

# Prediction of Future Values of Player Performance KPIs in Football

@SciSports

**Koen van Arem**

Supervised by Dr Floris Goes (SciSports),

and by Dr Jakob Söhl (TU Delft)

Mathematics of Data Science

TU Delft (EEMCS)

**June, 2024**

*A step towards the future of football.*



# Prediction of Future Values of Player Performance KPIs in Football

by

Koen van Arem

to obtain the degree of Master of Science  
at the Delft University of Technology,  
to be defended publicly on Friday June 28, 2024 at 14:00 PM.

Student number:	4959493
Project duration:	October 2, 2023 – June 28, 2024
Thesis committee:	Prof. dr. ir. M.B. van Gijzen, TU Delft
	Dr. J. Söhl, TU Delft, supervisor
	Dr. F. Goes-Smit, SciSports, supervisor
	Dr. R. J. Fokkink, TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.





## Abstract

The introduction of data-based modeling in football (soccer) in the last decade has led to the creation of models that describe player performance through key performance indicators (KPIs). However, relying solely on historical and current KPI values is insufficient for scouting departments, as predicting future values could significantly enhance transfer decision-making. This research aimed to identify the optimal model for forecasting the development of player performance KPIs over the next year, focusing on explainability, uncertainty quantification, and predictive performance.

To achieve this, we implemented linear models, tree-based models, and time-series-based kNN models to forecast two specific KPIs one year in the future: SciSkill, which measures the general quality of a player, and Estimated Transfer Value, representing the player's monetary value. Tree-based models showed the best predictive performance. The random forest in particular emerged as the best due to its explainable predictions, uncertainty quantification method based on bagging, and good predictive performance. In the Sciskill case study, the random forest model achieved low loss values, especially for young players. For the Estimated Transfer Value, the random forest model demonstrated the best predictive performance on the general set of players, and specifically on the subset of players valued at over €10 million.

Our findings suggest that tree-based models, particularly the random forest, are well-suited for predicting the future development of football player performance KPIs. Although it is important to monitor the predictive performance using the most recent data, the insights and the resulting models of this research can enhance scouting decisions via both data-informed and data-based decision-making. Finally, this research paves the way to study the influence of time series information or contextual information on player performance metrics.

## Acknowledgements

First and foremost, I would like to thank Dr. Floris Goes-Smit and Dr. Jakob Söhl for being my supervisors during this project. The skillful and personal guidance of Floris encouraged me to take into account the preferences of practitioners in my research and helped me to work in a multidisciplinary environment as a mathematician. I would also like to express my gratitude towards Jakob, for his enthusiastic and committed supervision, and for his structured feedback when needed.

I would like to express my gratitude to the SciSports for making this research possible. The data, tools, and knowledge available at this company created an ideal situation to work on the project. More specifically, I would like to thank the other members of the Data Analytics team for their willingness to think along and their enthusiasm about data science in football. I would also like to express my appreciation to everyone at SciSports for being great colleagues and for the enthusiastic games of padel.

I am grateful to Mirjam Bruinsma and Prof. Dr. Ing. Geurt Jongbloed for introducing me to football analytics and sports engineering, respectively. My appreciation also goes out to the TU Delft Sports Engineering Institute for involving me in exciting projects and events that encouraged me during my studies.

Furthermore, my appreciation goes out to my friends for their interest in the research and the joint study sessions. I am grateful to Sanne, who provided me with unconditional love and support during my studies. Finally, I would like to express my gratitude to my parents and brother, who always supported me to chase my dreams.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Literature review</b>	<b>5</b>
2.1	Player ratings . . . . .	5
2.1.1	Existing models . . . . .	7
2.1.2	Discussion of methods . . . . .	16
2.2	Transfer fee prediction . . . . .	20
2.2.1	Transfer and market value . . . . .	20
2.2.2	Existing models . . . . .	21
2.2.3	Discussion of the studies . . . . .	31
2.2.4	Economical context . . . . .	35
2.3	Predictive models . . . . .	37
2.3.1	Linear methods . . . . .	37
2.3.2	Tree-based methods . . . . .	40
2.3.3	Neural networks . . . . .	45
2.3.4	$k$ nearest neighbors . . . . .	48
2.3.5	Summary . . . . .	49
<b>3</b>	<b>Methods</b>	<b>53</b>
3.1	Data . . . . .	53
3.1.1	SciSkill . . . . .	53
3.1.2	Estimated Transfer Value . . . . .	54
3.2	Validation . . . . .	56
3.2.1	Train/test splits . . . . .	56

3.2.2	Choice of loss functions . . . . .	56
3.3	Models . . . . .	58
3.3.1	Linear models . . . . .	58
3.3.2	Tree-based models . . . . .	59
3.3.3	kNN models . . . . .	60
3.4	Hyperparameter tuning . . . . .	62
3.4.1	Adjusted time series cross validation . . . . .	62
3.4.2	Bayesian optimization algorithm . . . . .	63
<b>4</b>	<b>Results</b>	<b>67</b>
4.1	SciSkill case study . . . . .	67
4.1.1	Predictive performance on test set . . . . .	67
4.1.2	Predictive performance on age groups . . . . .	69
4.1.3	Predictive performance on subsets of outliers . . . . .	71
4.1.4	Summary predictive performance . . . . .	72
4.1.5	Predictions per year . . . . .	73
4.1.6	Feature importances . . . . .	74
4.2	Estimated Transfer Value case study . . . . .	75
4.2.1	Predictive performance on test set . . . . .	75
4.2.2	Performance on age groups . . . . .	78
4.2.3	Performance on subsets of outliers . . . . .	79
4.2.4	Summary predictive performance . . . . .	81
4.2.5	Predictions per year . . . . .	81
4.2.6	Feature importances . . . . .	82
4.3	Analysis time series adjustment cross-validation . . . . .	83
<b>5</b>	<b>Discussion</b>	<b>87</b>
5.1	Model quality . . . . .	87
5.1.1	Predictive performance . . . . .	87
5.1.2	Explainability . . . . .	89
5.1.3	Uncertainty quantification . . . . .	90
5.1.4	Determination of best model . . . . .	91
5.2	Used data . . . . .	92
5.3	Features . . . . .	93
5.4	Model training . . . . .	94
5.4.1	Hyperparameter optimization . . . . .	94

5.4.2	Feature selection . . . . .	96
5.4.3	Chosen models . . . . .	97
5.4.4	Validation methods . . . . .	100
<b>6</b>	<b>Conclusion</b>	<b>103</b>
<b>7</b>	<b>Applications and recommendations for future research</b>	<b>105</b>
7.1	Applications . . . . .