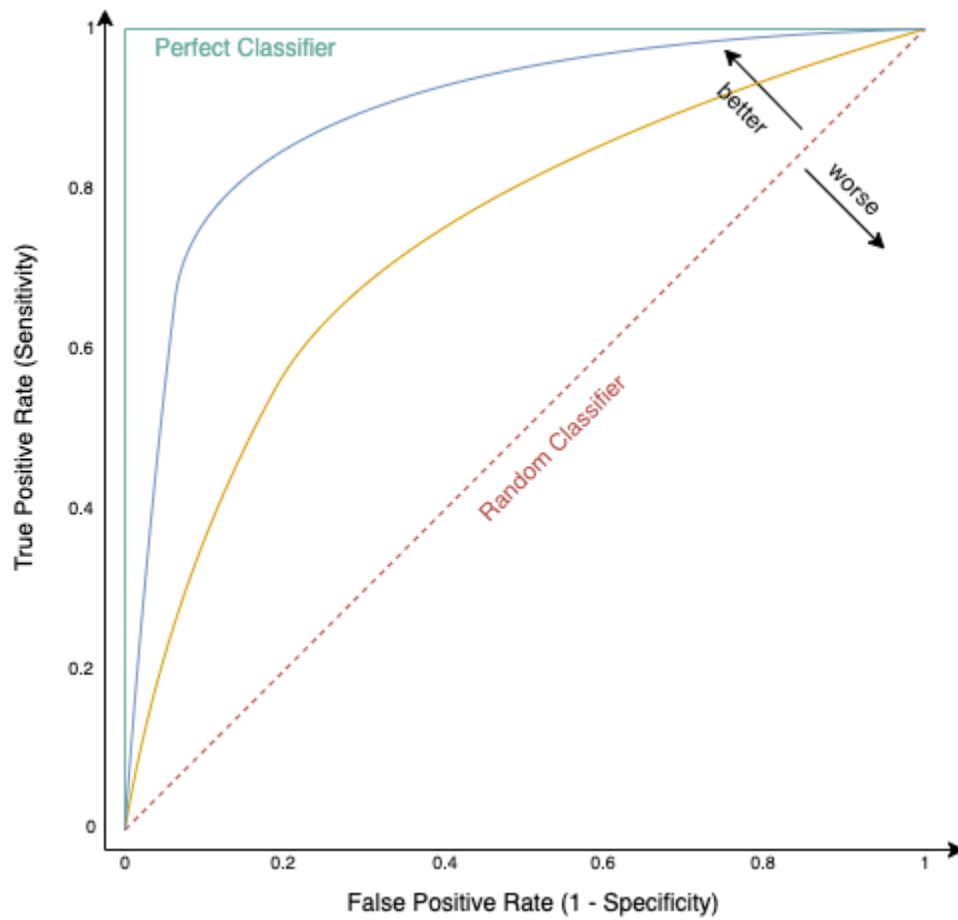


Figure 4.8: Receiver Operating Characteristic curve

4.2.5 Logarithmic loss (Cross Entropy)

This measure takes into account the uncertainty of a prediction penalizing, for each prediction, the farthest probabilities from the expected values:

$$\text{Log Loss} = -\frac{1}{N} \sum_{j=1}^K \sum_{i=1}^N y_{ij} \log(p_{ij})$$

where p_{ij} is the predicted probability for the observation i to be in class j and y_{ij} is the true outcome, which is equal to 1 if it is the true class and 0 otherwise. By penalizing more severely predictions that are further away from the expected probability, the Logarithmic loss is generally preferred to the Brier Score and is the most common classification metric.

4.3 Hyperparameter tuning methods

While the models' parameters are learned during the training phase, the hyperparameters, which control the learning process, are simply set when the model is initialized. Because these hyperparameters have a direct impact on how the model will perform, getting the best possible model means finding the most optimal set of hyperparameters. A Random Forest for instance, which is an ensemble model comprised of a collection of decision trees, will present severely different performances depending on how many decision trees will be used, or on what the maximum allowable depth for each decision tree will be. The learning rate, the number of layers and the neurons per layer, the activation functions or the number of epochs of neural networks are also hyperparameters. An hyperparameter space, that is explored by the most efficient way, has to be defined using an appropriate method. Then, models are evaluated on validation data, and the method selects the best one, which can be represented by the following equation:

$$x^* = \arg \min_{x \in \mathbb{X}} f(x)$$

where $f(x)$ is an objective score and x^* is the set of hyperparameters that minimizes this score.

4.3.1 Cross-Validation

In order to determine the validation subset used to evaluate the performance of the model during the hyperparameter search process, different validation techniques can be considered.

Hold-out method

The original sample is divided into two sub-samples, usually called the training set and the test set, respectively. Although the size of each sub-sample is arbitrary, the training subset is commonly larger than 60%. The model is then trained on the training subset and validated on the test sample.

k -fold method

Using this method, the original sample is divided into k sub-samples. $k-1$ sub-samples are then used to train the model and the remaining sub-sample is used as the validation subset. The operation is repeated to use each block as a validation subset. At the end of the procedure, k performance scores are obtained. The mean and the standard deviation of the k performance scores can then be calculated to estimate the bias and variance of the validation performance.

4.3.2 Navigating the Hyperparameter Space

Grid search

Arguably the most basic hyperparameter tuning method, *Grid search* evaluates the Cartesian product of a specified finite set of hyperparameter values, and selects the set which produces the best results, according to the chosen scoring method. Easily parallelized, this method becomes inefficient for high dimensionality hyperparameter spaces, since the number of evaluations exponentially increases as the number of hyperparameters grows: assuming a set of k parameters, and each of them has n distinct values, its computational complexity increases at a rate of $O(n^k)$.

Random search

This method differs from the *grid search* by randomly searching for hyperparameters instead of exhaustively. Each parameter setting is sampled from a distribution over possible parameter values. Compared to the *grid search*, it requires less time but is no guarantee of finding the optimal combination of hyperparameters. It is also easy to parallelize since each evaluation is independent. For large spaces however, as it reduces the probability of wasting time on a small poor-performing region of the hyperparameter space, it is in fact more efficient than the *grid search* method (see Figure 4.9 for a visual explanation). Since the number of total evaluations is set to a fixed value n before the optimization process starts, its computation complexity is $O(n)$.

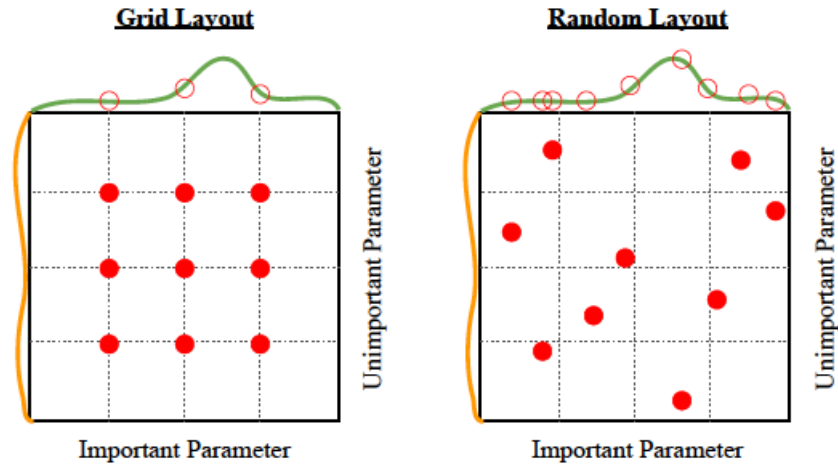


Figure 4.9: Grid & Random search comparison from *Introduction to Deep Learning* (Varma and Das 2018)

Tree-structured Parzen Estimator (TPE)

TPE is a sequential model-based optimization (SMBO). Unlike the two methods previously presented, it is possible to use the information from previous experiments to improve the next ones. The main idea is to build a probability model of the objective function, and to use it to select the most promising hyperparameters, by placing greater probability in regions where the true best hyperparameters lie. Then, the aim is to maximize the selection function, like the *Expected Improvement*, that is the criteria by which the next set of hyperparameters is chosen:

$$\text{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) \text{Pr}(y|x) dy$$

where \mathbf{x} are the hyperparameters, y is the objective function score, y^* is a threshold value of the objective function detailed below and $\text{Pr}(y|x)$ is the surrogate probability

model, also called the response surface, expressing the probability of y given x .

First, *TPE* tries to sample the response surface by random search. Then, it splits the observations in two groups: the best performing one and the other, defining a scoring value y^* as a threshold that splits the two groups. Then, it is possible to model the likelihood probability to be in each of these groups as:

$$\Pr(x|y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases}$$

By using these two different distributions of the hyperparameters and the Bayes rule $P(y|x) = P(x|y)P(y)/P(x)$, the *Expected Improvement* equation becomes:

$$\text{EI}_{y^*}(x) = \frac{\gamma y^* l(x) - l(x) \int_{-\infty}^{y^*} \Pr(y) dy}{\gamma l(x) + (1 - \gamma)g(x)} \alpha \left(\gamma + \frac{g(x)}{l(x)} (1 - \gamma) \right)^{-1}$$

Then, the *Expected Improvement* is proportional to the ratio $l(x)/g(x)$, and therefore, to maximize this ratio, it must prefer hyperparameters which are more likely under $l(x)$ than under $g(x)$. However, this criterion allows the model, detailed in Figure 4.10, to balance exploration versus exploitation.

TPE, in which time complexity is linearithmic ($O(n \log n)$), is slightly slower than the *random search* method, with a linear time complexity, but much better than *grid search*.

1. Test some hyperparameters & separate into:
 - best hyperparameters
 - bad hyperparameters
2. For new candidates (vertical lines), model probability to be in good or bad group
3. Expected Improvement for candidate:

$$P(\text{good}) / P(\text{bad})$$

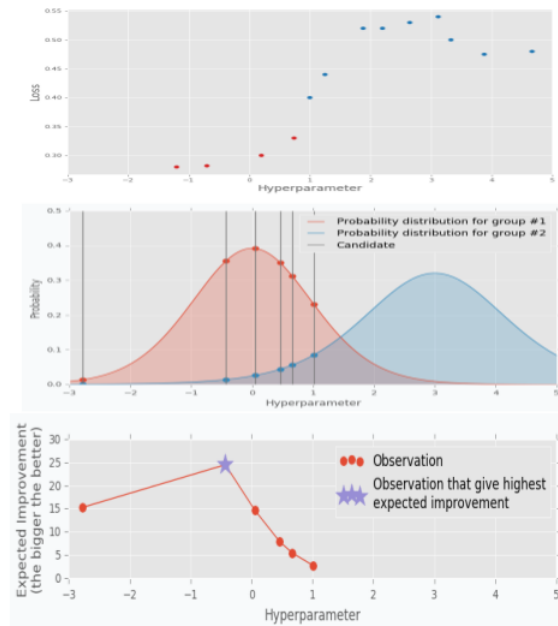


Figure 4.10: Tree-Structured Parzen Estimator details from NeuPy (Dr. Thorben Jensen)

4.4 Conclusion

Even if the efficiency of the classification algorithms previously presented is no longer to be proved, the means to optimize their performances by tuning hyperparameters is a good practice and the chosen evaluation metrics allow to have a good representation of their performances, the algorithm is not the only artisan of a performing model. And to reveal its full potential, a predictive analytics model needs to fully exploit the data at its disposal.

By applying domain knowledge to extract relevant information, one can transform the raw data into features that more accurately represent the problem underlying the predictive model and perform better.

5. Feature Engineering & Selection

Although the raw data scraped and stocked in the database seem to make sense for a human, the creation of meaningful features for the predictive models used requires a few selecting, transforming, and pre-processing stages. For any success in applied machine learning, the features are key: the better they are, the more flexible, simple, and efficient the model will be.

5.1 Feature Transformers

To change raw feature vectors into a representation that is more suitable for the downstream estimators, it is common to use scaling, normalization, or standardization methods for numerical features and encoding methods for categorical features, which have all been used with the implementation proposed by the “Scikit-learn” framework.

5.1.1 Numerical transformations

Because some Machine Learning algorithms are sensitive to feature scaling, it is common practice to rescale numerical features to use a common scale within the

dataset. For example, algorithms that use gradient descent as an optimization technique require the data to be scaled so that the gradient descent converges more quickly toward the minima. Algorithms based on distances like K-Nearest Neighbors (KNN) are also affected, because they use the distances between data points to determine their similarity: since both features have different scales, there is a chance that a higher weightage is given to features with a higher magnitude. This will impact the performance of the machine learning algorithm and obviously, we do not want our algorithm to be biased toward one feature.

Standardization Standardization is a scaling technique in which the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. Also known as Z-score normalization, the transformation is performed using the following formula:

$$X' = \frac{X - \mu}{\sigma}$$

where μ and σ are respectively the observed mean and the standard deviation of the feature.

Normalization Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. Also known as Min-Max scaling, the transformation is performed using the following formula:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where X_{min} and X_{max} are respectively the observed minimum and the maximum of the feature.

Other scaling methods can be considered for some issues, such as replacing the mean by the median and the standard deviation by the interquartile difference in the Standardization method, to obtain a more robust scaler for outliers.

5.1.2 Categorical transformations

Because many Machine Learning algorithms cannot directly operate on categorical data, transforming this type of variable into a numerical form is necessary. The two following methods are the most common, although many methods could be considered.

The One Hot Encoder This encoding method consists in representing a categorical variable with k categories in k binary variables in which the i^{th} binary variable represents the i^{th} category:

Colour	Red	Blue	Green
red	1	0	0
blue	0	1	0
green	0	0	1

Table 5.1: One Hot Encoding

Traditionally, this transformer is useful for features in which no ordinal relationship exists between categories.

The Ordinal Encoder This encoding method consists in assigning an integer value for each unique category of a categorical variable:

<u>Intensity</u>	<u>Intensity</u>
low	1
medium	2
high	3

Table 5.2: Ordinal Encoding

When the categories have an ordered relationship, this encoding method can be sufficient. Then, the encoded feature is used as a numerical one.

5.2 Feature Construction

Mainly driven by an understanding of both the domain and the problem at hand, for which proper definitions of the model's objectives and of the specific characteristics of the sport are necessary, the feature construction process requires an understanding of how the sport is played and what the factors which could potentially influence the model's target are. This was defined through personal knowledge of the sports, with help from the existing literature, and computed mainly using data manipulation frameworks like *Pandas* or *Dask*.

As shown within chapter 2, the data used for this study have several granularities:

- match-level data: common to both opponents involved in the match, it can be the importance of the match, the tournament or the league in which the match is played

for example.

- team-level data: concerning each opponent involved in the match, it can be the win/loss ratio since the beginning of the season for example. Usually, the different values of a specific feature are directly compared to get an idea of the influence on the target model.

- player-level data: concerning each player of a team for sports like soccer or basketball. It can be the market value of players for example. Usually, the different values of a specific feature are aggregated at a team level. For individual sports, like tennis, it is similar to the team-level data.

- event-level data: concerning every single match-event like a pass or a shot, this granularity is available only for soccer. They can be aggregated for each player or each team for example.

Because our main task deals with the outcome of the match, numerical data are aggregated at the team-level for soccer and basketball and at the player-level for tennis, before calculating their percentage difference.

Data can also be divided between match-related sources and external sources. External features do not concern events within the match and are known prior to the upcoming match. The league or tournament, the teams involved, or the distance each team must travel for the match are known. Match-related features, which concern the actual events within a match, are then not known until the match is played. Aggregating these features from previous matches is then necessary. For example, it is not possible to use the service points won by a tennis player during a match, but it is possible to use an average of this feature on different surfaces, since the beginning of the

tournament, or since the beginning of the career of the said player.

5.2.1 Common features

Dates and times (match-level data) are, potentially, rich sources of information. Some teams or players can have peaks of form at a specific period of the year or perform differently depending on the date and time of the match. Therefore, features related to the year, month, day, hour, or weekday are built from the datetime. Furthermore, two other features are computed:

- Timestamp: which is the number of seconds that have elapsed since the 1st January 1970

$$- KSP \text{ date} = Year + \frac{year \text{ day} - 0.5}{365 + leap \text{ year}}$$

where *leap year* is equal to 1 if it is a leap year and 0 else.

Then, opponents involved in the match (teams for soccer and basketball and players for tennis), and the league or the tournament in which the match is played (except for basketball, where only the NBA is considered) are used to create categorical features.

Skill rating features

A skill rating is a relative evaluation system of the ability of players and teams to win a match. Based on the results of previous matches, this type of metric is frequently used to rank the evaluated participants, like in tennis or football with respectively the ATP or the FIFA rankings.

Some papers have even been interested in creating new ranking systems, such as Ley, Christophe and Van de Wiele, Tom and Van Eetvelde, Hans (2019), who built a soccer-specific ranking system based on a Bivariate Poisson model to reflect the teams' current strengths. This ranking system also makes up for the shortcomings of the FIFA ranking system, such as too much variability, which can be caused by the choice of friendly matches for international teams, or the impossibility of using this system to propose a match result prediction.

Other rating systems have proved to be effective in the past however, with successful applications in the three considered sports. For the following rating systems, each participant is assigned points such that participants with the same number of points have the same strength. When a participant wins a match, points are gained proportionally to the difference in rating with the opponent (the increase is low if the winner was a favourite, high if the winner was an outsider) and similarly, points are lost by the defeated opponent (again in proportion to the difference in rating).

The Elo rating Developed by Arpad Elo, the Elo rating system was first used to evaluate the strengths of chess players in the 1960s. But this rating system has been successfully adopted in many use cases, like Hvattum and Arntzen (2010) that used a basic Elo rating and a goal-based extension for a soccer case, using the top four divisions of the English league system from 1993 to 2008. The Elo system is a zero-sum game in which the skill rating is a random variable that first follows a normal distribution, before preferring a logistic distribution, to allow for heavier distribution tails. Then, the participants' points are updated at the end of their

match as follows:

$$elo'_i = elo_i + K \times G \times (O - E)$$

where $E \in]0, 1[$ is the expected outcome and the observed outcome is $O \in \{0, 0.5, 1\}$ respectively corresponding to a loss, a draw or a win. The K -factor is a scaling parameter concerning the impact of more recent events. This parameter determines the ranking volatility, which can be increased for participants who have played fewer matches, provoking more important changes and a faster progression toward the real skill ability. The G parameter is also a scaling parameter, which does not exist in the original Elo rating system. It has been added to adapt to the specificities of the different use cases by varying the points exchanged during the match, depending on the importance of the match or the victory margin for example. The E parameter, representing the expected outcome, is defined by the following equation:

$$E = \frac{1}{1 + 10^{\frac{-dr_A}{400}}}$$

where $dr_A = elo_A - elo_B$ is the difference in points between two participants A and B . The closer E is to 1, the more the A participant will stand as favorite, and the closer E is to 0, the more the B participant will stand as favorite.

Then, E can be directly used to predict a probabilistic forecast of the outcome of a match. Inspired by Lasek (2016), who uses an ordered logit regression to estimate the expected outcomes of soccer matches, a constant has been added in the formulas used to calculate the expected outcomes and extend the use of Elo for soccer, including a

draw probability. Then, the expected outcomes are defined as follows:

$$P(A) = \frac{1}{1 + 10^{\frac{dr_A}{-c - \frac{dr_A}{400}}}}$$

$$P(B) = \frac{1}{1 + 10^{\frac{dr_A}{-c + \frac{dr_A}{400}}}}$$

$$\text{and } P(\text{draw}) = 1 - P(B) - P(A)$$

where $c \in \mathbb{R}_+$ is the constant parameter which determines the draw proportion.

In order to better fit with sport cases, some specificities are added:

- Home field advantage: because it is advantageous to play at home for soccer and basketball cases, a home advantage is manually incorporated by adding a constant value to the home team's Elo:

$$dr = elo_{home} - elo_{away} + HA_{[.]}$$

where $HA_{soccer} = 100$ and $HA_{basketball} = 87$

- Margin of victory: because a large victory can be the sign of a greater performance than a victory with a narrow score, a multiplier can be applied to the amount of exchanged Elo points. In this situation, the larger the score the team wins by, the more Elo points awarded to it at the end of the game are maximized:

- Soccer: According to *World Football Elo Ratings*, it can be useful to use the following multiplier:

$$G = \begin{cases} 1 & \text{if } MOV \in \{0, 1\}, \\ 1.5 & \text{if } MOV = 2, \\ \frac{11+MOV}{8} & \text{otherwise} \end{cases}$$

- Basketball: According to *Five Thirty Eight*, the following multiplier can be used:

$$G = \frac{(MOV + 3)^{0.8}}{7.5 + 0.006 \times elo\ diff}$$

- Year-to-Year Carry-Over: because from one season to the next a team's level can change (since players can be traded for example), a partial reset of the rating can be done at the beginning of each new season. According to *Five Thirty Eight*, a basketball team's Elo points can be updated as follows:

$$elo_{beginning\ new\ season} = 0.75 \times elo_{end\ previous\ season} + 0.25 \times elo_{start\ rating}$$

- Surface specificity: because some tennis players perform better on certain surfaces, it is possible to compute surface specific Elo points. Moreover, finding the mean between surface specific and general Elo points will generate the best trade-off.

The resulting models are the following:

- Tennis: $Elo(base_rating = 1500, K_factor = 32, Surface_specificity =$

True)

- Soccer: $Elo(base_rating = 1500, K_factor = 15, MOV = True, HA = 100, C = 0.25)$
- Basketball: $Elo(base_rating = 1500, K_factor = 20, YoY_adjustment = True, MOV = True, HA = 86)$

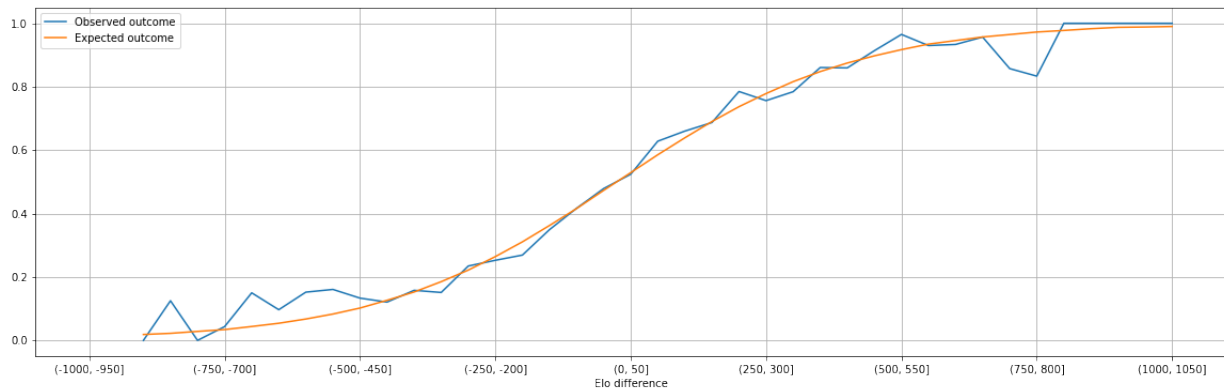


Figure 5.1: Win rate using observed and predicted Elo on tennis ATP from 2015 to 2018

The Glicko-2 rating Invented by Mark Glickman in 1995, this system is supposed to improve the Elo rating system by using a "rating reliability" (RD for rating deviation), measuring the accuracy of a participant's rating, with one RD being equal to one standard deviation.

In this rating system, every participant has a rating r , a rating deviation RD and a rating volatility σ . The volatility indicates the degree of expected fluctuation in a participant's rating. A higher value means that the participant has an erratic performance, and a low value means that the participant's performance is consistent.

The participant's strength can also be summarized in the form of an interval, with a 95% confidence interval ($r \pm RD$).

The TrueSkill rating Developed by Microsoft Research (see Herbrich, Minka, and Graepel (2007)), this skill-based ranking system is used for Xbox LIVE matchmaking service. This system quantifies players' True skill points using a Bayesian inference algorithm. Like the Glicko-2 rating system, this system is characterized by two numbers: the average skill of the participant (μ) and the degree of uncertainty in the participant's skill (σ) to define a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ representing the participant's rating. Then, the real skill of a participant is between $\mu \pm 2\sigma$ with 95% confidence.

Using the *Python* package developed by Lee (2018), the Gaussian distribution is initialized with $\mathcal{N}(25, (\frac{25}{3})^2)$. μ follows a participant's records for wins, draws or losses. A higher value means a higher game skill. σ follows the number of games, the lower the value the more games have been played and the higher the rating confidence is.

5.2.2 Tennis specific features

First, some features concerning the characteristics of a match are created. The importance of the match can be considered as an informative feature when used with players involved in a match, because these most certainly are not all equal when it comes to dealing with the pressure. Then, seven features are defined to reflect this importance:

-
- the tournament category and the ATP points: not every tournament holds the same interest and prestige. Moreover, the ATP points, that define the international ranking of each professional player, depend on the tournament category, as shown in Table 5.3. The tournament category is a categorical feature composed of 6 categories, the ATP points for the tournament winner and the ATP points involved in the match are two numerical features.
 - the round order and the match order: as the tournament progresses, the pressure for each game increases. Then, two ordinal features describe this effect.
 - single draw: the bigger a tournament is, the more rounds there will be and the more players will be involved. Then, an ordinal feature describes the number of players involved in a tournament.
 - the prize money: because it is usually proportional to the size and importance of the tournament, this feature is a good indicator. For example, since the 2010's, the winner of a Grand Slam tournament receives a pay-out of more than \$1,000,000, while the pay-out for the winner of an ATP 250 is usually less than \$100,000. This continuous feature, converted in dollars, depends on both the tournament and the season.

Tournament category	W	F	SF	QF	R16	R32	R64	R128	Q
Grand Slam	2000	1200	720	360	180	90	45	10	25
ATP Finals	+1100 (1 500 max)	+600 (1 000 max)	(200 for each round robin match win) (600 max)						
Masters 100	1000	600	360	180	90	45	10 (25)	(10)	25 (12)
500 Series	500	300	180	90	45	(20)			20 (10)
250 Series	250	150	90	45	20	(5)			12 (5)
Masters Next Gen									

Table 5.3: ATP point distribution since 2009

The fact that the surface is different depending on the tournament and that players have different preferences for different surfaces, a nominal feature with the following categories is used:

- grass: the ball moves at a faster pace with a lower bounce, favoring players with a good serve and good net playing skills. It is the fastest tennis court surface.
- hard: the bounce of the ball is high and predictable, favoring players with a good serve and base line players. This fast surface is the most commonly available in ATP tournaments.
- clay: this surface slows down the speed of the ball, reduces the skid and increases the bounce of the ball, favoring baseline players who use heavy spins.

Furthermore, a dummy variable indicates if the match is played outdoor or indoor.

Finally, because there were some missing values in the tournament category, a Ridge classifier is used to impute missing values based on the tournament name, prize money,

tournament surface and conditions (indoor or outdoor).

Some player-level features are also implemented, unrelated (detailed in Table 5.4) or related (detailed in Table 5.5) to match performance:

Feature	type	Details
ATP rank number	discrete	[1, 2 000]
ATP ranking points	discrete	[0, 16 950]
ATP move positions	discrete	[-1 999, +1 999]
Age	discrete	difference between birthdate and match date in number of days
Experience	discrete	difference between first apparition in ATP and match date in number of days
Height	continuous	in cm
Weight	continuous	in lbs
Body Mass Index	continuous	$\frac{0.45 \times Weight^2}{Height/100}$
Nationality	nominal	{fr, usa, ... , unknown}
Handedness	nominal	{right, left, ambidextrous, unknown}
Backhand	nominal	{one, two, unknown}
PlayerSeed	ordinal	{1, 2, ... , 16, 17+}
Walk Over Shift	dummy	1 if the player won his previous tournament match by withdrawal before the match, 0 else
Retirement Shift	dummy	1 if the player won his previous tournament match by withdrawal during the match, 0 else

Table 5.4: Non match-related ATP player-level features

Feature	type	Details
Aces	ratio	$\frac{\# \text{ aces}}{\# \text{ total serves}}$
Double faults	ratio	$\frac{\# \text{ double faults}}{\# \text{ total serves}}$
1 st serve	ratio	$\frac{\# \text{ 1}^{\text{st}} \text{ serves in}}{\# \text{ total serves}}$
1 st serve points won	ratio	$\frac{\# \text{ 1}^{\text{st}} \text{ serve points won}}{\# \text{ 1}^{\text{st}} \text{ serve points}}$
2 nd serve points won	ratio	$\frac{\# \text{ 2}^{\text{nd}} \text{ serve points won}}{\# \text{ 2}^{\text{nd}} \text{ serve points}}$
Break points saved	ratio	$\frac{\# \text{ break points saved}}{\# \text{ break points served}}$
1 st serve return points won	ratio	$\frac{\# \text{ 1}^{\text{st}} \text{ serve return points won}}{\# \text{ 1}^{\text{st}} \text{ serve return points}}$
2 nd serve return points won	ratio	$\frac{\# \text{ 2}^{\text{nd}} \text{ serve return points won}}{\# \text{ 2}^{\text{nd}} \text{ serve return points}}$
Break points converted	ratio	$\frac{\# \text{ break points converted}}{\# \text{ break points returned}}$
Service points won	ratio	$\frac{\# \text{ service points won}}{\# \text{ total serves}}$
Return points won	ratio	$\frac{\# \text{ return points won}}{\# \text{ total returns}}$
Total points won	ratio	$\frac{\# \text{ points won}}{\# \text{ total points}}$
Serve rating	continuous	$\% \text{ 1}^{\text{st}} \text{ serve} + \% \text{ 1}^{\text{st}} \text{ serve points won}$ $+ \% \text{ 2}^{\text{nd}} \text{ serve points won} + \% \text{ service games}$ won $+ \% \text{ aces} - \% \text{ double faults}$
Return rating	continuous	$\% \text{ 1}^{\text{st}} \text{ serve return points won}$ $+ \% \text{ 2}^{\text{nd}} \text{ serve return points won}$ $+ \% \text{ return games won} + \% \text{ break points}$ converted in minutes
Match duration	continuous	
Player sets, games & tie-breaks lost	discrete	
Win rate	continuous	
Win rate head-to-head	continuous	
Average rounds	continuous	

Table 5.5: Match-related ATP player-level features

Because the match duration feature has some missing values, an iterative imputation based on a Bayesian Ridge regressor using the total number of games in the match is used. Moreover, player rankings and player ATP points also have missing values for players with no international ranking. These observations are imputed with the worst ranking and the worst observed points (respectively 2 000 and 0). Then, the players' age, experience, height, and weight are imputed by their respective medians, and their nationality, handedness and backhand categories use an "unknown" category. Finally, the match-related features are aggregated using different horizons, detailed in Table 5.6:

Features	Aggregate horizons
Win rate	Season; Career; Career & Surface; Head-to-head; Head-to-head & Surface; 5 last head-to-head; 9 last matches
Average round	Season; Career; Career & Surface; Tourney Type
Serve & Return rating	Exponential Weighted Moving Average over career and career & surface
Match duration	Cumulative Sum over season and tournament
Sets; Games & Tiebreak lost	Cumulative Sum over tournament
Aces; Double faults; 1 st serve; 1 st serve points won; 2 nd serve points won; Break points saved; Break points converted; 1 st serve return points won; 2 nd serve return points won; Service; Return & Total points won	Career; Career & Surface; Season; Season & Surface; Tournament

Table 5.6: Aggregate horizons for match-related features

5.2.3 Basketball specific features

To determine the importance of an NBA match, a dummy feature determining if it is a playoff match, and a variable indicating the attendance for the match are used.

Then, for each team, a win-loss count from the beginning of the season and the beginning of the playoff period is used. A ratio of these features is also computed. Moreover, to identify possible fatigue for each team, a feature counting days off between each match and one calculating the distance traveled since the previous game are used. Another feature indicating the distance per day off is easily computed. For each match, the following statistics are observed:

- minutes played, field goals (include both 2-point field goals and 3-point field goals), field goal attempts (include both 2-point field goal attempts and 3-point field goal attempts), 3-point field goals, 3-point field goal attempts, free throws, free throw attempts, offensive rebounds, defensive rebounds, assists, steals, blocks, turnovers, personal fouls, points, plus/minus, box plus/minus, offensive rating and defensive rating.

Plus/Minus keeps track of the net changes in the score when a player is either on or off the court. Invented by Daniel Myers, Box Plus/Minus estimates player performance relative to the NBA average: since BPM is a per-100-possession stat where 0 is league average, +5 means the player is 5 points better than an average player over 100 possessions, -2 is replacement level, and -5 is really poor. Individual Offensive and Defensive Ratings are efficiency metrics developed by Oliver (2004). The offensive and defensive ratings are respectively the number of points produced and allowed by a player per hundred total individual possessions.

Finally, using APBRmetrics, such that the variables defined by Kubatko et al. (2007), the match-related features are also calculated, at team-level as in Table 5.7, or at player-level as in Table 5.8 (see Table A.1 for abbreviation details):

Features	Formulas
Points differential	$Tm Pts - Opp Pts$
Projected Winning	$(2.7 \times Points\ differential + 41) / 82$
Pythagorean Wins	$Tm Pts^{16.5} / (Tm Pts^{16.5} + Opp Pts^{16.5})$
Plays	$Tm FG / (Tm FGA - Tm ORB + Tm TOV)$
Offensive Rebounds (ORB)	$Tm ORB / (Tm ORB + Opp DRB)$
Defensive Rebounds (DRB)	$Tm DRB / (Tm DRB + Opp ORB)$
Total Rebounds	$Tm ORB + Tm DRB / 2$
Field Goals (FG)	$Tm FG / Tm FGA$
True Shootings	$Tm Pts / (2 \times (0.44 \times Tm FTA + Tm FGA))$
Possession	$0.4 \times \left[(Tm FGA + 0.4 \times Tm FTA - 1.07 \times \frac{Tm ORB}{Tm ORB + Opp DRB} \times (Tm FGA - Tm FG) + Tm TOV) + (Opp FGA + 0.4 \times Opp FTA - 1.07 \times \frac{Opp ORB}{Opp ORB + Tm DRB} (Opp FGA - Opp FG) + Opp TOV) \right]$
Plays	$0.44 \times Tm FTA + Tm FGA + Tm TO$
Offensive Efficiency	$(Tm Pts / Possession) \times 100$
Defensive Efficiency	$(Opp Pts / Possession) \times 100$
Efficiency Differential	$Offensive\ Efficiency - Defensive\ Efficiency$
Assist (AST) ratio	$(Tm AST / Possession) \times 100$
Turnover (TOV) ratio	$(Tm TOV / Poss) \times 100$
Defensive Rating	$(Opp Pts / Opp Poss) \times 100$
Value of Ball Possession	$(Tm Pts / Poss) \times 100$
Four Factors	$0.4 \times eFG\% + 0.1 \times TOV\% + 0.2 \times ORB\% + 0.15 \times (FT / FGA)$

Table 5.7: Formulas of basketball team-level features

Features	Formulas
Pts per Shot Attempt	$\frac{Pts}{FGA}$
Herfindal Index	$\left(\frac{Pts}{Tm Pts}\right)^2$
Touches	$FGA + TOV + \frac{FTA}{Tm FTA/Opp PF} + \frac{AST}{0.17}$
Pass %	$\frac{AST/0.17}{Touches} \times 100$
Shoot %	$\frac{FGA}{Touches} \times 100$
Fouled %	$\frac{FTA/(Tm FTA/Opp PF)}{Touches} \times 100$
TOV %	$\frac{TOV}{Touches} \times 100$
Personal Foul Efficiency	$\frac{STL + BLK}{PF}$
NBA Efficiency rating	$Pts + TRB + AST + STL + BLK - [(FGA - FG) + (FTA - FT) + TOV]$
Approximate Value	$\frac{NBA Efficiency rating^{0.75}}{21}$
Game Score	$[h]Pts + 0.4 \times FG - 0.7 \times FGA - 0.4 \times (FTA - FT) + 0.7 \times AST + 0.7 \times BLK - 0.4 \times PF - TOV$
Player Scoring Possession	$FG - 0.37 \times FG \times \frac{5 \times MP \times (Tm AST/Tm MP) - AST}{5 \times (Tm FG/Tm MP) - AST} + 0.37 \times AST + 0.5 \times FT$
Player Non Scoring Possession	$FGA - FG + 0.4 \times FTA + TOV$
Player Possession	$Player Scoring Possession + Player Non Scoring Possession$
Individual Floor %	$\frac{Player Scoring Possession}{Player Possession} \times 100$

Table 5.8: Formulas of basketball player-level features - 1/2

Features	Formulas
Tendex	$[h] \frac{1}{MP} \times [Pts + TRB + AST + STL + BLK - (FGA - FG) - 0.5 \times (FTA - FT) - TOV - PF]$
Win score	$Pts + TRB + STL + 0.5 \times (AST + BLK - FTA - PF) - FGA - TOV$
Unadjusted Player Efficiency Rating	$[h] \frac{1}{MP} \times \left[3Pts - \frac{PF \times Lg FT}{Lg PF} + 0.5 \times FT \times \left(2 - \frac{Tm AST}{3 \times Tm FG} \right) + 2/3 \times AST \right. \\ \left. + FG \times \left(2 - \frac{\left(2/3 - \frac{0.5 \times (Lg AST/Lg FG)}{2 \times (Lg FG/Lg FT)} \right) \times Tm AST}{Tm FG} \right) + Lg VBP \right. \\ \left. \times \left[Lg DRB\% \times [2 \times ORB + BLK - 0.2464 \times (FTA - FT) - (FGA - FG) - TRB] \right. \right. \\ \left. \left. + \frac{0.44 \times Lg FTA \times PF}{Lg PF} - (TOV + ORB) + STL + TRB - 0.1936 \times (FTA - FT) \right] \right]$
Player Efficiency Rating	$uPER \times \frac{Lg Pace}{Tm Pace} \times \frac{15}{Lg uPER}$
Player Impact Estimate	$[h] \frac{(Pts + FG + FT - FGA - FTA + DRB + ORB/2 + AST + STL + BLK/2 - PF - TOV)}{(Total Pts + Total FG + Total FT - Total FGA - Total FTA + Total DRB + Total ORB/2 + Total AST + Total STL + Total BLK/2 - Total PF - Total TOV)}$
Points Created	$[h] Pts + AST \times (2 - Lg VBP) + (TRB + STL + BLK) \times VBP - [(FGA - FG) + (FTA - FT) + TOV] \times Lg VBP - 0.5 \times Lg VBP \times PF$
Points Produced	$\frac{Offensive Rating \times (FGA + 0.44 \times FTA + TOV)}{100}$
Points Allowed	$\frac{DRTG}{100} \times Tm Possession \times \frac{0.2 \times MP}{Tm MP/5}$
Net Points	$Points Produced - Points Allowed$

Table 5.8: Formulas of basketball player-level features - 2/2

Because the features computed and observed above are match-related, they are aggregated using means and sums going back to the beginning of the player's career, the beginning of the season and the previous three, five and nine matches played. In addition, an exponential weighted moving average is used. Concerning player-level

features, they are also aggregated to team-level using the sum, mean, weighted mean (by minutes played), median, minimum, maximum and standard deviation.

5.2.4 Soccer specific features

Concerning match-level features, weather conditions are defined by an ordinal feature from 0 to 5 with no information on how the values are determined, and the attendance in the stadium uses a continuous feature. To complete the missing values of these two variables, a K-Nearest Neighbors is used, in which each missing value is imputed using the mean value from 15 nearest neighbors found in the training set. Features used to train this imputer are the team IDs, the division, the datetime, the attendance if the weather code is the target and inversely if the target is the attendance.

For team-related features, goals and goal differences are computed using match scores, and a Boolean feature determines whether the match is opposing two rival teams or not. A continuous feature, defined by league and by season, indicates UEFA points, which are based on points obtained by all clubs in a given season in the UEFA Champions League (UCL), UEFA Europa League (UEL) and UEFA Europa Conference League (UECL). Five ordinal features are built from the starting formation, which defines how many players are playing at each position line (defense, defensive midfield, midfield, offensive midfield, offense). Additionally, the team manager is used as a categorical feature and some information on the defensive and offensive tactics are also used as ordinal features. Their encodings are described in Table 5.9.

Features	Details	Encoding
Build up play speed	the speed in which attacks are put together	slow: 1 balanced: 2 fast: 3
Build up play dribbling	duel-oriented play	little: 1 normal: 2 lots: 3
Build up play passing	passing distance and support from teammates	short: 1 mixed: 2 long: 3
Build up play positioning	team's freedom of movement in the 1 st two third of the pitch	free: -1 organized: 1
Chance creation passing	amount of risk in pass decision and run support	safe: 1 normal: 2 risky: 3
Chance creation crossing	tendency / frequency of crosses into the box	little: 1 normal: 2 lots: 3
Chance creation shooting	tendency / frequency of shots taken	little: 1 normal: 2 lots: 3
Chance creation positioning	team's freedom of movement in the final third of the pitch	free: -1 organized: 1
Defence pressure	how high up the pitch the team will start pressuring	deep: 1 medium: 2 high: 3
Defence aggression	team's approach to tackling the ball possessor	contain: 1 double: 2 press: 3
Defence team width	how narrow or wide the team shape is set up when they does not have possession of the ball	wide: 1 normal: 2 narrow: 3
Defender line	shape and strategy of the defence	offside trap: -1 cover: 1

Table 5.9: Soccer tactic encoding

Also using video game data, fifteen discrete features, ranging from 0 to 100 (or 0 to 10 for a couple of features), and three continuous features are used:

- discrete
 - Overall, Attack, Midfield and Defensive ratings
 - Build up play speed, build up play speed, build up play dribbling and build up play passing
 - Chance creation passing, chance creation crossing and chance creation shooting
 - Defence pressure, defensive aggression and defensive team width
 - International and domestic prestige
- continuous
 - Transfer budget
 - Starting 11 and whole team average age

Finally, by using event data, it is possible to aggregate the following event-related features for each team at each match:

- Aerial challenges and Won Aerial challenges (with offensive and defensive specification): when 2 players challenge the possession of the ball in the air against each other. The player that wins the ball is deemed to have won the duel. When more than two players are involved, the closest player to the challenge winner is given an aerial challenge loss.
- Clearances: a defensive action where a player kicks the ball away from his own goal with no intended recipient.

-
- **Corners and accurate corners:** when the ball has left the field of play resulting in a corner. A won corner is collected for the team being awarded a corner, and a lost corner for the team that conceded a corner in favor of the opposing team. A taken corner is added when the player taking the corner has carried out the action, usually by doing a cross or a pass.
 - **Dispossessed:** when a player is in possession and not attempting to beat a tackler.
 - **Attempted and won dribbles:** an attempt by a player to beat an opponent when he has possession of the ball. A successful dribble means the player beats the defender while retaining possession, unsuccessful ones are when the dribbler is tackled.
 - **Errors:** when a player makes an error, which leads to a goal or a conceded shot. Also used for spills and attempted claims or saves by a goalkeeper which directly leads to a second attempt to score.
 - **Fouls committed:** any infringement that is penalised as foul play by a referee, except offsides.
 - **Interceptions:** when a player reads an opponent's pass and intercepts the ball by moving into the line of the intended pass.
 - **Caught offsides:** awarded to the player deemed to be in an offside position where a free kick is awarded. If two or more players are in an offside position when the pass is played, the player considered to be the most active and trying to play the ball is given offside.
 - **Total and accurate passes:** any intentional played ball from one player to

another. Passes include open play passes, goal kicks, corners and free kicks played as pass – but exclude crosses, keeper throws and throw-ins.

- Key passes: the final pass or pass-cum-shot leading to the recipient of the ball having an attempt at goal without scoring.
- Blocked shots: when any clear attempt to score that is going on target and is blocked by an outfield player, where there are other defenders or a goalkeeper behind the blocker.
- Shots off target: when any clear attempt to score misses the goal post without making contact with another player, or would have missed the goal post if not stopped by a goalkeeper's save or by an outfield player, or even if it directly hits the frame of the goal post and a goal is not scored.
- Shots on target: when any goal attempt that goes into the net regardless of intent or would have gone into the net but for being saved by the goalkeeper or is stopped by a player who is the last-man with the goalkeeper having no chance of preventing the goal.
- Total shots
- Missed tackles: when a player attempts to challenge for the ball and does not make it.
- Successful tackles: when a player connects with the ball in a ground challenge where he successfully takes the ball away from the player in possession.
- Total and accurate throw-in
- Touches: sum of all events where a player touches the ball, so excludes things like lost aerial or lost challenge.

- Deep completed passes: a pass that is targeted to the zone within 20 meters of the opponent goal.
- Offensive passes: a pass from the opposition pitch.
- Defensive actions: tackles, interceptions, challenges (failed tackles) or fouls within its pitch.
- Passes per defensive actions: the number of offensive passes allowed by the defending team by the total number of defensive actions.
- Key passes: a final pass or pass-cum-shot leading to the recipient of the ball having an attempt at goal without scoring.
- Assists: a final touch leading to the recipient of the ball scoring a goal. If the final touch (as defined in bold) is deflected by an opposition player, the initiator is only given a goal assist if the receiving player was likely to receive the ball without the deflection having taken place.
- Chances created: key passes plus assists.
- $Chances\ created+ = \frac{\#shots}{minutes\ played} \times 90 \times (1 + \frac{\#goals}{\#shots}) + \frac{\#key\ passes}{minutes\ played} \times 90 \times (1 + \frac{\#assists}{\#key\ passes})$

Then the mean, the sum, the weighted mean by minutes played by each player, the median, the minimum, the maximum and the standard deviation of those features are calculated for each team. To conclude, because those features are match-related they are also aggregated using expanding, rolling mean and sum, since the beginning of the available data, the beginning of the season and the previous three, five and nine matches. Furthermore, cumulative sums since the beginning of the available data and the season are used, as well as an exponential weighted moving average.

Expected Goals (xG)

Despite the debate over the origin of this metric, Macdonald (2012) was the first to use the term «expected goals», in a paper on ice hockey, before the concrete use of this concept by Lucey et al. (2015) with football data.

The expected goal is an estimate of the number of goals a team should score in a match. Each shot has an associated goal probability, based on the following variables:

- the distance: in general, the closer you get to the goal, the higher the xG score.
- the angle: in general, the sharper the shot angle, the lower the xG score.
- body part used for the shot: right foot, left foot or head.
- shooting situation: open play, free-kick or penalty.
- information pass type before the shot: location, angle, speed etc...

These event probabilities are then summed up to obtain the non-decreasing expected goal curves of each team during the match. This measure allows to evaluate the performance of players and teams. In a sport such as soccer, where the occurrence of goals is low, the final score does not always provide a true picture of the match and the performance of the teams. By statistically measuring the quality of goal opportunities created and conceded, it seems possible to have a more detailed analysis of the match and can be a good feature to predict the result of an upcoming match, as shown by Steffen, Gerville-Réache, and Bisoffi (2019).

Then, using the *Python* package created by Robberechts and Davis (2020), to train and analyze expected goal models in soccer, some xG models are trained using Whoscored events data. Because some pre-initialized models, with optimal hyperparameters, are

available with *Soccer xG*, it is not necessary to use a validation set. Then, the 2011 to 2016 seasons are used as a training set, and the next one as a test set, on which classification metrics are computed as shown in Table 5.10.

Features	Model	Brier score	AUC of ROC	residual area of ROC
Advanced	XGBoost	0.074	0.82	382
Advanced	Logistic Regression	0.076	0.8	561
Basic	XGBoost	0.077	0.79	488
Basic	Logistic Regression	0.078	0.78	670

Table 5.10: Soccer xG model results

in which basic features are:

- categorical: body part
- numerical: (x, y) location, distance to goal and shot angle

and advanced features add the following features:

- categorical: body part, type and success of the two previous passes before the shot
- numerical: (x, y) location, distance to goal and shot angle of the two previous passes before the shot

Then, the XGBoost model with advanced features is chosen to predict xG score for soccer actions.

The corresponding ROC and calibration curves, and the heatmap, highlighting score

probabilities depending on the pitch location, are depicted in Figure 5.2 and Figure 5.3.

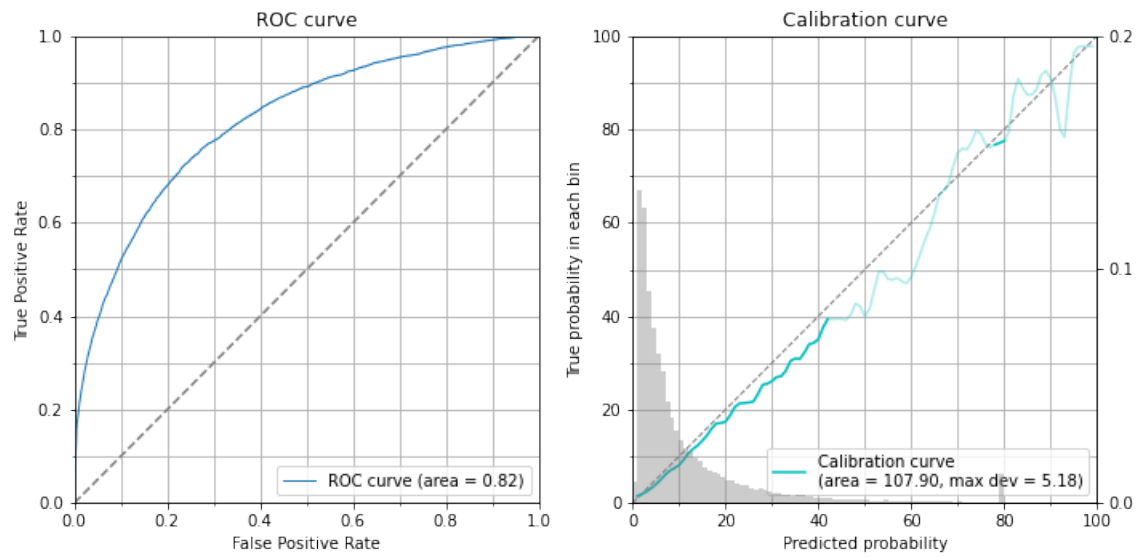


Figure 5.2: xG ROC and calibration curves

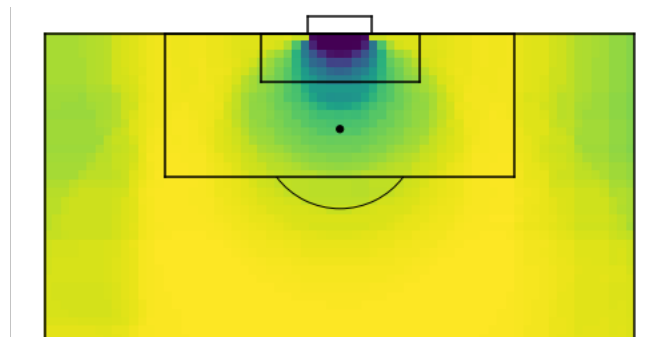


Figure 5.3: xG heatmap

Valuing on-the-ball actions

The xG previously presented only consider shots and goals, which are very rare events, representing only 2% of all events during a match, and are rarely practiced by some players like the defenders. So other metrics make it possible to assign to each action performed by a soccer player a value that reflects the usefulness of this action to win the game. But two reasons make this difficult for actions that are not shots:

- Just considering the direct effect of an action is not a good idea, because actions can produce longer lasting effects: a player should not just be rewarded for moving the ball into a good shooting position, but also for moving into threatening positions which can lead to a high probability of the ball reaching an even better position for shooting.
- Even knowing the longer lasting effects, knowing how to assign credit to each action in a sequence is not obvious. It would seem undesirable to penalize a great pass that does not lead to a shot.

The idea is to consider a soccer match as a sequence of n consecutive on-the-ball actions $[a_1, a_2, \dots, a_n]$ (e.g., $[dribble, pass, \dots, interception]$), and to assign a numeric value to each of these actions. Rather than directly assigning values to actions, the idea is to assign values to game states, and express the action usefulness as the difference between the post-action game state and the pre-action game state:

$$U(a_i) = V(S_i) - V(S_{i-1})$$

where V captures the value of a particular game state, $S_{i-1} = \{a_1, \dots, a_{i-1}\}$ is the pre-action game state, and $S_i = \{a_1, \dots, a_{i-1}, a_i\}$ is the post-action game state.

As shown by Van Roy et al. (2020), xT and $VAEP$ approaches can be complementary candidates to value actions in this way.

Expected Threat (xT) The expected threat or xT model is a possession-based model. It divides matches into possessions, which are periods of the game where the same team has the control of the ball. The main idea is that players perform actions with the intention to increase their team's chance of scoring and the chance of scoring can be captured by only considering the location of the ball.

Then, as shown in Figure 5.4, the pitch is divided into a $M \times N$ grid, in which each zone z is assigned a value $xT(z)$ that reflects how threatening teams are at that location, in terms of scoring.

The value of each zone is learned with a Markov decision process, initializing all zones at zero, and using the following formula:

$$\mathbf{xT}_{x,y} = (s_{x,y} \times g_{x,y}) + (m_{x,y} \times \sum_{z=1}^M \sum_{w=1}^N T_{(x,y) \rightarrow (z,w)} \mathbf{xT}_{z,w})$$

where (x, y) is a location on the grid, $s_{x,y}$ the percentage of attempting a shot in this location, $m_{x,y}$ is their percentage of moving the ball, and $g_{x,y}$ is the probability of scoring in this location. $T_{(x,y) \rightarrow (z,w)}$ is the move transition matrix, expressing the probability to move from (x, y) to (z, w) .

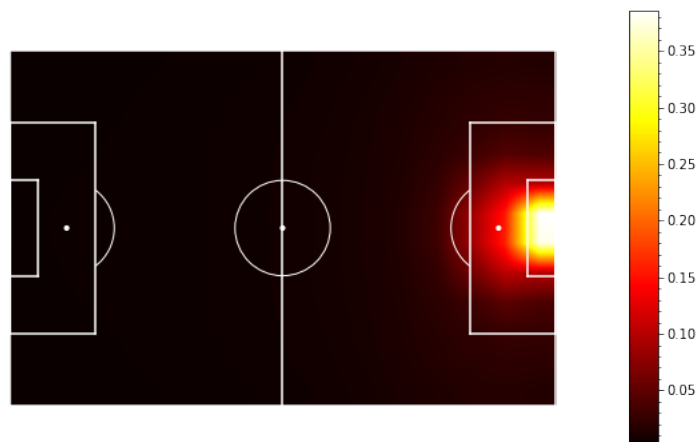


Figure 5.4: 16×22 xT interpolated heatmap

Valuing Actions by Estimating Probabilities (VAEP) Slightly different, the VAEP, introduced by Decroos et al. (2019), supposes that each player tends to perform an action with the intention to increase their team’s chance of scoring a goal in the near future or to decrease their team’s chance of conceding a goal in the near future:

$$V(S_i) = P_{score}(S_i, t) - P_{concede}(S_i, t),$$

where $P_{score}(S_i, t)$ and $P_{concede}(S_i, t)$ are the probabilities that team t which possesses the ball in state S_i will respectively score or concede in the next 10 actions.

To estimate $P_{score}(S_i, t)$ and $P_{concede}(S_i, t)$, a gradient boosted binary classifier is trained on historical data to predict how a game state will turn out based on what

happened in similar game states that arose in previous games. Then, each game state is represented with characteristics of the action itself such as its location and type, or the distance and the angle to the goal, the context of the action, such as the current tempo of the game, using features like distance covered and time elapsed between consecutive actions. The current game context is also used, by looking at the remaining time in the match and the current score differential.

Then, $P_{score}(S_i, t)$ has a positive label if the team that possesses the ball after the action a_i scores in the subsequent k actions, and $P_{concede}(S_i, t)$ has a positive label if the team that possesses the ball after action a_i concedes a goal in the subsequent k actions. Then two models are independently trained, and the change in scoring and conceding probabilities can be expressed as follows:

$$\begin{aligned}\Delta P_{score}(a_i, t) &= P_{score}^k(S_i, t) - P_{score}^k(S_{i-1}, t) \\ \Delta P_{concede}(a_i, t) &= P_{concede}^k(S_i, t) - P_{concede}^k(S_{i-1}, t)\end{aligned}$$

Finally, the total VAEP can be expressed using the following formula:

$$V_{VAEP}(a_i) = \Delta P_{score}(a_i, t) - \Delta P_{concede}(a_i, t)$$

Then, the example of an action is visually described, in order to identify the impact of the characteristics of an action on the probability of scoring or taking a goal in the next actions, and the resulting VAPE, with the Figure 5.5.

As other match-related features, xG, xT and VAEP are aggregated at team level, and on previous matches, with the same operations previously expressed.

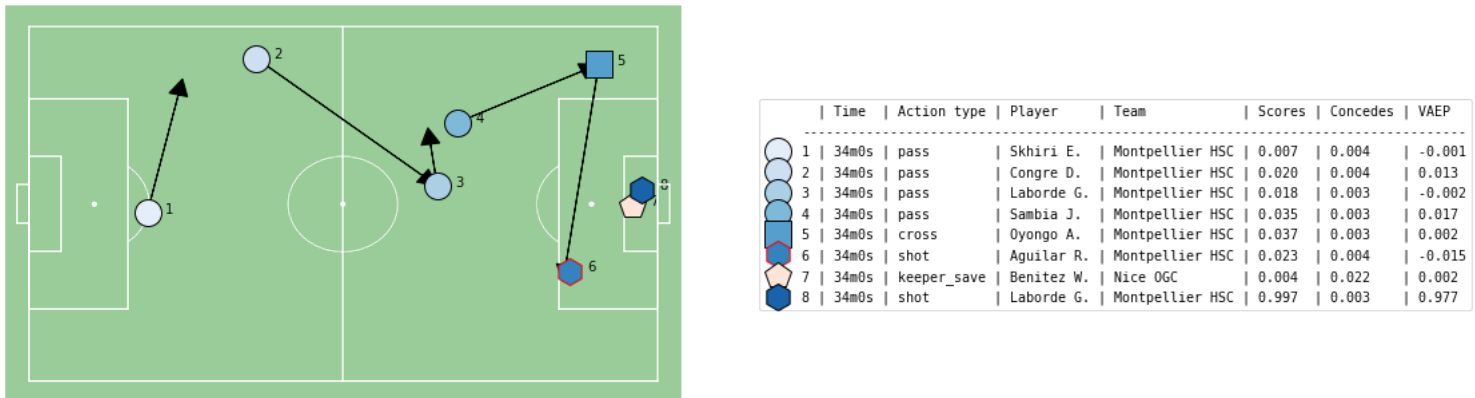


Figure 5.5: VAEP of victory goal of MHSC-OGCN during the 2017/2018 season

5.3 Feature Selection

Unfortunately, the treatments outlined above tend to create a large number of features, not all of which are necessarily useful. Selecting a subset of relevant features can make training faster (or feasible in some cases), can increase the performance of our models by removing irrelevant/non-informative features, which would introduce unnecessary noise, and make the model simpler, therefore easier to understand and to explain. To this end, three types of feature selection can be used.

Filter based

Faster and less computationally expensive, these methods, whose process is detailed in Figure 5.6, select features based on statistics, independently from the machine learning algorithm model. Then, only predictors with important relationships with the target variable would be included in the model.

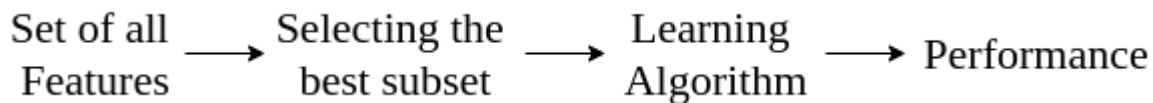


Figure 5.6: Filter method design

Following this, a correlation measure can be used, to quantify the relationships between the features and the target variable. The xi correlation metric, developed by Chatterjee (2020), is used, because it is as simple as the classical measures of statistical association (like Pearson's correlation coefficient, Spearman's ρ or Kendall's τ), it is a consistent estimator of some measure of dependence which is 0 if and only if the variables are independent and 1 if and only if one is a measurable function of the other, and it has a simple asymptotic theory under the hypothesis of independence, like the classical coefficients. Moreover, it can also detect associations that are not monotonic and be used with one-hot encoded categorical variables.

Such a coefficient is presented below:

For (X, Y) a pair of random variables, where Y is not a constant, data are arranged as $(X_{(1)}, Y_{(1)}), \dots, (X_{(n)}, Y_{(n)})$ such that $X_{(1)} \leq \dots \leq X_{(n)}$ and supposing that X_i 's have no ties:

$$\xi_n(X, Y) := 1 - \frac{3 \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{n^2 - 1}$$

where r_i is the rank of $Y_{(i)}$, that is the number of j such that $Y_{(j)} \leq Y_{(i)}$.

In the case of ties among the X_i 's, an increasing rearrangement by uniformly breaking ties at random is used:

$$\xi_n(X, Y) := 1 - \frac{n \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{2 \sum_{i=1}^n l_i (n - l_i)}$$

where l_i is the number of j such that $Y_{(j)} \geq Y_{(i)}$ and there are no ties among Y_i 's.

Unfortunately, this kind of method looks at individual features to identify its relative importance, feature dependencies are then ignored.

Using the x_i correlation metric, 62, 20 and 66 features were respectively selected for tennis, basketball and soccer (see Table A.2, Table A.3, Table A.4).

Wrapper based

Wrapper methods, whose process is detailed in Figure 5.7, select features using a machine learning algorithm that is fitted on a given subset. Then, it measures the "usefulness" of features, by comparing the algorithm performance with different combinations of features.

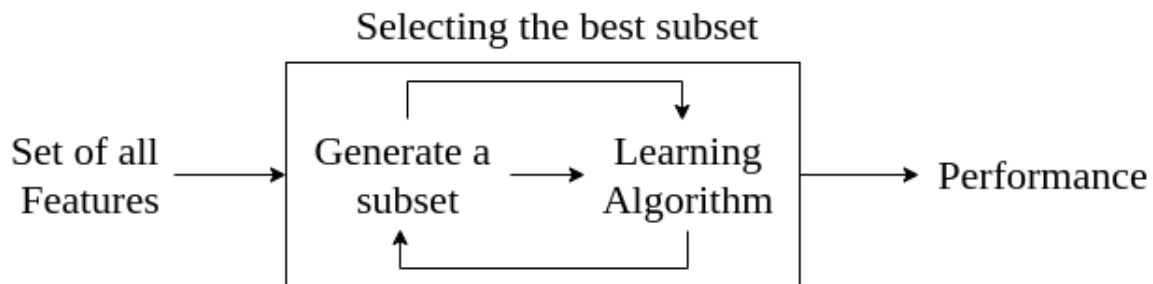


Figure 5.7: Wrapper method design

Initially implemented on the statistical programming language R by Kursa and Rudnicki (2010), *Boruta* is an all-relevant feature selection algorithm, which is very useful when we have no idea about which one is important in relation to our target variable. Contrary to many other popular wrapper methods, it is not necessary to

define a feature importance threshold at which the variables are selected, the *Boruta* algorithm does it all by itself.

The idea behind *Boruta* is very simple. First, the training dataset is duplicated, and the values of each column are randomly shuffled to obtain permuted features, also called shadow features. Then, a machine learning estimator, such as a tree ensemble method (like *Random Forest*) which can capture non-linear relationships between predictors, is trained with the original and shadow features. The feature importance describes which features are relevant by averaging over all the trees in the forest how each feature decreases the impurity of the splits of each original feature. It is then compared to the highest importance recorded among the shadow features. If the feature importance is higher than this threshold, the feature is flagged as a "hit". With this process, features are not compared between themselves, but with a randomized version of themselves, and are defined as "useful" if it can do better than the best randomized feature.

This process, visually detailed in Figure 5.8, is repeated n times, with a new random shuffle, and new shadow features for each trial. Then, a "hit" vector is obtained by feature.

Because each trial can give a binary outcome (hit or not hit) for each feature, the experiment follows a binomial distribution, shown in Figure 5.9, with parameters n trials and probability 0.5, defining the maximum level of uncertainty about the feature. Then, using the probability mass function of this binomial distribution, 3 areas can be detected:

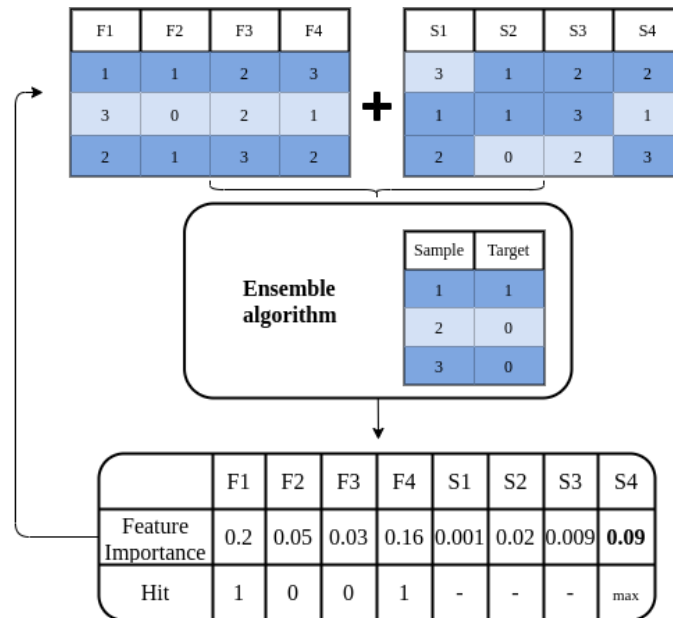


Figure 5.8: Boruta process

- a refusal area, considered as noise and dropped, with several hits located at the lowest tail of the probability mass function.
- an irresolution area, indecisive features, with a number of hits located at the centre of the probability mass function.
- an acceptance area, considered as predictive and kept, with a number of hits located at the highest tail of the probability mass function.

These tails are determined by an α parameter, generally fixed at 5%, which determines the part of the distribution considered as tail.

Even if this kind of method is computationally costly and can be time-consuming, it considers feature dependencies. Unfortunately, since it is a based machine learning model there is a risk of over fitting.

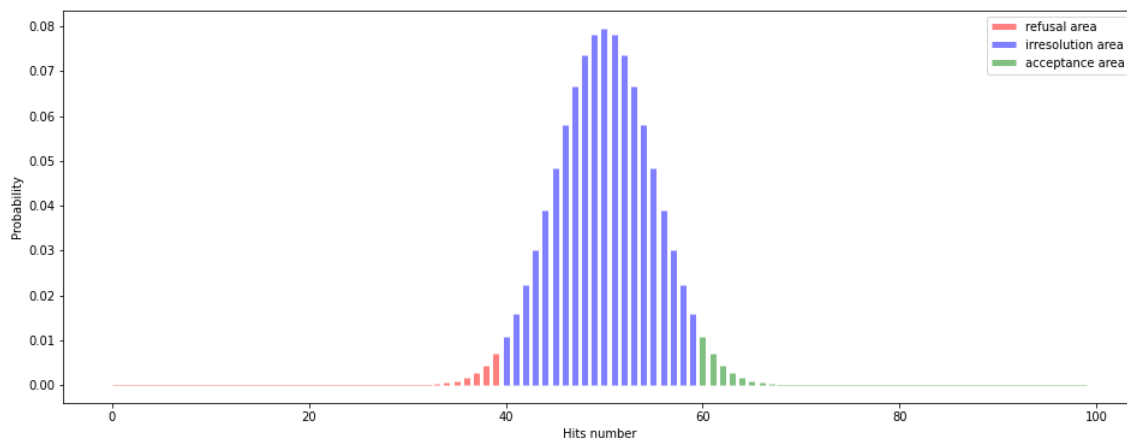


Figure 5.9: Probability Mass Function of a Binomial distribution with $n = 100$, $p = 0.5$ and decision areas with $\alpha = 0.005$

Using this algorithm, 29, 85, 94 features are respectively selected for tennis, basketball and soccer cases (see Figure A.1, Figure A.2, Figure A.3).

Embedded

Embedded methods, whose process is detailed in Figure 5.10, combine the qualities of both filter and wrapper methods, by integrating the feature selection as part of the learning algorithm. Thus, it is possible to use regularization techniques, like the *Lasso*, the *Ridge* or the *Elastic Nets* methods, with *Logistic Regression*, presented in chapter 4, or an algorithm approach, like the tree-based algorithm does, by using only some features to split the data.

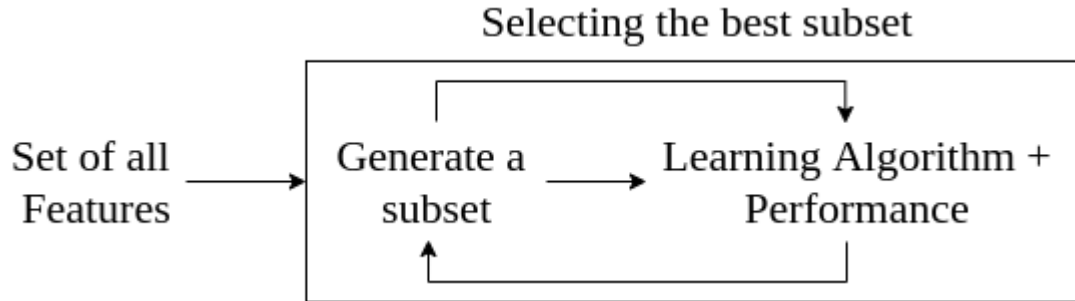


Figure 5.10: Embedded method design

5.4 Conclusion

Finally, mainly using *Pandas*, a *Python* package that provides a fast, flexible and powerful open source data analysis / manipulation tool, “Scikit-learn”, a *Python* module for machine learning, and *Dask*, a flexible parallel computing library to manage large volumes like soccer event data, 130, 2125 and 120 features are respectively built for tennis, basketball and soccer, before categorical encoding.

Because many classification algorithms need numerical inputs, this number of features can reach 574, 5240 and 1901 for tennis, basketball and soccer respectively, after a one-hot encoding of categorical features.

This preprocessing takes about 1.5 hours for tennis, 1:45 hours for basketball and 3.5 hours for soccer, using 12 CPUs and 32 GBs of memory locally for tennis and basketball, and an *AWS* instance of 96 CPUs and 384 GBs for soccer, due to the large volume of data generated by the event data.

6. Experiments & Results

There are many ways of responding to our classification problems that can be considered. Although some of the processing steps are not challenged, such as feature building, it is possible to compare the effectiveness of the transformations of some features like the scaling of numerical features, or the encoding of categorical features, the selected subset of features, the classifier or even the hyperparameters used to initialize it. Through multiple experiments using the different ways previously described, it is possible to determine the best model to use to maximize the performance metric of interest.

6.1 Benchmark

Theoretically, a sport odd should be fixed as the inverse of the probability that the event E , concerned by the bet, occurs:

$$Fair\ Odd_E = \frac{1}{P(E)} \quad \forall E \in \Omega$$

where Ω is the set of all possible outcomes and $P(\Omega) = 1$

In order to generate revenues, bookmakers apply a commercial margin on their odds:

$$\text{Commercial Odd}_E = \text{Fair Odd}_E - \text{Commercial Margin}$$

In addition to the recovered data detailed in chapter 2, some observed odds have been downloaded from Tennis-Data.co.uk, Sports Book Reviews Online and Football-Data.co.uk. As previously expressed, these odds, from *Bet365*, the main bookmaker in the United Kingdom, for soccer and tennis, and from a Nevada land-based bookmaker, for basketball, include a commercial margin shown in Figure 6.1.

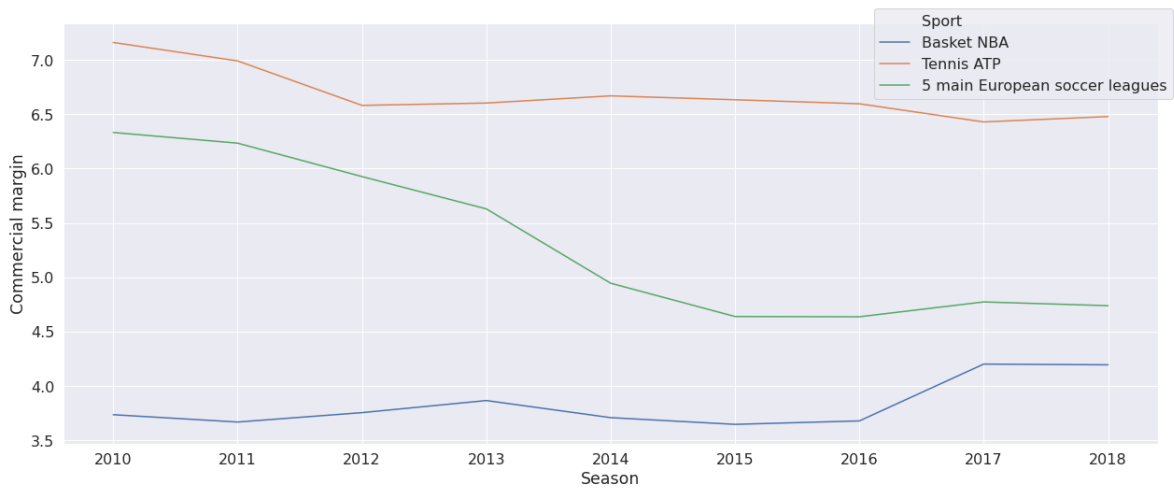


Figure 6.1: Bookmakers' commercial margins

Using the assumption that this margin is evenly distributed among all the selections of a bet, or the events X of the set of all the possible outcomes Ω , the following approximation is used:

$$P_{bookmaker}(E) \approx \frac{1/Commercial\ Odd_E}{\sum_{X \in \Omega} 1/Commercial\ Odd_X}$$

where $\sum_{X \in \Omega} P_{bookmaker}(E) = P(\Omega) = 1$

Then, evaluation metrics is computed, on a test set, using these probabilities. A benchmark of existing bookmakers is built and can be compared to the forecast of the probabilistic model.

Basketball

Concerning basketball, the evaluation metrics calculation is released depending on whether or not the match takes place during the playoffs.

Season	Playoffs	n	Log Loss	Brier	ROC AUC	Accuracy
2017	0	1230	0.592	0.407	0.734	0.686
2017	1	82	0.570	0.386	0.688	0.732
2017	-	1312	0.591	0.406	0.733	0.689
2018	0	1230	0.593	0.407	0.731	0.672
2018	1	82	0.630	0.432	0.689	0.634
2018	-	1312	0.595	0.409	0.729	0.669

Table 6.1: Bookmaker metrics on NBA

The probabilities induced by the bookmakers' odds seem to be rather constant, whether the match takes place in the playoffs or not, and over time.

Tennis

Regarding tennis, the evaluation metric calculation is released according to the main match characteristics, i.e. the surface, the quality of the tournament and the progress within the tournament.

Season	Surface	Series	Round	n	Log Loss	Brier	ROC AUC	Accuracy
2017	Clay	-	-	803	0.596	0.411	0.742	0.665
2017	Grass	-	-	323	0.571	0.39	0.77	0.696
2017	Hard	-	-	1500	0.585	0.402	0.753	0.679
2017	-	ATP 250	-	1104	0.621	0.432	0.711	0.645
2017	-	ATP 500	-	434	0.588	0.404	0.751	0.672
2017	-	Masters 1000	-	566	0.603	0.416	0.734	0.665
2017	-	Masters Cup	-	15	0.655	0.448	0.687	0.666
2017	-	Grand Slam	-	508	0.491	0.322	0.843	0.761
2017	-	-	Round Robin	12	0.572	0.396	0.736	0.666
2017	-	-	1st Round	1173	0.593	0.409	0.742	0.670
2017	-	-	2nd Round	752	0.575	0.391	0.769	0.692
2017	-	-	3rd Round	176	0.587	0.404	0.752	0.664
2017	-	-	4th Round	48	0.506	0.328	0.834	0.729
2017	-	-	Quarterfinals	264	0.606	0.421	0.727	0.666
2017	-	-	Semifinals	134	0.588	0.404	0.75	0.671
2017	-	-	Final	67	0.59	0.403	0.757	0.656
2017	-	-	-	2626	0.587	0.403	0.752	0.677
2018	Clay	-	-	808	0.599	0.415	0.735	0.678
2018	Grass	-	-	322	0.597	0.412	0.74	0.658
2018	Hard	-	-	1494	0.589	0.405	0.751	0.689
2018	-	ATP 250	-	1105	0.635	0.446	0.687	0.630
2018	-	ATP 500	-	431	0.592	0.406	0.752	0.719
2018	-	Masters 1000	-	566	0.595	0.409	0.743	0.683
2018	-	Masters Cup	-	15	0.65	0.457	0.758	0.666
2018	-	Grand Slam	-	507	0.499	0.328	0.84	0.761
2018	-	-	Round Robin	12	0.552	0.372	0.861	0.75
2018	-	-	1st Round	1176	0.589	0.405	0.751	0.691
2018	-	-	2nd Round	747	0.605	0.42	0.729	0.665
2018	-	-	3rd Round	176	0.518	0.346	0.83	0.744
2018	-	-	4th Round	48	0.501	0.318	0.831	0.812
2018	-	-	Quarterfinals	264	0.627	0.436	0.707	0.655
2018	-	-	Semifinals	134	0.588	0.406	0.741	0.664
2018	-	-	Final	67	0.673	0.48	0.644	0.582
2018	-	-	-	2624	0.593	0.409	0.744	0.682

Table 6.2: Bookmaker metrics on ATP

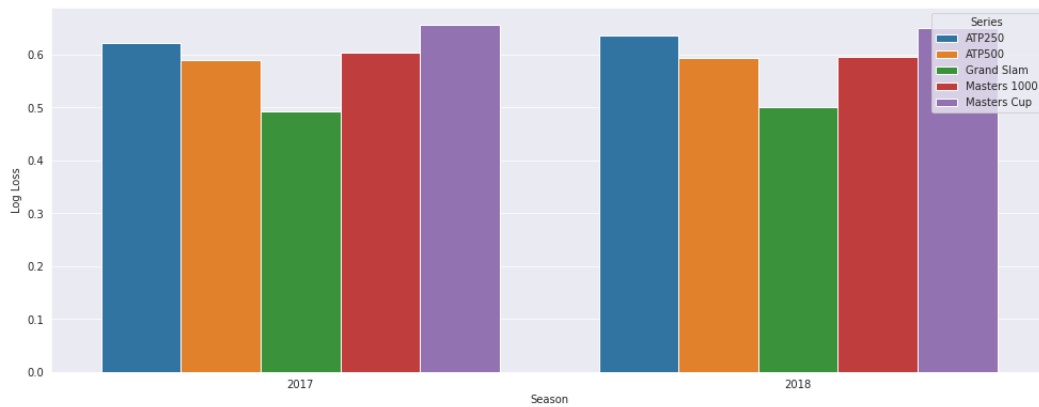


Figure 6.2: Evaluation of probabilities induced by *Bet365* tennis odds

Therefore, we notice almost similar performances, depending on the surface on which the match takes place, with a very slightly lower quality for clay matches.

Soccer

Finally, for soccer, performance metrics are split depending on the championship (see Table 6.3).

As shown with Figure 6.3, the probabilities induced by the bookmakers' odds reveal better performances on Italian and English matches.

Season	Championship	n	Log Loss	RPS	ROC AUC	Accuracy
2017	Bundesliga	306	0.998	0.596	0.651	0.503
2017	Premier League	380	0.945	0.560	0.704	0.547
2017	Ligue 1	380	0.953	0.566	0.685	0.547
2017	Serie A	380	0.892	0.522	0.736	0.597
2017	La Liga	380	0.960	0.571	0.657	0.550
2017	-	1826	0.947	0.562	0.690	0.551
2018	Bundesliga	306	0.963	0.571	0.674	0.536
2018	Premier League	380	0.893	0.522	0.714	0.584
2018	Ligue 1	380	1.005	0.603	0.652	0.482
2018	Serie A	380	0.945	0.559	0.721	0.550
2018	La Liga	380	1.010	0.603	0.652	0.487
2018	-	1826	0.963	0.571	0.688	0.527

Table 6.3: Bookmaker metrics on 5 main European soccer leagues

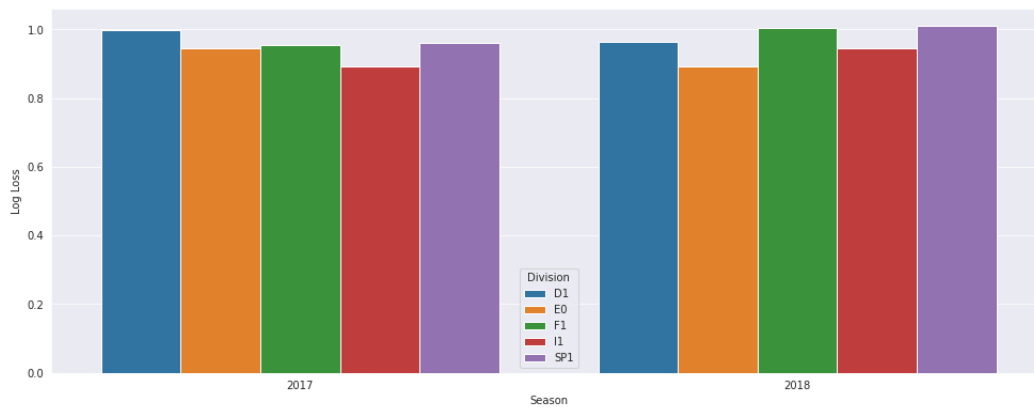


Figure 6.3: Evaluation of probabilities induced by *Bet365* soccer odds

6.2 Data split

Overfitting is the fact of producing a model that fits too closely or exactly into the set of data used during the training of the model, running the risk of not fitting with additional future observations. Thus, it is the noise, or the irrelevant information or randomness, in the training data that are learned, to minimize the error of this set, and not the signal, or the true underlying patterns of data, which are more general, and more useful to generalize this learning. To detect this issue, the dataset is split in a training set, to train and tune the model, and a test set, to evaluate the performance. By comparing the performance of those two sets, models with a satisfactory goodness of fit can be identified: it will be those which had the best bias-variance tradeoff. On the other hand, overfitting models, which do much better on the training set than the testing set, are discarded.

To prevent this overfitting problem, the cross-validation method can be used. By partitioning the training dataset into multiple mini train-test splits, models can be tuned, like defining hyperparameters, and overfitting issues can be avoided by keeping the test set as a truly unseen dataset to select the final model. Then, a variation of the k -fold method, described in section 4.3.1, is used. Detailed in Figure 6.4, this method respects temporality of our data: in the k^{th} split, the k oldest folds, are used as train set and the $(k + 1)^{\text{th}}$ fold, as test set.

For this content, the test set, used to evaluate the performance of the selected model and get an idea of its performance in production usage, is fixed as the two last seasons

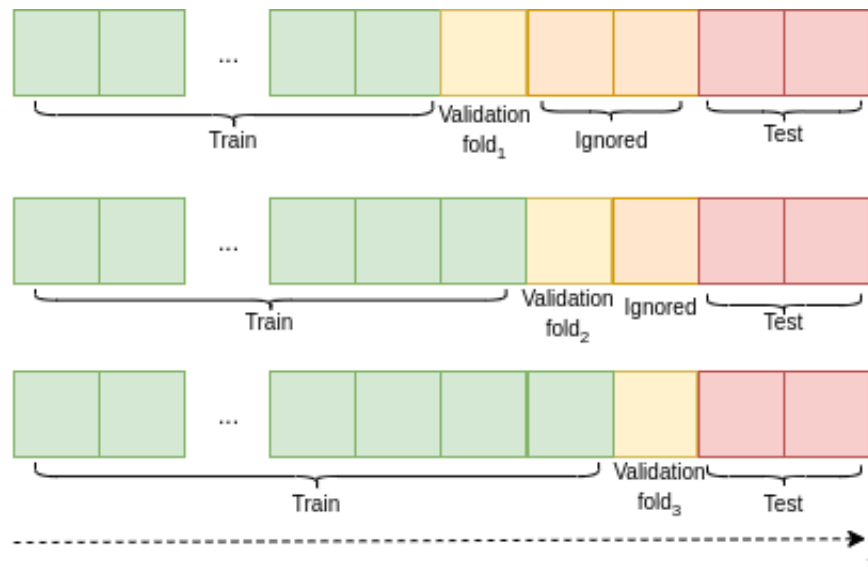


Figure 6.4: Temporal 3-fold cross-validation split

(2017 and 2018). In addition, the 2015 and 2016 seasons will be used as validation set, with a temporal 4-fold cross validation split.

6.3 Hyperparameter tuning

In order to identify the best model to predict the event probabilities, the algorithms detailed in chapter 3, and the data obtained after the preprocessing presented in chapter 5 and split following section 6.2’s process, have been used. Moreover, the hyperparameter search algorithms are used with “Optuna”, an open source hyperparameter optimization framework to automate hyperparameter search, for models implemented with “Scikit-learn”, “XGBoost” or *CatBoost*. For multi-layer perceptron, “Auto-Keras”, an efficient neural architecture search system based on Keras is used. To go even further, numerical features are standardized, except for

tree-based algorithms, for which it is useless, and categorical features are one-hot encoded.

Logistic regression is fitted with and without feature selection, with ℓ_1 , ℓ_2 and elastic net regularizations, with different solvers to resolve the optimization problem and using a logarithmic scale to explore different inverses of the regularization strength, ranging from 0.0001 to 100, during a grid search.

As can be seen in table 6.4, the best results are provided with an ℓ_2 regularization using a Logistic regression, to reach a log loss of 0.569. This experiment, also conducted on basketball and soccer data, resulted in a log loss of 0.595 and 0.969, using the *saga* resolver and the *elasticnet* regularization, with an inverse of regularization strength of 0.001 and an ℓ_1 ratio of 0.3 for the basketball case, and with the top 66 features according to *xicor* coefficient on the soccer one.

Selection method / Regularization	Solver	Inverse of regularization strength	L1 ratio	Mean time (in sec)	Std time	Mean Log Loss	Std Log Loss
-	lbfgs	-	-	8	0.34	0.5723	0.009
-	sag	-	-	22.65	1.54	0.5721	0.0086
-	saga	-	-	22.2	1.94	0.5707	0.0089
$\text{xicor}_{top\ 65}$	saga	-	-	12.35	0.65	0.5739	0.0083
l1	liblinear	10	-	54.3	3.43	0.5751	0.0081
l1	liblinear	0.1	-	17.46	3.97	0.5693	0.0093
l1	liblinear	0.01	-	6.35	0.53	0.5708	0.0083
l1	saga	1	-	62.52	1.77	0.5702	0.009
l1	saga	0.1	-	38.11	1.26	0.5692	0.0093
l1	saga	0.01	-	14.05	1.07	0.5708	0.0083
l2	liblinear	0.1	-	34.39	2.04	0.5708	0.0087
l2	liblinear	0.01	-	17.58	1.39	0.5689	0.0089
l2	liblinear	0.001	-	7.63	0.89	0.5706	0.0080
l2	lbfgs	0.1	-	11.02	0.53	0.5705	0.0089
l2	lbfgs	0.01	-	10.00	0.69	0.5689	0.0089
l2	lbfgs	0.001	-	6.74	0.21	0.5706	0.008
l2	saga	1	-	36.69	1.32	0.5706	0.0089
l2	saga	0.01	-	25.88	0.84	0.5689	0.0089
l2	saga	0.001	-	5.92	0.73	0.5706	0.008
elasticnet	saga	1	0.9	60.32	3.28	0.5702	0.0090
elasticnet	saga	1	0.5	61.97	2.34	0.5704	0.0090
elasticnet	saga	1	0.3	62.93	2.38	0.5705	0.0089
elasticnet	saga	0.1	0.9	47.53	2.25	0.5692	0.0093
elasticnet	saga	0.1	0.7	49.41	2.38	0.5691	0.0093
elasticnet	saga	0.1	0.1	62.14	3.20	0.5697	0.009
elasticnet	saga	0.01	0.7	19.73	0.58	0.5702	0.0084
elasticnet	saga	0.01	0.1	42.36	1.52	0.5691	0.009
elasticnet	saga	0.001	0.9	4.83	0.14	0.5789	0.0069
elasticnet	saga	0.001	0.7	5.73	0.58	0.5773	0.007
elasticnet	saga	0.001	0.1	8.47	0.43	0.5722	0.0077

Table 6.4: Partial Logistic Regression grid-search results on soccer data

Still using an exploration of the hyperparameter space with the help of the grid search algorithm, K-Nearest Neighbors is fitted with a number of neighbors ranging from 1 to 300, using a linear scale with an increment of 4, with Euclidean, Manhattan and Minkowski distance metrics and with weights either depending on the distance or not.

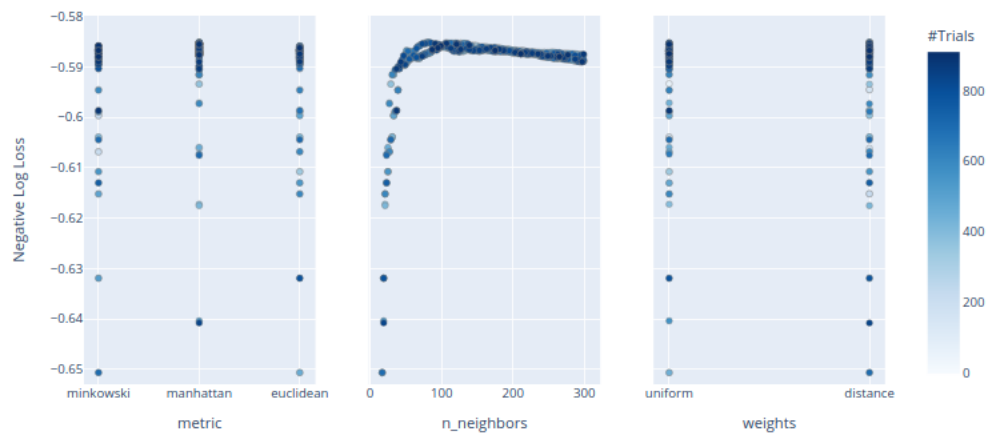


Figure 6.5: KNN performances depending on hyperparameters during Grid search on tennis data

As shown in figure 6.5, the hyperparameter with the greatest impact on the model performance is the number of neighbors. However, the performance of this algorithm is well below those previously presented, reaching 0.585 for tennis, 0.601 for basketball and 0.979 for soccer, with respectively 83, 279 and 299 neighbors, the Manhattan metric and weights based on the distance.

Due to a polynomial time complexity with respect to the number of samples, the SVM hyperparameter search space is greatly reduced compared to other algorithms.

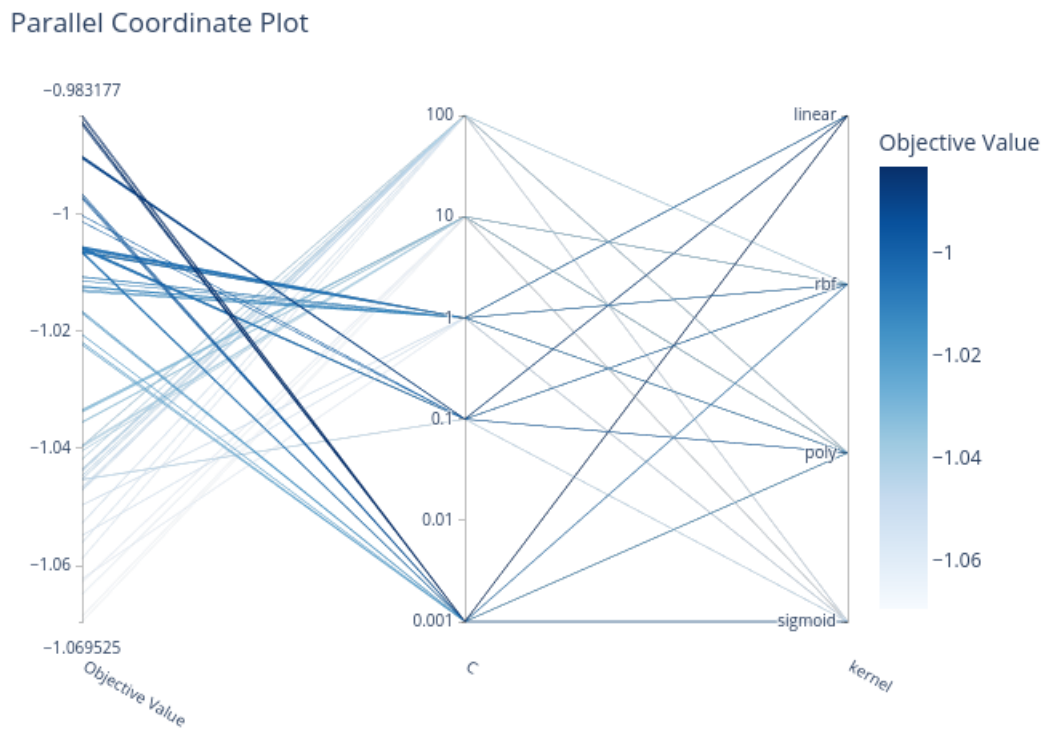


Figure 6.6: SVM performances depending on hyperparameters during Grid search on soccer data

As for using KNN algorithm, the performance of this algorithm, is well below those obtained with Logistic Regression, reaching 0.585 for tennis, 0.601 for basketball and 0.983 for soccer.

An increase in the number of hyperparameters and their possible ranges of values enlarges the hyperparameter space, thus the TPE algorithm is preferred to explore the hyperparameter search spaces of tree-based models and neural networks.

Decision Tree classifier is fitted using Gini and entropy criteria to measure the quality

of splits, with a maximum depth of the tree between 1 and 500, the minimum number of samples required to split an internal node between 2 and 1000, and a minimum number of samples required to be at a leaf node between 1 and 1000. The same hyperparameter search spaces are used for Random Forest and Extra Trees, with a number of trees chosen on a logarithmic distribution, from 100 to 1000.

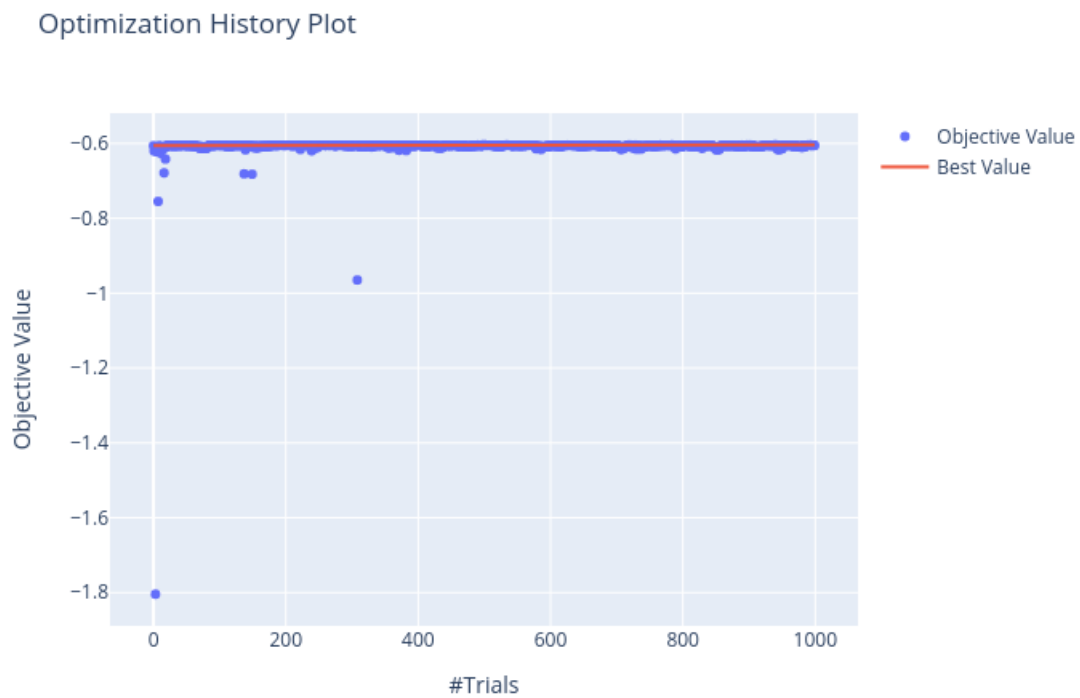


Figure 6.7: Decision Tree TPE performance history plot using basketball data

As shown in figure 6.7, and similarly for the tennis and soccer cases, except very bad settings, Decision Trees seem to have constant performances, independently of their hyperparameters. The best models reach a log loss value of 0.585 for tennis, 0.604 for

basketball and 0.983 for soccer.

As is often the case, these results can be improved with a bagging technique, with Random Forests, and Extra Trees.

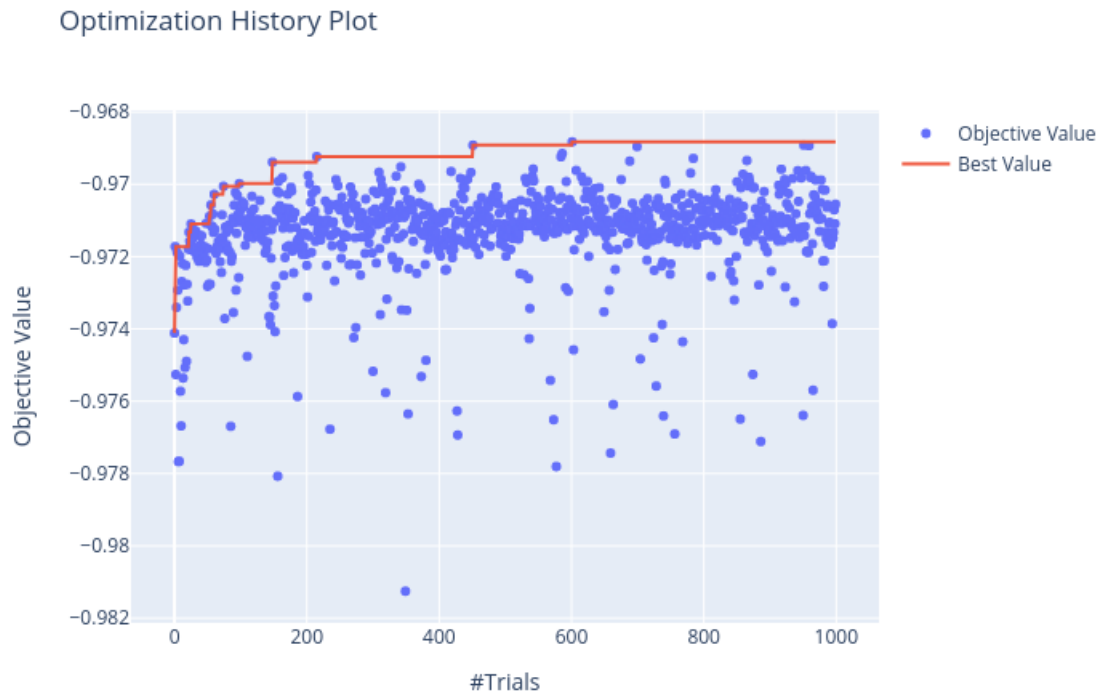


Figure 6.8: Random Forest optimization history using TPE on soccer data

Contrary to *Decision Trees* in figure 6.7, the configuration of Random Forests matters.

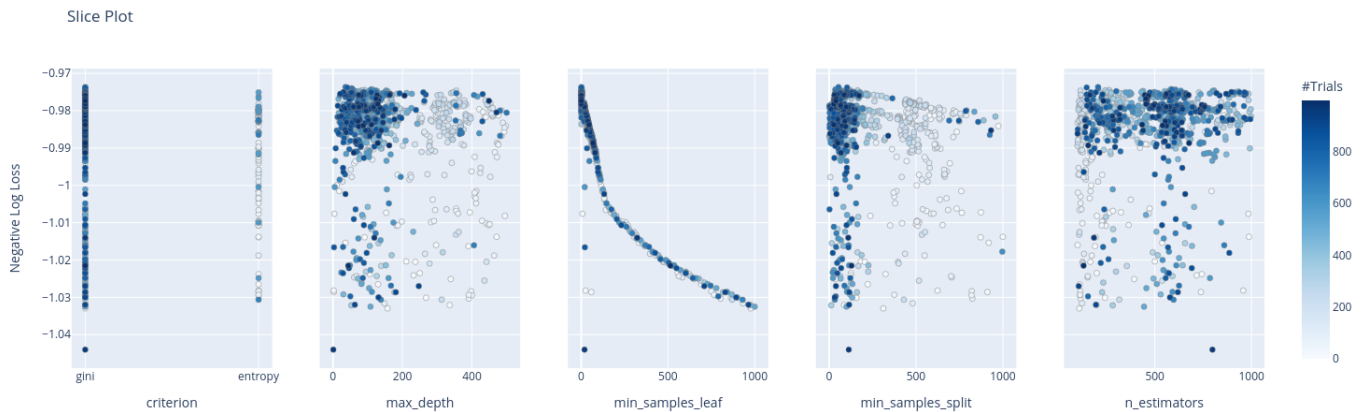


Figure 6.9: Extra Trees performances depending on hyperparameters during Bayesian search on soccer data

Whether it is Extra Trees or Random Forests, whether it is about soccer, tennis or basketball, the same conclusions on the hyperparameters can be drawn. As can be seen on 6.9, the hyperparameter with the highest impact on the model performance is the minimum number of samples required to be at a leaf node ($min_samples_leaf$), with which the best performances are obtained with relatively low values. It is the same (with a lower impact) with the minimum number of samples required to split an internal node ($min_samples_split$). On the other hand, nothing obvious appears with different numbers of trees tried.

With Random Forests, the log loss obtained is 0.57 for tennis, 0.596 for basketball and 0.969 for soccer. With Extra Trees, the log loss obtained is 0.577 for tennis, 0.607 for basketball and 0.974 for soccer.

As much as bagging methods can improve the performance of *Decision Trees*, the same is true for for boosting such as gradient boosting machines with *XGBoost* and

Catboost. Concerning boosting algorithms, many hyperparameters have been tuned:

Hyperparameter	Distribution	Range
Booster	Categorical	[gbtree, gblinear, dart]
Lambda	Logarithmic	[0.00000001, 5]
Alpha	Logarithmic	[0.00000001, 1]
Max depth	Uniform	[1, 21]
Eta	Logarithmic	[0.00000001, 1]
Gamma	Logarithmic	[0.00000001, 5]
Min child weight	Logarithmic	[1, 100]
Max delta step	Logarithmic	[0.00000001, 100]
Subsample	Logarithmic	[0.00000001, 1]
Col sample by tree	Logarithmic	[0.00000001, 1]
Col sample by level	Logarithmic	[0.00000001, 1]
Col sample by node	Logarithmic	[0.00000001, 1]
Grow policy	Categorical	[depthwise, lossguide]
Sample type	Categorical	[uniform, weight]
Normalize type	Categorical	[tree, forest]
Rate drop	Logarithmic	[0.00000001, 1]
Skip drop	Logarithmic	[0.00000001, 1]

Table 6.5: Hyperparameter search space used during Bayesian optimization of XGBoost

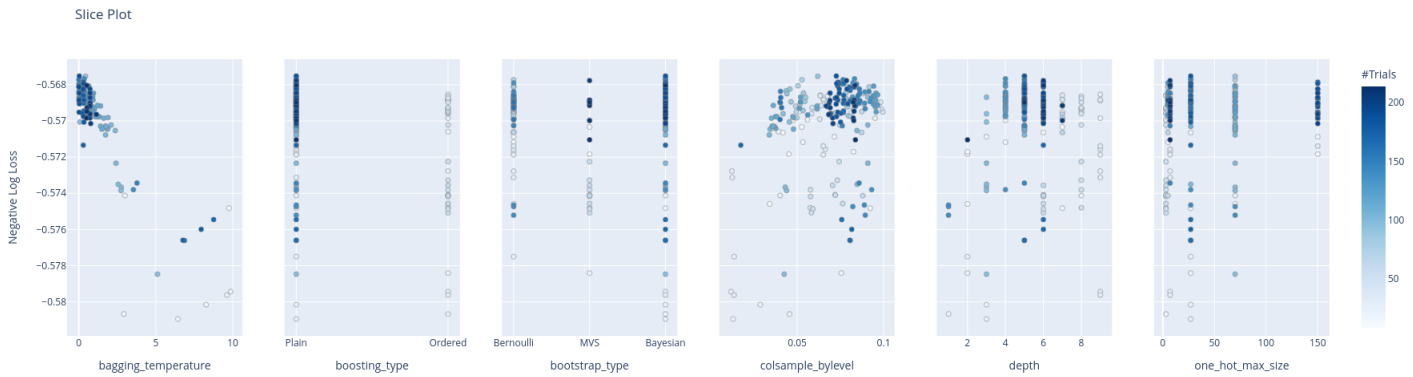


Figure 6.10: Catboost performances depending on hyperparameters during Bayesian search on tennis data

With *XGBoost*, the log loss obtained is 0.576 for tennis, 0.606 for basketball and 0.975 for soccer. With *Catboost*, the log loss obtained is 0.568 for tennis, 0.592 for basketball and 0.968 for soccer.

Due to a large training time, the multi-layer perceptron hyperparameter search space is limited to 100 different neural network structures.

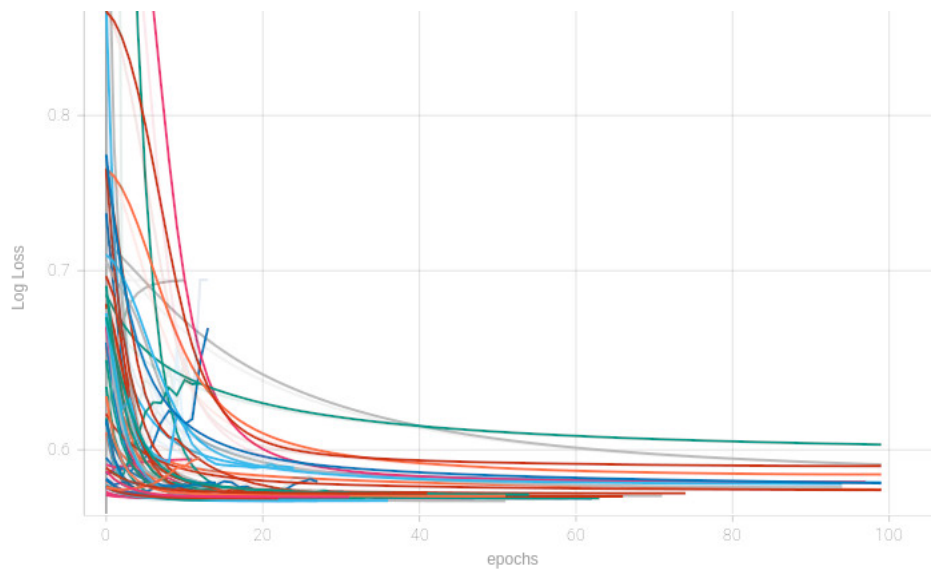


Figure 6.11: Neural Networks optimization history using TPE on tennis data

With Neural Networks, the log loss obtained is 0.575 for tennis, 0.596 for basketball and 0.972 for soccer.

In view of these results, a regular underperformance of some models is noticed, such as with the KNN and the SVM. Neural networks, that require some expertise and more tuning, do not produce better results. To go even further, Decision Trees are often outperformed by their ensemble method versions. Thus, the use of bagging and boosting provides good results and, despite its simplicity, Logistic Regression is also very competitive. Faced with performance measures that are sometimes very close, choice was made to use a Logistic Regression, with ℓ_2 regularization for tennis, a *Catboost* model for basketball and a Random Forest, with Boruta feature selection for soccer.

6.4 Model results

To understand what decisions the model is making, the SHapley Additive exPlanations (SHAP by Lundberg and Lee 2017) are used. Based on a concept coming from game theory, the Shapley values, SHAP, quantify the contribution that each feature brings to the prediction made by the model by using the average marginal contributions of a feature value across all possible coalitions of other features. These marginal contributions are defined as the impact on the forecast of adding a feature. Because adding each feature to all the possible combinations of other features is a computationally unfeasible approach, SHAP use model-type specific estimation methods. So, models become as easily interpretable as a linear model:

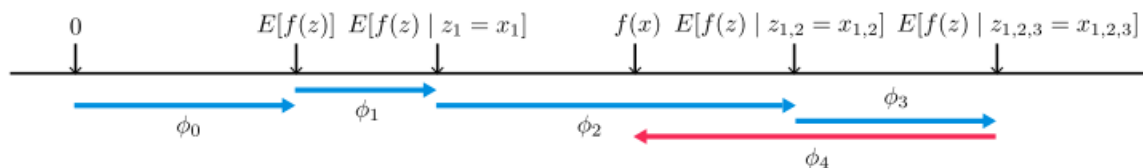


Figure 6.12: SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature, from “A Unified Approach to Interpreting Model Predictions”

Tennis

To obtain the best possible results, incremental learning is used, so that the model learns from the observations after performing a prediction. Thus, the model predicts the outcome of the games for the coming month, before partially training the model with these observations.

Below, a summary plot combining feature importance, determined by the feature order on y-axis, with feature effects, interpretable according to the color and the

location on the x-axis.

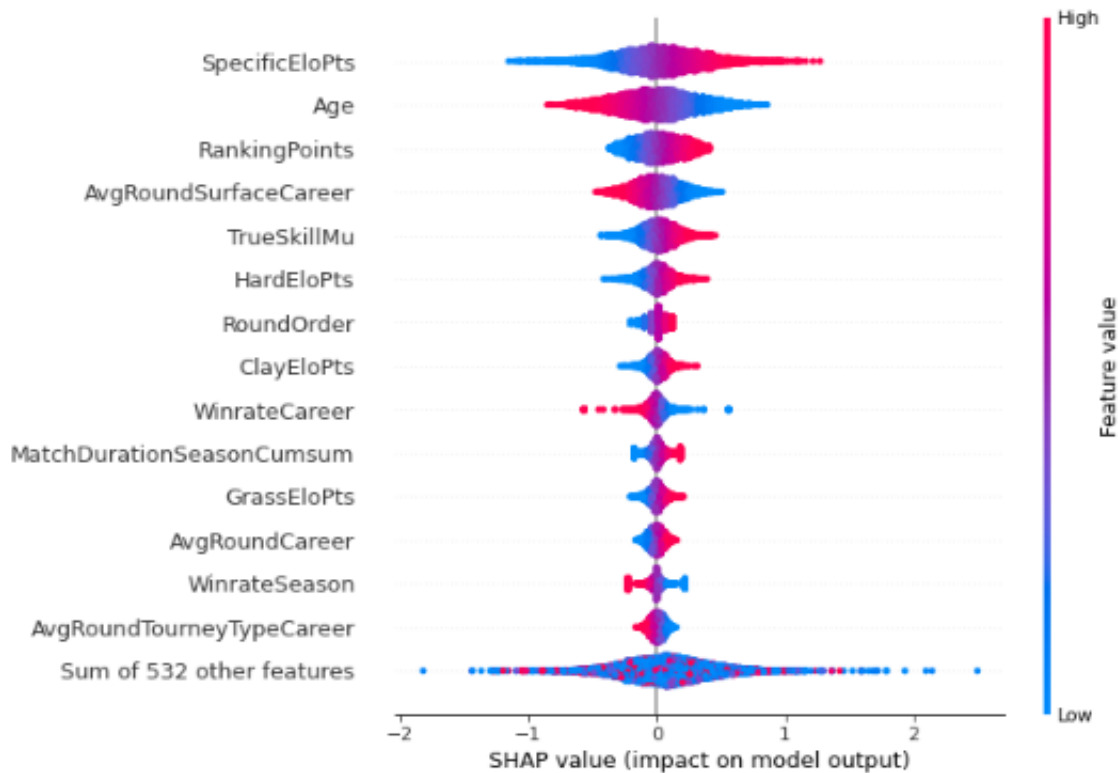


Figure 6.13: Feature impact in logistic regression model applied to ATP data based on SHAP value

At the sight of the perfect distribution of the color level on each side of the vertical bar, an excellent correlation with the target variable can be concluded for these top variables. Moreover, the feature rating systems play a decisive role in selecting the model, with 6 features (4 based on Elo, 1 on True Skill and a last one on ATP ranking) out of 14 in this top. Then, high values of these features cause higher predictions.

Below, a local interpretability of the 2018 Roland Garros final, between Rafael Nadal and Dominic Thiem. For this match, the model is predicting a victory for R. Nadal

at 0.68, and *Bet365* offered odds of 1.22 (i.e an estimated probability after unmargining of 0.78).

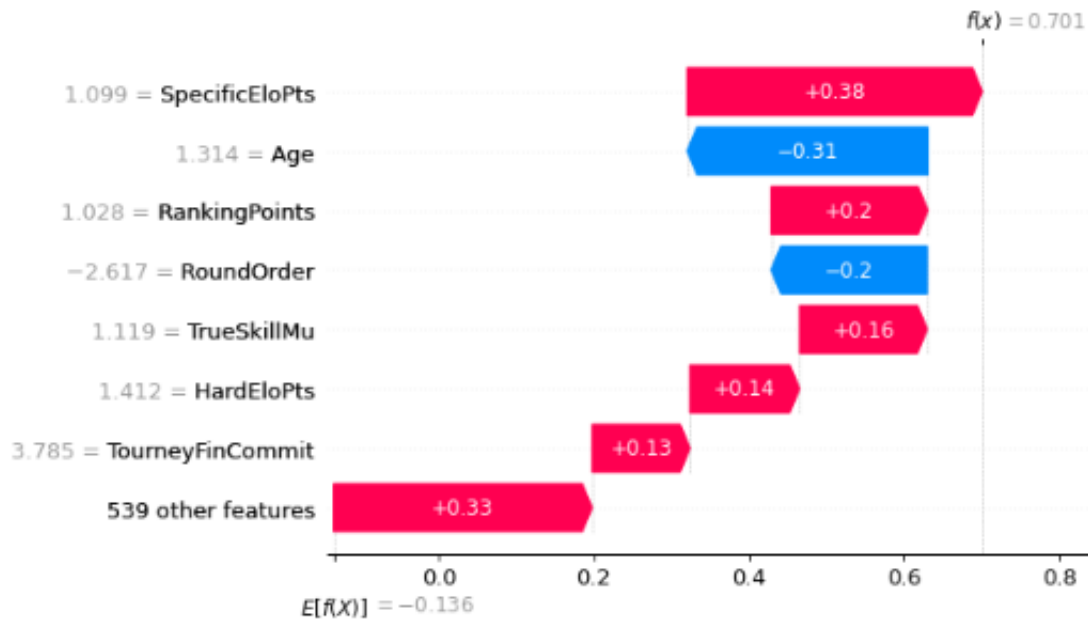


Figure 6.14: Explanation of the prediction of the 2018 Roland Garros final between R. Nadal and D. Thiem using logistic regression

So, despite an age that is not playing in his favor, R. Nadal is mainly announced as the probable winner of the match by the model thanks to rating features and all the other variables.

Season	Surface	Series	Round	n	Log Loss	Brier	ROC AUC	Accuracy
2017	Clay	-	-	806	0.603	0.417	0.734	0.664
2017	Grass	-	-	324	0.589	0.402	0.757	0.688
2017	Hard	-	-	1503	0.596	0.409	0.745	0.675
2017	-	ATP 250	-	1108	0.629	0.439	0.702	0.647
2017	-	ATP 500	-	435	0.594	0.408	0.748	0.659
2017	-	Masters 1000	-	567	0.615	0.425	0.724	0.663
2017	-	Masters Cup	-	15	0.667	0.463	0.780	0.600
2017	-	Grand Slam	-	508	0.510	0.335	0.834	0.755
2017	-	-	Round Robin	12	0.604	0.420	0.777	0.583
2017	-	-	1st Round	1180	0.601	0.416	0.733	0.657
2017	-	-	2nd Round	752	0.590	0.400	0.759	0.695
2017	-	-	3rd Round	176	0.600	0.415	0.741	0.670
2017	-	-	4th Round	48	0.513	0.334	0.811	0.750
2017	-	-	Quarterfinals	264	0.607	0.421	0.732	0.674
2017	-	-	Semifinals	134	0.626	0.438	0.704	0.626
2017	-	-	Final	67	0.572	0.385	0.795	0.761
2017	-	-	-	2633	0.598	0.411	0.743	0.673
2018	Clay	-	-	810	0.617	0.431	0.712	0.653
2018	Grass	-	-	324	0.618	0.431	0.712	0.657
2018	Hard	-	-	1503	0.599	0.412	0.743	0.679
2018	-	ATP 250	-	1112	0.651	0.460	0.663	0.616
2018	-	ATP 500	-	435	0.598	0.410	0.747	0.701
2018	-	Masters 1000	-	567	0.606	0.418	0.732	0.663
2018	-	Masters Cup	-	15	0.589	0.411	0.740	0.666
2018	-	Grand Slam	-	508	0.518	0.342	0.831	0.761
2018	-	-	Round Robin	12	0.521	0.346	0.925	0.750
2018	-	-	1st Round	1184	0.613	0.425	0.718	0.663
2018	-	-	2nd Round	752	0.613	0.424	0.725	0.656
2018	-	-	3rd Round	176	0.539	0.360	0.810	0.727
2018	-	-	4th Round	48	0.509	0.326	0.864	0.833
2018	-	-	Quarterfinals	264	0.616	0.429	0.715	0.670
2018	-	-	Semifinals	134	0.612	0.424	0.724	0.656
2018	-	-	Final	67	0.657	0.461	0.648	0.626
2018	-	-	-	2637	0.607	0.420	0.729	0.669

Table 6.6: Model metrics on ATP

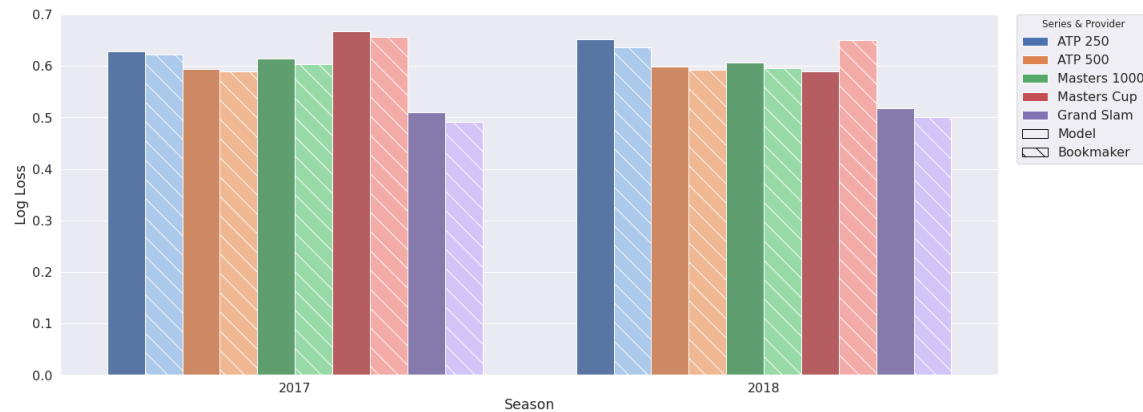


Figure 6.15: Performance comparison between probabilities forecast and the estimation of the probabilities induced by the bookmakers' odds on tennis ATP

From the Table 6.6 and Figure 6.15 above, the results appear to be very close to those observed on Table 6.2. Indeed, the difference in log loss metric is only 0.011 over the 2017 season, and 0.014 over the 2018 season. Although not very useful for probabilistic forecast, but more easily interpreted to see how close the performances are, *Bet365* has determined the winner of the match in 3567 out of 5250 games, compared to 3537 out of 5270 for the model used over the 2 seasons. In addition, a better performance of the model is noticed on highlighted cells, which concern the tournament finals, and the 2018 Masters Cup.

Basketball

For basketball, in order to maximize the performance of the *Catboost* model, the model is re-trained at the beginning of each season, and at the beginning of the playoffs.

Below, the summary plot is split between regular and playoff periods of the 2017 season:

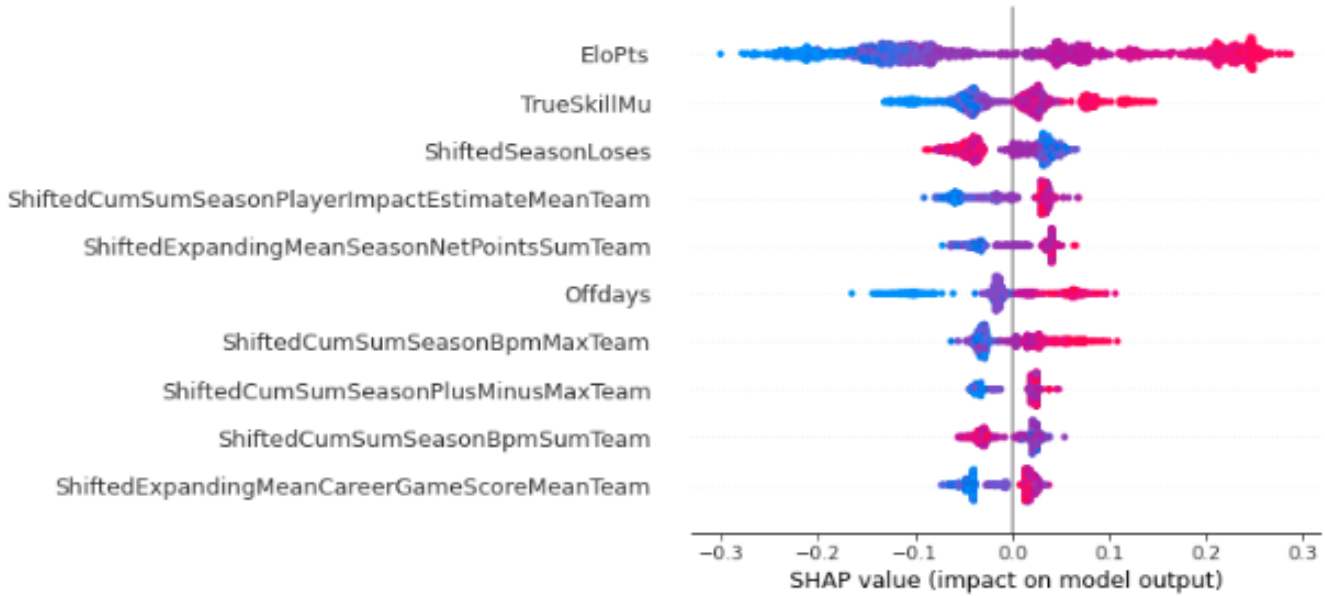


Figure 6.16: 2017 NBA regular period

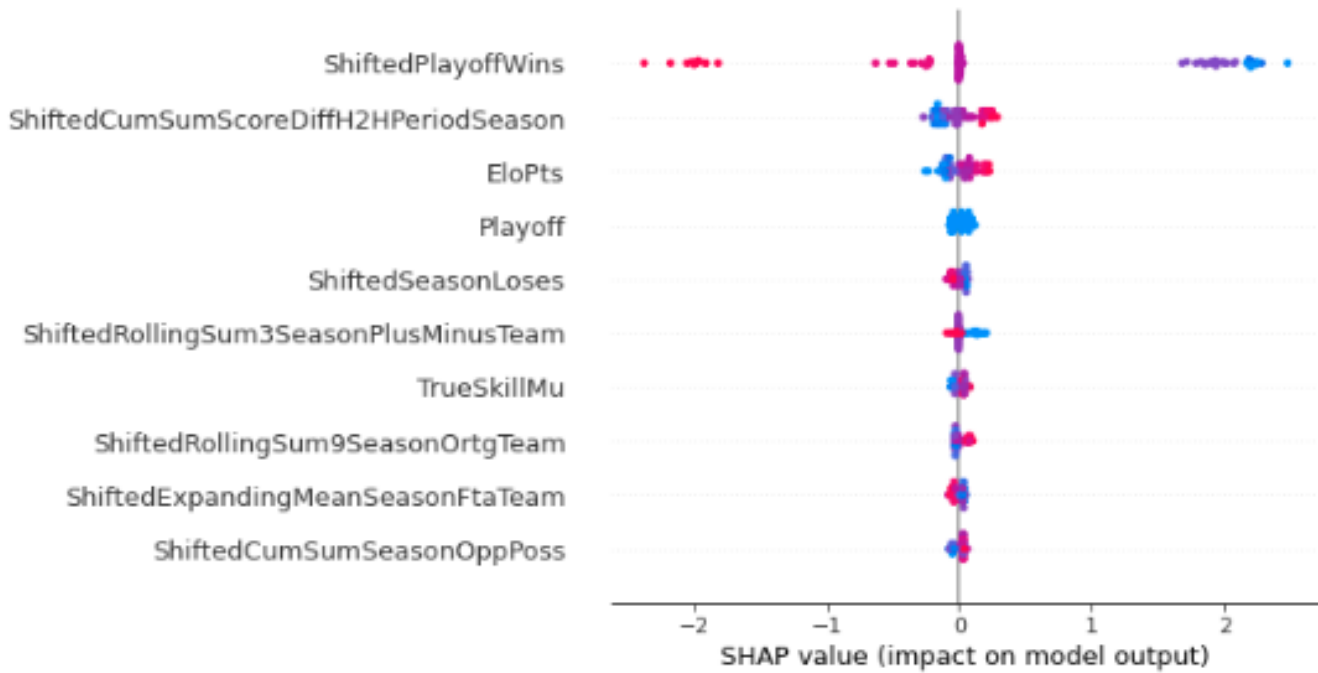


Figure 6.17: 2017 NBA playoff period

Figure 6.18: Feature impact in Catboost model applied to NBA data based on SHAP value

Depending on the period of the season, the model does not use the same features to take a decision. Thus, during the regular period, the rating system features are of primary importance. So are the Sports Analytics features, aggregated over the season, when it concerns the team, and since the beginning of their career, when it concerns the players. For the playoffs, the results of the previous games between the 2 teams during this playoff period are the most important. This is certainly due to the fact that, as only the best teams reach this stage of the competition, their level is relatively close. In addition, because the teams play several times in a very short period of time (about 1 week), the previous results are an excellent proxy for the outcome of the game.

Below, a local interpretability of the last game of the 2017 and 2018 NBA finals, opposing Golden State and Cleveland for the former and Golden State and Toronto for the latter. For these games, the model is predicting a victory for Golden State at 0.83 in 2017 and a Toronto win at 0.91 in 2018. The bookmaker offered odds of 1.55 for Golden State in 2017 and 2.24 for Toronto in 2018 (i.e an estimated probability after unmarging of 0.62 and 0.43). In view of Golden State's 108-85 victory in 2017 and Toronto's 114-110 victory in 2018, the model is better than the odds offered by the bookmaker.

As shown in Figure 6.17, the most important feature for the playoff games is the result of the previous playoff games between the two opponents. Then, with respectively 3 and 2 wins out of 3 during the previous games of the final, Golden State and Toronto have put all the chances on their side according to the model.

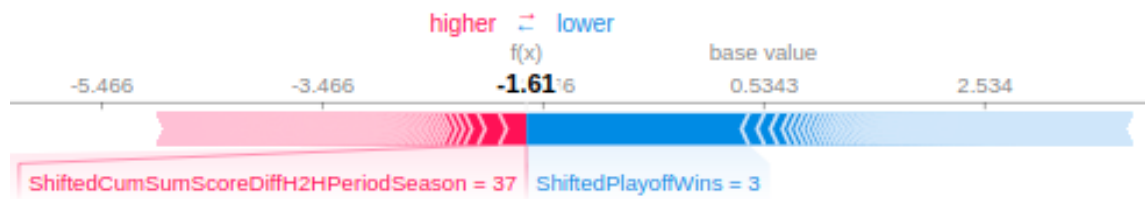


Figure 6.19: 2017 NBA final opposing Golden State to Cleveland

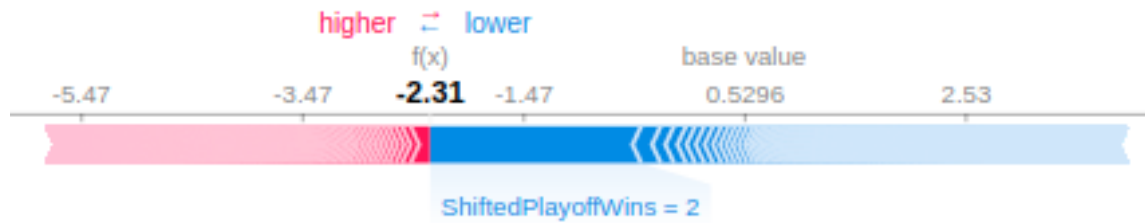


Figure 6.20: 2018 NBA final opposing Golden State to Toronto

Figure 6.21: Explanation of the prediction of the NBA finals using Catboost

Season	Playoff	n	Log Loss	Brier	ROC AUC	Accuracy
2017	0	1230	0.602	0.416	0.723	0.684
2017	1	82	0.362	0.224	0.918	0.841
2017	-	1312	0.587	0.404	0.738	0.694
2018	0	1230	0.604	0.416	0.718	0.675
2018	1	82	0.422	0.264	0.891	0.817
2018	-	1312	0.592	0.407	0.732	0.697

Table 6.7: Model metrics on NBA

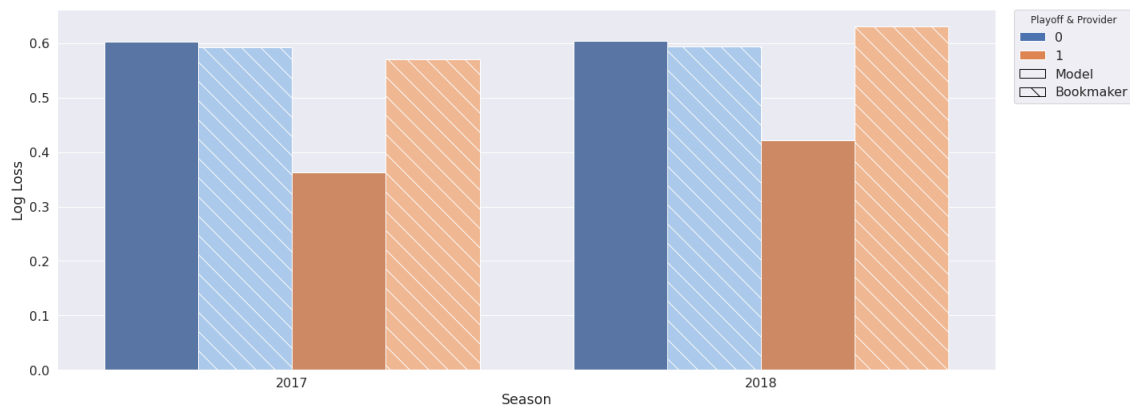


Figure 6.22: Performance comparison between probabilities forecast and the estimation of the probabilities induced by the bookmakers' odds on NBA

From Table 6.7 and Figure 6.22 above, the results per season appear to be very close to those observed on Table 6.1. With very slightly worse results for the regular period, and much better results during the playoffs (highlighted above), thanks to the *Shifted head-to-head Wins during playoff* and *Shifted head-to-head Score diff during playoff* features, the *Catboost* model achieves slightly better results. Thus, the bookmaker designated the correct favorite in 1782 games against 1825 for the model, out of 2624 of the 2017 and 2018 NBA seasons.

Soccer

To track previous games and learn from the most recent ones, the Random Forest model is re-trained at the beginning of each month during the season.

Below, the summary plot is a global explanation, to understand how the model makes decisions:



Figure 6.23: Feature impact in Random Forest model applied to soccer data based on SHAP value

As for tennis and basketball, the rating system features have a major impact in the decision making of the model by trusting 3 out of 9 top features. In addition, the variables constructed using Sports Analytics, such as VAEP, xG, or xT are paramount, occupying 4 of the top 9 places with their aggregating versions.

Below, a local interpretability of the 2017 and 2018 Champion's League finals, opposing Real Madrid to Liverpool for the former and Tottenham to Liverpool for the latter. For these games, the model forecast a victory for Real Madrid at 0.55 in 2017 and a Liverpool win at 0.376 in 2018. *Bet365* offered odds of 2.3 for Real Madrid in 2017 and 1.95 for Liverpool in 2018 (i.e an estimated probability after unmargining of 0.41 and 0.5). In view of Real Madrid's 3-1 victory and Liverpool's 2-0 victory, the bookmaker and the model has chosen the right favorite.

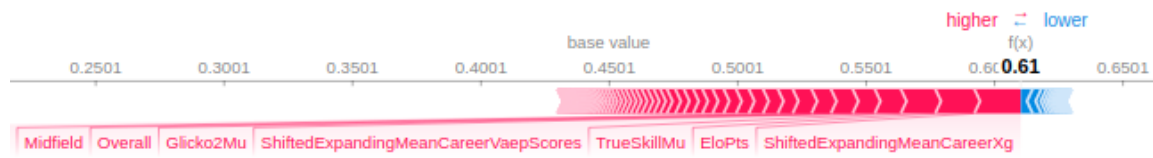


Figure 6.24: 2017 Champion’s League final opposing Real Madrid to Liverpool

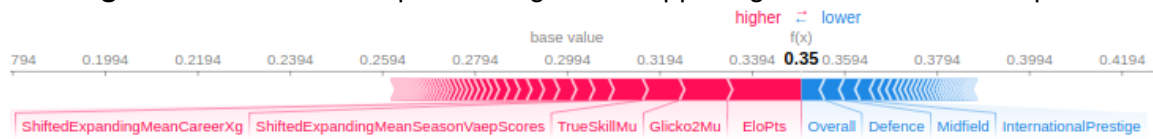


Figure 6.25: 2018 Champion’s league final opposing Tottenham to Liverpool

Figure 6.26: Explanation of the prediction of the Champion’s League finals using Random Forest

As for the global interpretation of the model in Figure 6.23, decision making for the Champion’s league finals is also mainly based on the rating system features.

Season	Championship	n	Log Loss	RPS	ROC AUC	Accuracy
2017	Bundesliga	306	1.013	0.606	0.624	0.486
2017	Premier League	380	0.958	0.570	0.691	0.547
2017	Ligue 1	380	0.970	0.575	0.667	0.539
2017	Serie A	380	0.924	0.543	0.719	0.565
2017	La Liga	380	0.985	0.586	0.636	0.526
2017	Europa League	205	1.016	0.609	0.632	0.482
2017	Champion's League	125	0.953	0.563	0.680	0.536
2017	-	1826	0.969	0.575	0.672	0.535
2018	Bundesliga	306	0.96	0.571	0.676	0.552
2018	Premier League	380	0.909	0.531	0.706	0.586
2018	Ligue 1	380	1.024	0.615	0.629	0.484
2018	Serie A	380	0.975	0.579	0.689	0.528
2018	La Liga	380	1.025	0.613	0.628	0.463
2018	Europa League	205	0.983	0.583	0.659	0.531
2018	Champion's League	125	0.947	0.554	0.705	0.600
2018	-	1826	0.979	0.583	0.67	0.522

Table 6.8: Model metrics on 5 main European soccer leagues

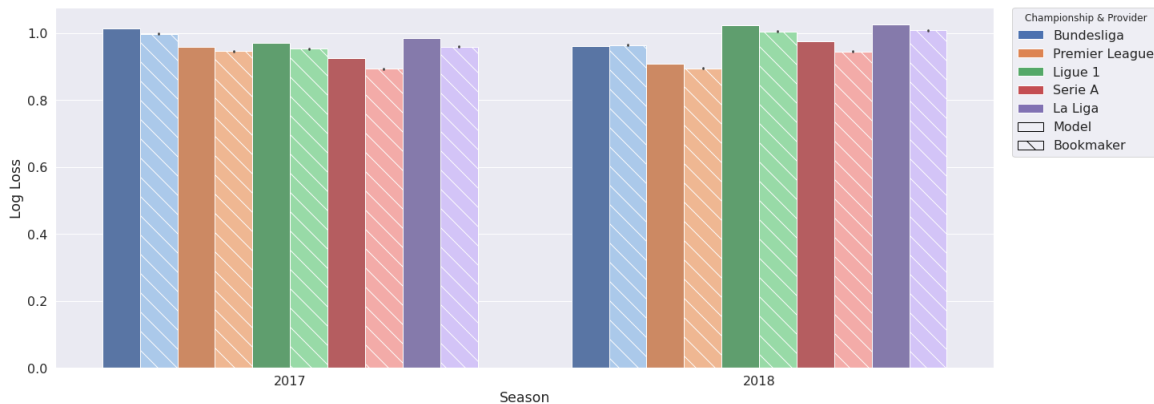


Figure 6.27: Performance comparison between probabilities forecast and the estimation of the probabilities induced by the bookmakers' odds on the 5 main European leagues

From Table 6.8 and Figure 6.27 above, the results appear to be close to those observed on Table 6.3. Indeed, the difference in log loss metric is only 0.022 over the 2017 season, and 0.016 over the 2018 season. Although not very useful for probabilistic forecast, but more easily interpreted to see how close the performances are, *Bet365* has determined the winner of the match in 1969 out of 3652 games, compared to 1931 out of 3652 for the model used over the 2 seasons. In addition, a better performance of the model is noticed on highlighted cells, which concerns the 2018 Bundesliga.

6.5 Conclusion

In view of the closeness of the results obtained by the classification models with the odds offered by the bookmakers, it seems reasonable to say that these models can be used to build reliable sports odds.

The outcome of the match being the main bet in the world of sports betting, it

is logically the one that has been previously put forward. Moreover, it is the only market on which it is possible to obtain reliable open source odds, in order to build a benchmark.

However, by simply changing the target variable, it is possible to construct a whole bunch of other odds. For example, by re-training soccer with the number of goals scored by each team in the game as target variables, hundreds of odds can be built on the total number of goals in the game, and per team, the goal difference, the exact score etc...

Below, assuming the independence of the goals scored by each team, the score matrix obtained by forecasting the 2018 Champion's League opposing Tottenham to Liverpool, with this multilabel classifier:

		Liverpool					
		0	1	2	3	4	5+
Tottenham	0	0.075(+0.0054)	0.09(+0.0023)	0.058(+0.0015)	0.026(+0.0007)	0.008(+0.0002)	0.003
	1	0.111(-0.0086)	0.132 (+0.0095)	0.086(+0.0022)	0.039(+0.0009)	0.012(+0.0003)	0.004(+0.0001)
	2	0.062(-0.0048)	0.074(-0.0057)	0.048(+0.0035)	0.022(+0.0006)	0.007(+0.0002)	0.002
	3	0.02(-0.0015)	0.031(-0.0024)	0.020(-0.0015)	0.009(+0.0006)	0.002	0.001
	4	0.01(-0.0008)	0.011(-0.0008)	0.007(-0.0005)	0.003(-0.0002)	0.001	0.0004
	5+	0.002(-0.0002)	0.002(-0.0002)	0.001(-0.0001)	0.0007	0.0002	0.00008

Table 6.9: 2018 Champion's League probabilities of final scores

Despite Liverpool's 2-0 victory, the model predicts 1-1 as the most likely final score.

By summing the blue, the gray and the red areas in Table 6.9, the winning, drawing and losing probabilities can also be easily expressed. Thus, this model always identifies Liverpool as the favorite of the match, but with a 36.65% chance of winning, against 37.56% previously. In order to match with the previous model, it is possible to adjust the probabilities with the match result probability difference, proportionally to each final score, as specified in Table 6.9.

To ensure the quality of the forecast made by such a model, the probabilities of winning, drawing and losing have been recalculated, and log loss of 0.970 and 0.979 have been respectively obtained considering the 2017 and 2018 seasons (i.e 0.001 worse than initial model).

In the end, it is also possible to use such a model in other sports betting markets, basketball and tennis being good examples. With simulations aimed at determining whether a basketball team will make the playoffs or whether a tennis player will win a tournament, or simply by modifying the target variable to predict point spreads for example.

7. Conclusion

This work presented a data-driven system to predict the outcome of the major sports for the European bookmaker market. It differs mainly from the previous works on the subject, mentioned in chapter 3, by its management of a large volume of data. Indeed, the processing of several GiBs of data, retranscribing dozens of years of matches and describing even the events during the match, in the case of soccer, was facilitated by the technological means today at our disposal. However, it remains close to the most recent approaches on the topic, by its use of classification algorithms, feature selection or hyperparameter optimization, related to machine learning practices, but also for its particular interest for the creation of new features, which can reach several thousands, related to "Sports Analytics". In view of the presentation of the results obtained in chapter 6 and the similarities between the various models used, this large volume of data and the quality of the variables used are mainly the reasons explaining the quality of the results obtained. Presenting results that are similar to the probabilities induced by the bookmakers' odds, the trained models can legitimately be used for the elaboration of sports odds. However, we are far from Marty McFly's Sports Almanac from "Back to the Future": as in any sport, "hazard" plays an important part in

determining the outcome of a match and uncertainty is always present, whoever the competitors involved in the match may be.

As this work cannot cover all the applications that require statistical modeling of event probabilities in sports for a bookmaker, many avenues remain to be explored.

In the short term, we think that it would be interesting to study:

- The extension to other leagues in the same sports disciplines. Indeed, to get a maximum amount of relevant data at our disposal, this study focused on and limited itself to the most popular championships and tournaments of the 3 sports concerned. Using a professional sports data provider such as Opta could allow for a better coverage of sport events in terms of data, and extend the approach used to women's sports for example.
- The use of other neural network architectures. Indeed, an architecture such as Recurrent Neural Networks, that uses sequences of data, could make it possible to no longer have to question past aggregation horizons when creating features.
- The extension to other markets. As presented in section 6.5, a bookmaker may be interested in kinds of other events than the outcome of the match. Thus, a model of the final score of a match is proposed here, but many other events such as the number of aces in tennis, the score difference in basketball or the scorer in a football match are possible. Moreover, odds for derivative markets, composed of several dependent events, could be proposed using methods such as copulas.
- The use of a rating proportional to the importance of the event. From a bookmaker's point of view, not all matches have the same importance. A World

Cup final does not attract the same bets as an ordinary League 1 match, so it is preferable for a bookmaker to use a more accurate model for major events. This aspect could therefore be used when evaluating the models.

- The use of a live model. The models previously presented only used the information available at the start of the match to suggest pre-live odds. However, as bettors are increasingly interested in betting during the match, it would be interesting to propose a model that could be updated according to the events taking place during the match.
- The use of experts' opinion. Finally, in view of the overperformance of certain sports traders or confirmed bettors, the opinion of sports experts could be used as a variable in the model to improve the predictions of the most important matches.

Bibliography

Books

Brownlee, J. (2020). *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer. URL: <http://www-stat.stanford.edu/~tibs/ElemStatLearn/>.

Oliver, D. (2004). *Basketball On Paper: Rules And Tools For Performance Analysis*. Potomac Books Inc.

Poisson, S.D. (1837). *Recherches sur la probabilité des jugements en matière criminelle et en matière civile: précédées des règles générales du calcul des probabilités*. Nineteenth Century Collections Online (NCCO): Science, Technology, and Medicine: 1780-1925. Bachelier. ISBN: 978-0-608-35646-4. URL: <https://books.google.fr/books?id=uB80AAAAQAAJ>.

Varma, Subir and Sanjiv Das (Apr. 2018). *Introduction to Deep Learning*.

Moroney, M. J. (1956). "Facts from figures". In: 3rd ed. Penguin Books. Chap. Goals, Floods, and Horse-kicks - The Poisson Distribution. ISBN: 9780140202366.

Articles

Aitchison, J. and S. D. Silvey (June 1957). “The Generalization of Probit Analysis to the case of multiple responses”. In: *Biometrika* 44.1-2, pp. 131–140. ISSN: 0006-3444. DOI: 10.1093/biomet/44.1-2.131. eprint: <https://academic.oup.com/biomet/article-pdf/44/1-2/131/752594/44-1-2-131.pdf>. URL: <https://doi.org/10.1093/biomet/44.1-2.131>.

Altman, N. S. (1992). “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression”. In: *The American Statistician* 46.3, pp. 175–185. ISSN: 00031305. URL: <http://www.jstor.org/stable/2685209>.

Angelini, Giovanni and Luca De Angelis (2017). “PARX model for football matches predictions”. en. In: p. 26.

Barnett, T. and Stephen R. Clarke (2005). “Combining player statistics to predict outcomes of tennis matches”. In: DOI: 10.1093/IMAMAN/DPI001.

Bayes, T. (1763). “An essay towards solving a problem in the doctrine of chances”. In: *Phil. Trans. of the Royal Soc. of London* 53, pp. 370–418.

Berkson, Joseph (1944). “Application of the Logistic Function to Bio-Assay”. In: *Journal of the American Statistical Association* 39.227, pp. 357–365. ISSN: 01621459. URL: <http://www.jstor.org/stable/2280041>.

Bliss, C. I. (1934). “The methods of Probit”. In: *Science* 79.2037, pp. 38–39. ISSN: 0036-8075. DOI: 10.1126/science.79.2037.38. eprint: <https://science.sciencemag.org/content/79/2037/38.full.pdf>. URL: <https://science.sciencemag.org/content/79/2037/38>.

Bradley, R.A and M.E Terry (Dec. 1952). “Rank Analysis of incomplete block designs: the method of paired comparisons”. In: *Biometrika* 39.3-4, pp. 324–345. ISSN: 0006-3444. DOI: 10.1093/biomet/39.3-4.324. eprint: https://academic.oup.com/biomet/article-pdf/39/3-4/324/288822/biomet_39_3_324.pdf.

oup.com/biomet/article-pdf/39/3-4/324/930466/39-3-4-324.pdf. URL: <https://doi.org/10.1093/biomet/39.3-4.324>.

Breiman, Leo (Oct. 2001). “Machine Learning, Volume 45, Number 1 - SpringerLink”. In: *Machine Learning* 45, pp. 5–32. DOI: 10.1023/A:1010933404324.

Cattelan, Manuela, Cristiano Varin, and David Firth (2013). “Dynamic Bradley–Terry modelling of sports tournaments”. In: *Journal of the Royal Statistical Society Series C* 62.1, pp. 135–150. URL: <https://EconPapers.repec.org/RePEc:bla:jorssc:v:62:y:2013:i:1:p:135-150>.

Champagne, Loïc and Léo Gerville-Réache (2015). “Autour des procédures de classement : exemples du tennis, du tennis de table et du golf”. fr. In: *Journal de la société française de statistique* 156.2, pp. 5–24. URL: http://www.numdam.org/item/JSFS_2015__156_2_5_0/.

Constantinou, Anthony and Norman Fenton (Jan. 2012). “Solving the Problem of Inadequate Scoring Rules for Assessing Probabilistic Football Forecast Models”. In: *Journal of Quantitative Analysis in Sports* 8. DOI: 10.1515/1559-0410.1418.

Constantinou, Anthony, Norman Fenton, and Martin Neil (Dec. 2012). “pi-football: A Bayesian network model for forecasting Association Football match outcomes. Knowledge-Based Systems, 36, 322-339”. In: *Knowledge-Based Systems* 36, pp. 332–339. DOI: 10.1016/j.knosys.2012.07.008.

Cortes, Corinna and Vladimir Vapnik (Sept. 1995). “Support-Vector Networks”. In: *Mach. Learn.* 20.3, 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: <https://doi.org/10.1023/A:1022627411411>.

Critchlow, Douglas E. and Michael A. Fligner (1991). “Paired comparison, triple comparison, and ranking experiments as generalized linear models, and their implementation on GLIM”. In: *Psychometrika* 56.3. Place: Germany Publisher:

Springer, pp. 517–533. ISSN: 1860-0980(Electronic),0033-3123(Print). DOI: 10.1007/BF02294488.

Crowder, Martin et al. (2002). “Dynamic modelling and prediction of English Football League matches for betting”. In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 51.2, pp. 157–168. DOI: <https://doi.org/10.1111/1467-9884.00308>. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9884.00308>. URL: <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9884.00308>.

Davidson, Roger R. (1970). “On Extending the Bradley-Terry Model to Accommodate Ties in Paired Comparison Experiments”. In: *Journal of the American Statistical Association* 65.329, pp. 317–328. ISSN: 01621459. URL: <http://www.jstor.org/stable/2283595>.

Dixon, Mark J and Stuart G Coles (1997). “Modelling association football scores and inefficiencies in the football betting market”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 46.2, pp. 265–280.

Epstein, Edward S. (1969). “A Scoring System for Probability Forecasts of Ranked Categories”. In: *Journal of Applied Meteorology (1962-1982)* 8.6, pp. 985–987. ISSN: 00218952, 2163534X. URL: <http://www.jstor.org/stable/26174707>.

Fix, Evelyn and J. L. Hodges (1989). “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. In: *International Statistical Review / Revue Internationale de Statistique* 57.3, pp. 238–247. ISSN: 03067734, 17515823. URL: <http://www.jstor.org/stable/1403797>.

Forrest, D. and R. Simmons (2000). “Forecasting sport: the behaviour and performance of football tipsters”. In: *International Journal of Forecasting* 16, pp. 317–331.

- Friedman, Jerome H. (2001). “Greedy Function Approximation: A Gradient Boosting Machine”. In: *The Annals of Statistics* 29.5, pp. 1189–1232. ISSN: 00905364. URL: <http://www.jstor.org/stable/2699986>.
- Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006). “Extremely Randomized Trees”. In: *Machine Learning* 36, pp. 3–42. DOI: 10.1007/s10994-006-6226-1. URL: <https://hal.archives-ouvertes.fr/hal-00341932>.
- Goddard, John and Ioannis Asimakopoulos (2004). “Forecasting football results and the efficiency of fixed-odds betting”. In: *Journal of Forecasting* 23.1, pp. 51–66. DOI: <https://doi.org/10.1002/for.877>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.877>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/for.877>.
- Graham, I. and H. Stott (2008). “Predicting bookmaker odds and efficiency for UK football”. In: *Applied Economics* 40.1, pp. 99–109. DOI: 10.1080/00036840701728799. eprint: <https://doi.org/10.1080/00036840701728799>. URL: <https://doi.org/10.1080/00036840701728799>.
- Hill, I. D. (1974). “Association Football and Statistical Inference”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 23.2. Publisher: [Wiley, Royal Statistical Society], pp. 203–208. ISSN: 0035-9254. DOI: 10.2307/2347001. URL: <https://www.jstor.org/stable/2347001> (visited on 04/20/2021).
- Huang, Tzu-Kuo, Ruby C. Weng, and Chih-Jen Lin (2006). “Generalized Bradley-Terry Models and Multi-Class Probability Estimates”. In: *Journal of Machine Learning Research* 7.4, pp. 85–115. URL: <http://jmlr.org/papers/v7/huang06a.html>.
- Hvattum, Lars Magnus and Halvard Arntzen (2010). “Using ELO ratings for match result prediction in association football”. In: *International Journal of Forecasting* 26.3, pp. 460–470. URL: <https://ideas.repec.org/a/eee/intfor/v26yi3p460-470.html>.

Karlis, Dimitris and Ioannis Ntzoufras (1997). “On modelling soccer data”. en. In: *Student* 3, pp. 229–244.

— (2003). “Analysis of sports data by using bivariate Poisson models”. en. In: p. 13.

Klaassen, Franc J.G.M. and Jan R. Magnus (July 2003). “Forecasting the winner of a tennis match”. en. In: *European Journal of Operational Research* 148.2, pp. 257–267. ISSN: 03772217. DOI: 10.1016/S0377-2217(02)00682-3. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377221702006823> (visited on 05/06/2021).

Koning, Ruud H. (2000). “Balance in Competition in Dutch Soccer”. In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 49.3, pp. 419–431. DOI: <https://doi.org/10.1111/1467-9884.00244>. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9884.00244>. URL: <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9884.00244>.

Koopman, Siem Jan and Rutger Lit (2012). “Forecasting football match results in national league competitions using score-driven time series models”. en. In: p. 42.

Kubatko, Justin et al. (Feb. 2007). “A Starting Point for Analyzing Basketball Statistics”. In: *Journal of Quantitative Analysis in Sports* 3, pp. 1–1. DOI: 10.2202/1559-0410.1070.

Kursa, Miron B. and Witold R. Rudnicki (2010). “Feature Selection with the Boruta Package”. In: *Journal of Statistical Software* 36.11, 1–13. DOI: 10.18637/jss.v036.i11. URL: <https://www.jstatsoft.org/index.php/jss/article/view/v036i11>.

Kuypers, Tim (2000). “Information and efficiency: an empirical study of a fixed odds betting market”. In: *Applied Economics* 32.11, pp. 1353–1363. URL: <https://EconPapers.repec.org/RePEc:taf:applec:v:32:y:2000:i:11:p:1353-1363>.

- Lam, Max W. Y. (2018). “One-Match-Ahead Forecasting in Two-Team Sports with Stacked Bayesian Regressions”. In: *Journal of Artificial Intelligence and Soft Computing Research* 8.3, pp. 159–171. DOI: doi:10.1515/jaiscr-2018-0011. URL: <https://doi.org/10.1515/jaiscr-2018-0011>.
- Lee, Alan J. (Jan. 1997). “Modeling Scores in the Premier League: Is Manchester United *Really* the Best?” en. In: *CHANCE* 10.1, pp. 15–19. ISSN: 0933-2480, 1867-2280. DOI: 10.1080/09332480.1997.10554791. URL: <http://www.tandfonline.com/doi/full/10.1080/09332480.1997.10554791> (visited on 05/04/2021).
- Ley, Christophe and Van de Wiele, Tom and Van Eetvelde, Hans (2019). “Ranking soccer teams on the basis of their current strength : a comparison of maximum likelihood approaches”. eng. In: *STATISTICAL MODELLING* 19.1, 55–77. ISSN: 1471-082X. URL: {<http://dx.doi.org/10.1177/1471082x18817650>}.
- Li, Chin-Shang et al. (1999). “Multivariate Zero-Inflated Poisson Models and Their Applications”. In: *Technometrics* 41.1, pp. 29–38. ISSN: 00401706. URL: <http://www.jstor.org/stable/1270992>.
- Magnus, Jan R. and Franc J. G. M. Klaassen (1999). “On the Advantage of Serving First in a Tennis Set: Four Years at Wimbledon”. In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 48.2, pp. 247–256. ISSN: 00390526, 14679884. URL: <http://www.jstor.org/stable/2681190>.
- Maher, M. J. (Sept. 1982). “Modelling association football scores”. en. In: *Statistica Neerlandica* 36.3, pp. 109–118. ISSN: 0039-0402, 1467-9574. DOI: 10.1111/j.1467-9574.1982.tb00782.x. URL: <http://doi.wiley.com/10.1111/j.1467-9574.1982.tb00782.x> (visited on 04/20/2021).
- Mchale, I.G. and Alex Morton (Apr. 2011). “A Bradley-Terry type model for forecasting tennis match results”. In: *International Journal of Forecasting* 27, pp. 619–630. DOI: 10.1016/j.ijforecast.2010.04.004.

-
- Mchale, I.G. and Phil Scarf (May 2011). “Modelling the dependence of goals scored by opposing teams in international soccer matches”. In: *Statistical Modelling* 11, pp. 219–236. DOI: 10.1177/1471082X1001100303.
- McKelvey, Richard D. and William Zavoina (1975). “A statistical model for the analysis of ordinal level dependent variables”. In: *The Journal of Mathematical Sociology* 4.1, pp. 103–120. DOI: 10.1080/0022250X.1975.9989847. eprint: <https://doi.org/10.1080/0022250X.1975.9989847>. URL: <https://doi.org/10.1080/0022250X.1975.9989847>.
- Murphy, A. (1970). “THE RANKED PROBABILITY SCORE AND THE PROBABILITY SCORE: A COMPARISON”. In: *Monthly Weather Review* 98, pp. 917–924.
- Newton, P. and J. Keller (2005). “Probability of Winning at Tennis I. Theory and Data”. In: *Studies in Applied Mathematics* 114, pp. 241–269.
- Pedregosa, F. et al. (2011). “Scikit-learn”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Rao, P. V. and L. L. Kupper (1967). “Ties in Paired-Comparison Experiments: A Generalization of the Bradley-Terry Model”. In: *Journal of the American Statistical Association* 62.317, pp. 194–204. DOI: 10.1080/01621459.1967.10482901. eprint: <https://www.tandfonline.com/doi/pdf/10.1080/01621459.1967.10482901>. URL: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10482901>.
- Reep, C., Richard Pollard, and B. Benjamin (Jan. 1971). “Skill and Chance in Ball Games”. In: *Journal of the Royal Statistical Society. Series A (General)* 134, p. 623. DOI: 10.2307/2343657.
- Rosenblatt, F. (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65 6, pp. 386–408.

- Rue, Håvard and Oyvind Salvesen (Jan. 2000). “Predicting and Retrospective Analysis of Soccer Matches in a League”. In.
- Wurp, Hendrik van der et al. (2019). “Generalised Joint Regression for Count Data with a Focus on Modelling Football Matches”. In: DOI: 10.48550/ARXIV.1908.00823. URL: <https://arxiv.org/abs/1908.00823>.
- Zermelo, E. (1929). “Die Berechnung der Turnier-Ergebnisse als ein Maximumproblem der Wahrscheinlichkeitsrechnung.” In: *Mathematische Zeitschrift* 29, pp. 436–460. URL: <http://eudml.org/doc/168081>.
- Akiba, Takuya et al. (2019). “Optuna”. In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Beckler, Matthew, Hongfei Wang, and Michael Papamichael (2008). “NBA Oracle”. In: URL: https://www.mbeckler.org/coursework/2008-2009/10701_report.pdf.
- Chen, Tianqi and Carlos Guestrin (2016). “XGBoost”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: ACM, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: <http://doi.acm.org/10.1145/2939672.2939785>.
- Decroos, Tom et al. (2019). “Actions Speak Louder Than Goals: Valuing Player Actions in Soccer”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’19. Anchorage, AK, USA: ACM, pp. 1851–1861. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330758. URL: <http://doi.acm.org/10.1145/3292500.3330758>.
- Herbrich, Ralf, Tom Minka, and Thore Graepel (2007). “TrueSkill(TM): A Bayesian Skill Rating System”. In: *Advances in Neural Information Processing Systems 20*. MIT Press, pp. 569–576. URL: <https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/>.

-
- Ho, Tin Kam (1995). “Random Decision Forests”. In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*. ICDAR '95. USA: IEEE Computer Society, p. 278. ISBN: 0818671289.
- Jin, Haifeng, Qingquan Song, and Xia Hu (2019). “Auto-Keras”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, pp. 1946–1956.
- Lasek, Jan (2016). “Euro 2016 Predictions using team rating systems”. In: *3rd Workshop on Machine Learning and Data Mining for Sports Analytics*.
- Lucey, Patrick et al. (2015). “"Quality vs Quantity": Improved Shot Prediction in Soccer using Strategic Features from Spatiotemporal Data”. In.
- Macdonald, Brian (2012). “An Expected Goals Model for Evaluating NHL Teams and Players”. In.
- Robberechts, Pieter and Jesse Davis (2020). “How data availability affects the ability to learn good xG models”. In: *International Workshop on Machine Learning and Data Mining for Sports Analytics*. Springer, pp. 17–27.
- Steffen, Paul, Léo Gerville-Réache, and Nicolas Bisoffi (June 2019). “PARIS SPORTIFS AU FOOTBALL : L'INTERET DES EXPECTED GOALS”. In: *JDS 2019*. Nancy, France. URL: <https://hal.archives-ouvertes.fr/hal-02150047>.
- Torres, Renato Amorim (2013). “Prediction of NBA games based on Machine Learning Methods”. In: URL: https://homepages.cae.wisc.edu/~ece539/fall113/project/AmorimTorres_rpt.pdf.
- Van Roy, Maaïke et al. (2020). “Valuing On-the-Ball Actions in Soccer: A Critical Comparison of xT and VAEP”. In: *Proceedings of the AAAI-20 Workshop on Artificial Intelligence in Team Sports*. AITS. AI in Team Sports Organising Committee.

Websites and Documentations

2KMTCentral (n.d.). URL: <https://2kmtcentral.com/>.

ATP Tour (n.d.). URL: <https://www.atptour.com/>.

Football-Data.co.uk (n.d.). URL: <https://www.football-data.co.uk/>.

SoFIFA (2009). URL: <https://sofifa.com/>.

Sports Book Reviews Online (n.d.). URL: <https://www.sportsbookreviewsonline.com/>.

Tennis-Data.co.uk (n.d.). URL: <http://www.tennis-data.co.uk/>.

FIFA (n.d.). *World Football Elo Ratings*. URL: <https://www.eloratings.net/>.

Safak, Ali and Cristiano Acconci (2008). *Whoscored*. URL: <https://www.whoscored.com/>.

Seidel, Matthias (2000). *Transfermarkt*. URL: <https://www.transfermarkt.com/>.

Silver, Nate (n.d.). *Five Thirty Eight*. URL: <https://fivethirtyeight.com/>.

Virshbo, Jay (2004). *Basketball-Reference*. URL: <https://www.basketball-reference.com/>.

Dask Development Team (2016). *Dask*. URL: <https://dask.org>.

Geek, Seat (2020). *TheFuzz*. URL: <https://github.com/seatgeek/thefuzz>.

Huggins, Jason (2004). *Selenium*. URL: <https://selenium.dev/selenium/docs/api/py/>.

Richardson, Leonard (2007). *Beautiful Soup*. URL: <https://www.crummy.com/software/BeautifulSoup/>.

Ritz, Kenneth (2011). *Requests*. URL: <https://requests.readthedocs.io/en/master/>.

A. Appendix

Tm	Team
Opp	Opponent
Pts	Points
FG	Field Goals
FGA	Field Goal Attempts
ORB	Offensive Rebounds
DRB	Defensive Rebounds
TRB	Total Rebounds
FT	Free Throws
FTA	Free Throw Attempts
AST	Assists
STL	Steals
BLK	Blocks
TOV	Turnovers
PF	Personal Fouls
Poss	Possession
3P	3-Point Field Goals
3PA	3-Point Field Goal Attempts
eFG%	Effective Field Goal Percentage
TS%	True Shooting Percentage
USG%	Usage Percentage
DRtg	Defensive Rating
ORtg	Offensive Rating
BPM	Box Plus/Minus
3PAr	3-Point Attempt Rate
FTr	Free Throw Attempt Rate

Table A.1: Basketball abbreviation table

- RankNumber
- RankingPoints
- Age
- SecondServePointsWonSeasonRatio
- BreakPointsSavedSeasonRatio
- FirstServeReturnWonSurfaceSeasonRatio
- ReturnPointsWonSurfaceSeasonRatio
- TotalPointsWonSeasonRatio
- TotalPointsWonTourneyRatio
- TotalPointsWonSurfaceSeasonRatio
- DoubleFaultsCareerRatio
- ServeRatingSurfaceCareerEwm
- ReturnRatingCareerEwm
- MatchDurationTourneyCumsum
- AvgRoundSeason
- AvgRoundSurfaceCareer
- EloPts
- SpecificEloPts
- HardEloPts
- ClayEloPts
- CarpetEloPts
- TrueSkillSigma
- Glicko2Mu
- Glicko2Sigma
- AtpLogRank
- WinrateH2h
- EwmWinrateCareer
- EwmWinrateSurfaceCareer
- Exp

Figure A.1: Tennis features selected using *Boruta* algorithm

Feature	Xi	Feature	Xi
SpecificEloPts	0.446	BreakPointsSavedCareerRatio	0.355
EloPts	0.442	Exp	0.354
RankNumber	0.436	PlayerSeed1_nan	0.354
AtpLogRank	0.434	ServeRatingSurfaceCareerEwm	0.354
RankingPoints	0.433	WeightLbs	0.353
TrueSkillMu	0.433	FirstServePointsWonCareerRatio	0.353
Glicko2Mu	0.426	Rolling9WinrateCareer	0.353
HardEloPts	0.405	SecondServePointsWonSeasonRatio	0.352
TotalPointsWonSurfaceCareerRatio	0.389	MatchDurationTourneyCumsum	0.350
TotalPointsWonCareerRatio	0.389	PlayerGamesLoseTourneyCumsum	0.349
AvgRoundTourneyTypeCareer	0.386	BreakPointsSavedSeasonRatio	0.349
AvgRoundSurfaceCareer	0.385	PlayerSeed1_1	0.348
MatchDurationSeasonCumsum	0.384	BreakPointsSavedSurfaceCareerRatio	0.348
AvgRoundCareer	0.381	FirstServePointsWonSurfaceSeasonRatio	0.347
AvgRoundSeason	0.379	FirstServeReturnWonSurfaceCareerRatio	0.347
TrueSkillSigma	0.377	FirstServePointsWonSeasonRatio	0.347
CarpetEloPts	0.373	PlayerSeed2_1	0.347
Glicko2Sigma	0.371	Backhand1_unknown backhand	0.347
TotalPointsWonSeasonRatio	0.371	PlayerSeed1_WC	0.346
ClayEloPts	0.370	ServicePointsWonTourneyRatio	0.346
BMI	0.367	PlayerSetsLoseTourneyCumsum	0.346
SecondServePointsWonCareerRatio	0.366	AcesTourneyRatio	0.346
ServicePointsWonCareerRatio	0.366	PlayerSeed1_2	0.346
GrassEloPts	0.365	PlayerSeed2_WC	0.346
SecondServePointsWonSurfaceCareerRatio	0.364	ReturnPointsWonSurfaceCareerRatio	0.345
TotalPointsWonSurfaceSeasonRatio	0.364	AcesSurfaceCareerRatio	0.345
ServicePointsWonSeasonRatio	0.363	SecondServeReturnWonSurfaceCareerRatio	0.345
ServicePointsWonSurfaceCareerRatio	0.362	ReturnPointsWonCareerRatio	0.345
ServicePointsWonSurfaceSeasonRatio	0.362	SecondServePointsWonSurfaceSeasonRatio	0.344
PlayerSeed2_nan	0.359	WinrateSeason	0.344
FirstServePointsWonSurfaceCareerRatio	0.357		

Table A.2: Tennis ξ_i correlation

- ShiftedExpandingMeanSeasonBpmTeam
- ShiftedCumSumSeasonBpmTeam
- ShiftedExpandingMeanSeasonEfg
- ShiftedExpandingMeanSeasonOrtg
- ShiftedExpandingMeanSeasonPythWins
- ShiftedRollingSum5SeasonPythWins
- ShiftedRollingSum9SeasonPythWins
- ShiftedExpandingMeanSeasonOffensiveEfficiency
- ShiftedSeasonWins
- ShiftedSeasonLoses
- ShiftedSeasonWIRatio
- EloPts
- TrueSkillMu
- Glicko2Mu
- ShiftedCumSumCareer_3pMedianTeam
- ShiftedCumSumCareerStlMeanTeam
- ShiftedCumSumCareerPfMeanTeam
- ShiftedExpandingMeanCareerPlusMinusMinTeam
- ShiftedExpandingMeanCareerPlusMinusMaxTeam
- ShiftedExpandingMeanSeasonPlusMinusMinTeam
- ShiftedExpandingMeanSeasonPlusMinusMaxTeam
- ShiftedCumSumCareerPlusMinusMinTeam
- ShiftedCumSumCareerPlusMinusMaxTeam
- ShiftedCumSumSeasonPlusMinusMinTeam
- ShiftedCumSumSeasonPlusMinusMaxTeam
- ShiftedExpandingMeanCareerOrtgMedianTeam
- ShiftedExpandingMeanSeasonOrtgMedianTeam
- ShiftedCumSumSeasonOrtgMedianTeam
- ShiftedExpandingMeanCareerDrtgSumTeam
- ShiftedExpandingMeanCareerDrtgMeanTeam
- ShiftedExpandingMeanCareerDrtgMaxTeam
- ShiftedExpandingMeanSeasonDrtgMeanTeam
- ShiftedExpandingMeanSeasonDrtgMaxTeam
- ShiftedRollingSum5SeasonDrtgMaxTeam
- ShiftedEwmSeasonBpmWeightedMeanTeam
- ShiftedExpandingMeanCareerBpmSumTeam
- ShiftedExpandingMeanSeasonBpmSumTeam
- ShiftedExpandingMeanSeasonBpmMedianTeam
- ShiftedExpandingMeanSeasonBpmMaxTeam
- ShiftedCumSumCareerBpmMaxTeam
- ShiftedCumSumSeasonBpmSumTeam
- ShiftedCumSumSeasonBpmMedianTeam
- ShiftedCumSumSeasonBpmMaxTeam
- ShiftedCumSumSeasonPlayerImpactEstimateSumTeam
- ShiftedCumSumSeasonPlayerImpactEstimateMeanTeam
- ShiftedExpandingMeanCareerNetPointsMeanTeam
- ShiftedExpandingMeanSeasonNetPointsSumTeam
- ShiftedExpandingMeanSeasonNetPointsWeightedMeanTeam
- ShiftedCumSumSeasonNetPointsSumTeam
- ShiftedCumSumSeasonNetPointsWeightedMeanTeam
- ShiftedRollingSum5SeasonNetPointsSumTeam
- ShiftedRollingSum5SeasonNetPointsWeightedMeanTeam
- ShiftedRollingSum9SeasonNetPointsWeightedMeanTeam
- ShiftedExpandingMeanSeasonPointsCreatedMaxTeam
- ShiftedExpandingMeanCareerBpmWeightedMeanTeam
- ShiftedExpandingMeanCareerBpmMedianTeam
- ShiftedCumSumSeasonBpmWeightedMeanTeam
- ShiftedExpandingMeanSeasonBpmWeightedMeanTeam
- ShiftedRollingSum3SeasonBpmSumTeam
- ShiftedRollingSum5SeasonBpmSumTeam
- ShiftedRollingSum9SeasonBpmSumTeam
- ShiftedRollingSum9SeasonBpmMaxTeam
- ShiftedExpandingMeanCareerApproximateValueMaxTeam
- ShiftedCumSumCareerApproximateValueMeanTeam
- ShiftedCumSumSeasonApproximateValueMeanTeam
- ShiftedCumSumCareerNbaEfficiencyRatingMeanTeam
- ShiftedExpandingMeanCareerGameScoreMeanTeam
- ShiftedCumSumSeasonGameScoreMeanTeam
- ShiftedExpandingMeanCareerWinScoreMeanTeam
- ShiftedExpandingMeanSeasonWinScoreMeanTeam
- ShiftedExpandingMeanSeasonWinScoreWeightedMeanTeam
- ShiftedCumSumCareerWinScoreMeanTeam
- ShiftedCumSumCareerWinScoreWeightedMeanTeam
- ShiftedCumSumCareerWinScoreMedianTeam
- ShiftedCumSumSeasonWinScoreSumTeam
- ShiftedCumSumSeasonWinScoreMeanTeam
- ShiftedCumSumSeasonWinScoreWeightedMeanTeam
- ShiftedCumSumCareerPlayerEfficiencyRatingMeanTeam
- ShiftedCumSumSeasonPlayerEfficiencyRatingMeanTeam
- ShiftedCumSumSeasonPlayerEfficiencyRatingMedia
- ShiftedExpandingMeanCareerPlayerImpactEstimate
- ShiftedExpandingMeanCareerPlayerImpactEstimate
- ShiftedExpandingMeanSeasonPlayerImpactEstimate
- ShiftedExpandingMeanSeasonPlayerImpactEstimate
- ShiftedCumSumCareerPlayerImpactEstimateMeanTeam

Figure A.2: Basketball features selected using *Boruta* algorithm

Feature	ξ_i
EloPts	0.452
Glicko2Mu	0.444
ShiftedExpandingMeanSeasonPythWins	0.440
TrueSkillMu	0.440
ShiftedSeasonLoses	0.438
ShiftedSeasonWIRatio	0.435
ShiftedSeasonWins	0.434
ShiftedCumSumSeasonWinScoreMeanTeam	0.431
ShiftedCumSumCareerWinScoreMeanTeam	0.431
ShiftedExpandingMeanSeasonNetPointsSumTeam	0.430
ShiftedCumSumCareerApproximateValueMeanTeam	0.427
ShiftedCumSumSeasonNetPointsSumTeam	0.426
ShiftedCumSumCareerGameScoreMeanTeam	0.426
ShiftedCumSumSeasonPlusMinusMaxTeam	0.426
ShiftedCumSumSeasonBpmMaxTeam	0.425
ShiftedExpandingMeanSeasonWinScoreSumTeam	0.425
ShiftedExpandingMeanCareerPlayerImpactEstimateMeanTeam	0.425
ShiftedCumSumSeasonNetPointsWeightedMeanTeam	0.425
ShiftedCumSumSeasonPlayerImpactEstimateMeanTeam	0.424
ShiftedExpandingMeanSeasonWinScoreMeanTeam	0.424

Table A.3: Basketball ξ_i correlation

- Season
- Month
- KSPDate
- Timestamp
- Attendance
- ShiftedEwmSeasonXt
- ShiftedExpandingMeanCareerXt
- ShiftedExpandingMeanSeasonXt
- ShiftedCumSumCareerXt
- ShiftedCumSumSeasonXt
- ShiftedRollingSum3SeasonXt
- ShiftedRollingSum5SeasonXt
- ShiftedRollingSum9SeasonXt
- ShiftedEwmSeasonXg
- ShiftedExpandingMeanCareerXg
- ShiftedExpandingMeanSeasonXg
- ShiftedCumSumCareerXg
- ShiftedCumSumSeasonXg
- ShiftedRollingSum3SeasonXg
- ShiftedRollingSum5SeasonXg
- ShiftedRollingSum9SeasonXg
- ShiftedEwmSeasonVaepConcedes
- ShiftedExpandingMeanCareerVaepConcedes
- ShiftedExpandingMeanSeasonVaepConcedes
- ShiftedCumSumCareerVaepConcedes
- ShiftedCumSumSeasonVaepConcedes
- ShiftedRollingSum3SeasonVaepConcedes
- ShiftedRollingSum5SeasonVaepConcedes
- ShiftedRollingSum9SeasonVaepConcedes
- ShiftedEwmSeasonVaepScores
- ShiftedExpandingMeanCareerVaepScores
- ShiftedExpandingMeanSeasonVaepScores
- ShiftedCumSumCareerVaepScores
- ShiftedCumSumSeasonVaepScores
- ShiftedRollingSum3SeasonVaepScores
- ShiftedRollingSum5SeasonVaepScores
- ShiftedRollingSum9SeasonVaepScores
- ShiftedEwmSeasonDefensiveVaep
- ShiftedExpandingMeanCareerDefensiveVaep
- ShiftedExpandingMeanSeasonDefensiveVaep
- ShiftedCumSumCareerDefensiveVaep
- ShiftedCumSumSeasonDefensiveVaep
- ShiftedRollingSum3SeasonDefensiveVaep
- ShiftedRollingSum5SeasonDefensiveVaep
- ShiftedRollingSum9SeasonDefensiveVaep
- ShiftedEwmSeasonOffensiveVaep
- ShiftedCumSumCareerOffensiveVaep
- ShiftedEwmSeasonPassesPerDefensiveActions
- ShiftedRollingSum3SeasonPassesPerDefensiveActions
- ShiftedRollingSum5SeasonPassesPerDefensiveActions
- ShiftedExpandingMeanCareerOffensiveVaep
- ShiftedExpandingMeanSeasonOffensiveVaep
- ShiftedCumSumSeasonOffensiveVaep
- ShiftedRollingSum5SeasonOffensiveVaep
- ShiftedRollingSum9SeasonOffensiveVaep
- ShiftedEwmSeasonVaep
- ShiftedExpandingMeanCareerVaep
- ShiftedExpandingMeanSeasonVaep
- ShiftedCumSumCareerVaep
- ShiftedCumSumSeasonVaep
- ShiftedRollingSum3SeasonVaep
- ShiftedRollingSum5SeasonVaep
- ShiftedRollingSum9SeasonVaep
- LeaguePoints
- Overall
- Attack
- Midfield
- Defence
- InternationalPrestige
- DomesticPrestige
- TransferBudget
- StartingXiAverageAge
- WholeTeamAverageAge
- BuildUpPlaySpeedValue
- BuildUpPlayPassingValue
- ChanceCreationPassingValue
- ChanceCreationCrossingValue
- ChanceCreationShootingValue
- DefencePressureValue
- DefenceAggressionValue
- DefenceTeamWidthValue
- EloPts
- TrueSkillMu
- TrueSkillSigma
- Glicko2Mu
- Glicko2Phi
- Glicko2Sigma
- HomeChanceCreationCrossingType
- HomeDefencePressureType
- HomeDefenceAggressionType
- AwayStartingFormationNameLine0
- AwayStartingFormationNameLine2
- AwayBuildUpPlayPassingType
- AwayBuildUpPlayDribblingType

Figure A.3: Soccer features selected using *Boruta* algorithm

Feature	ξ_i	Feature	ξ_i
Overall	0.290	ShiftedRollingSum9SeasonXg	0.241
ShiftedExpandingMeanCareerVaepScores	0.289	ShiftedCumSumSeasonVaepConcedes	0.243
ShiftedExpandingMeanSeasonVaepScores	0.285	ShiftedCumSumCareerXg	0.243
ShiftedExpandingMeanCareerXg	0.282	ShiftedExpandingMeanSeasonOffensiveVaep	0.241
Attack	0.280	ShiftedRollingSum9SeasonVaep	0.237
ShiftedCumSumSeasonVaepScores	0.280	ShiftedRollingSum9SeasonXt	0.236
EloPts	0.279	ShiftedRollingSum5SeasonVaep	0.234
TrueSkillMu	0.278	ShiftedEwmSeasonXt	0.233
Midfield	0.277	ShiftedRollingSum5SeasonXg	0.231
Defence	0.277	TrueSkillSigma	0.226
Glicko2Mu	0.275	ShiftedRollingSum5SeasonXt	0.224
TransferBudget	0.272	ShiftedEwmSeasonXg	0.223
InternationalPrestige	0.267	ShiftedRollingSum3SeasonVaep	0.223
ShiftedExpandingMeanCareerVaep	0.267	DefencePressureValue	0.221
ShiftedCumSumSeasonXg	0.265	DefenceAggressionValue	0.221
ShiftedExpandingMeanCareerXt	0.263	ShiftedRollingSum3SeasonXg	0.221
DomesticPrestige	0.261	ShiftedEwmSeasonVaep	0.221
ShiftedCumSumSeasonXt	0.261	ShiftedRollingSum5SeasonOffensiveVaep	0.21
ShiftedRollingSum5SeasonVaepScores	0.260	ShiftedRollingSum9SeasonOffensiveVaep	0.217
ShiftedEwmSeasonVaepScores	0.258	Attendance	0.217
ShiftedRollingSum3SeasonVaepScores	0.257	BuildUpPlaySpeedValue	0.217
ShiftedRollingSum9SeasonVaepScores	0.257	ChanceCreationPassingValue	0.216
ShiftedExpandingMeanSeasonXg	0.256	ChanceCreationShootingValue	0.215
ShiftedCumSumSeasonOffensiveVaep	0.255	DefenceTeamWidthValue	0.214
ShiftedCumSumCareerVaepScores	0.254	ShiftedExpandingMeanSeasonDefensiveVaep	0.213
ShiftedCumSumCareerOffensiveVaep	0.254	ChanceCreationCrossingValue	0.213
ShiftedCumSumSeasonVaep	0.253	BuildUpPlayPassingValue	0.212
ShiftedExpandingMeanCareerOffensiveVaep	0.252	ShiftedEwmSeasonOffensiveVaep	0.212
ShiftedCumSumCareerXt	0.252	ShiftedRollingSum3SeasonXt	0.211
ShiftedCumSumCareerVaep	0.251	ShiftedEwmSeasonPassesPerDefensiveActions	0.211
ShiftedExpandingMeanSeasonXt	0.250	ShiftedExpandingMeanCareerDefensiveVaep	0.210
ShiftedExpandingMeanSeasonVaep	0.245	ShiftedRollingSum9SeasonDefensiveVaep	0.209
ShiftedCumSumCareerVaepConcedes	0.243	HomeBuildUpPlayPassingType	0.209

Table A.4: Soccer ξ_i correlation


```
class BasketEloRanking(BasketRanking):
    def __init__(self,
                 EXP_BASIS: int = 10,
                 BASE_RATING: int = 1500,
                 ELO_WIDTH: int = 500,
                 K_FACTOR: int = 20,
                 BASE_MEAN_DIFF: int = 5,
                 YoY_ADJUST: bool = True,
                 MOV: bool = False,
                 HCA: str = "CST",
                 MOV_PARAMS: dict = {"NUM_PTS": 3, "NUM_POWER": .8, "DENOM_PTS": 7.5, "DENOM_MULTIPLIER": .006}
                 ) -> None: ...

    def fit(self, X: pd.DataFrame, y: pd.Series): ...

    def transform(self, X: pd.DataFrame, y: pd.Series) -> pd.DataFrame: ...

    def predict_proba(self, X: pd.DataFrame, y=None) -> np.ndarray: ...

    def _write_ratings(self, match_dict: dict,
                     elo_dict: dict,
                     seasons_dict: dict) -> None: ...

    def _write_forecasts(self, match_dict: dict, rating_dict: dict) -> None: ...

    def _update_rating(self, match_dict: dict, rating_dict: dict, games_dict: dict) -> None:
        elo_delta = self._compute_elo_delta(match_dict[Y_LABEL_],
                                           match_dict[EXP_OUTCOME_])
        elo_delta_multiplier = self._get_elo_delta_multiplier(match_dict)

        home_id = match_dict[HOME_ID_]
        away_id = match_dict[AWAY_ID_]

        rating_dict[home_id] += int(elo_delta * elo_delta_multiplier)
        rating_dict[away_id] -= int(elo_delta * elo_delta_multiplier)

    def _get_elo_delta_multiplier(self, match_dict: dict) -> float: ...

    def _compute_margin_of_victory(self, match_dict: dict) -> float: ...
```

Figure A.4: Code example: Basketball Elo ranking

```

class TeamsMetricsComputer(BasketMetricsComputer):
    """
    https://www.researchgate.net/publication/4985986\_A\_Starting\_Point\_for\_Analyzing\_Basketball\_Statistics
    """

    def __init__(self):...

    def fit(self, X: pd.DataFrame, y=None) -> 'TeamsMetricsComputer':...

    def transform(self, X: pd.DataFrame, y=None) -> pd.DataFrame:
        X_ = X.copy()

        X_[[f"{feature}OppTeam" for feature in _INDIV_STATISTICS]] = self._compute_opp_team_features(
            X_[[f"{feature}Team" for feature in _INDIV_STATISTICS]])

        for group_name, by in [_BY_TEAM_CAREER, _BY_TEAM_SEASON]:
            self._compute_vertical_agg(X_, _AGG_PLAYERS_STATISTICS + _TEAM_STATISTICS, group_name, by)

        self._compute_teams_metrics(X_)

        return X_[self.classes_]

    @staticmethod
    def _compute_opp_team_features(X: pd.DataFrame) -> pd.DataFrame:...

    def _compute_teams_metrics(self, X: pd.DataFrame) -> None:...

    @staticmethod
    def _compute_true_shooting_prc(pts: int, fga: int, fta: int) -> float:
        return pts / (2 * (fga + .44 * fta))

    @staticmethod
    def _compute_plays(fga: int, fta: int, tov: int) -> float:
        return fga + .44 * fta + tov

    def _compute_poss(self, tm_fga: int, tm_fta: int, tm_orb: int, opp_drb: int, tm_fg: int, tm_tov: int,
                     opp_fga: int, opp_fta: int, opp_orb: int, tm_drb: int, opp_fg: int, opp_tov: int) -> float:
        offensive_rebounds_prc = self._compute_rebound_prc(tm_orb, opp_drb)
        opp_offensive_rebounds_prc = self._compute_rebound_prc(opp_orb, tm_drb)
        tm_poss = tm_fga + .4 * tm_fta - 1.07 * offensive_rebounds_prc * (tm_fga - tm_fg) + tm_tov
        opp_poss = opp_fga + .4 * opp_fta - 1.07 * opp_offensive_rebounds_prc * (opp_fga - opp_fg) + opp_tov
        return .5 * (tm_poss + opp_poss)

    @staticmethod
    def _compute_rebound_prc(rb: int, opp_inv_rb: int) -> float:
        return rb / (rb + opp_inv_rb)

    @staticmethod
    def _compute_4_factor(shooting: float, turnovers: float, rebounding: float, free_throws: float) -> float:
        return .4 * shooting + .25 * turnovers + .2 * rebounding + .15 * free_throws

```

Figure A.5: Code example: Basketball teams metrics computation

```
@staticmethod
def _compute_herfindahl_index(pts: int, tm_pts: int) -> float:
    return (pts / tm_pts) ** 2

@staticmethod
def _compute_personal_foul_efficiency(stl: int, blk: int, pf: int) -> float:
    if pd.isnull(pf):
        pfe = np.nan
    elif pf != 0:
        pfe = (stl + blk) / pf
    elif stl + blk == 0:
        pfe = np.nan
    else:
        pfe = stl + blk

    return pfe

@staticmethod
def _compute_true_shooting_attempts(fga: int, fta: int) -> float:
    return fga + .44 * fta

@staticmethod
def _compute_points_produced(fga: int, fta: int, tov: int, off_rating: float) -> float:
    return ((fga + .44 * fta + tov) * off_rating) / 100

@staticmethod
def _compute_points_allowed(drtg: float, mp: float, tm_mp: float, tm_fga: int, tm_tov: int, tm_fta: int,
                             tm_orb: int) -> float:
    tm_poss = .96 * (tm_fga + tm_tov + .44 * tm_fta - tm_orb)
    return (drtg / 100) * ((.2 * (mp / (tm_mp / 5))) * tm_poss)

@staticmethod
def _compute_points_per_shoot(fg: int, _3p: int, fga: int) -> float:
    return (2 * (fg - _3p) + 3 * _3p) / fga if fga != 0 else np.nan

@staticmethod
def _compute_touches(fga: int, tov: int, fta: int, ast: int, tm_fta: int, opp_pf: int) -> float:
    if tm_fta == 0:
        tm_fta += 1
    if opp_pf == 0:
        opp_pf += 1
    return fga + tov + (fta / (tm_fta / opp_pf)) + (ast / .17)

@staticmethod
def _compute_pass_percent(ast: int, touches: float) -> float:
    return ((ast / .17) / touches) * 100 if touches != 0 else np.nan

@staticmethod
def _compute_shoot_percent(fga: int, touches: float) -> float:
    return (fga / touches) * 100 if touches != 0 else np.nan
```

Figure A.6: Code example: Basketball player metrics computation 1/2

```

@staticmethod
def _compute_fouled_percent(fta: int, touches: float, tm_fta: int, opp_pf: int) -> float:
    if tm_fta == 0:
        tm_fta += 1
    if opp_pf == 0:
        opp_pf += 1
    return (fta / (tm_fta / opp_pf)) / touches * 100 if touches != 0 else np.nan

@staticmethod
def _compute_tov_percent(tov: int, touches: float) -> float:
    return (tov / touches) * 100 if touches != 0 else np.nan

@staticmethod
def _compute_player_possession(fga: int, fg: int, fta: int, ft: int, ast: int, mp: float, tov: int,
                              tm_orb_prc: float, tm_ast: int, tm_mp: int, tm_fg: int) -> float:
    q = 5 * mp * tm_ast / tm_mp - ast
    r = 5 * tm_fg / tm_mp - ast if 5 * tm_fg / tm_mp != ast else 5 * tm_fg / tm_mp - ast - 1

    return fg - (fga + fta - fg - ft) * tm_orb_prc + .37 * fg * q / r + tov + .4 * fta

@staticmethod
def _compute_scoring_possession(fg: int, ast: int, ft: int, mp: int, tm_ast: int, tm_mp: int,
                                tm_fg: int) -> float:
    q = 5 * mp * tm_ast / tm_mp - ast
    r = 5 * tm_fg / tm_mp - ast if 5 * tm_fg / tm_mp != ast else 5 * tm_fg / tm_mp - ast - 1

    return fg - .37 * fg * q / r + .37 * ast + .5 * ft

@staticmethod
def _compute_non_scoring_possession(fga: int, fg: int, fta: int, tov: int):
    return fga - fg + .4 * fta + tov

@staticmethod
def _compute_player_impact_estimate(pts: int, fg: int, ft: int, fga: int, fta: int, drb: int, orb: int,
                                    ast: int,
                                    stl: int, blk: int, pf: int, tov: int) -> float:
    return pts + fg + ft - fga - fta + drb + orb / 2 + ast + stl + blk / 2 - pf - tov

@staticmethod
def _compute_tendex(pts: int, reb: int, ast: int, stl: int, blk: int, fga: int, fg: int, fta: int,
                   ft: int, tov: int, pf: int, mp: float) -> float:
    return (pts + reb + ast + stl + blk - (fga - fg) - .5 * (fta - ft) - tov - pf) / mp

@staticmethod
def _compute_win_score(pts: int, reb: int, ast: int, stl: int, blk: int, fga: int, fta: int, tov: int,
                       pf: int) -> float:
    return pts + reb + stl + .5 * (ast + blk - fta - pf) - fga - tov

```

Figure A.7: Code example: Basketball player metrics computation 2/2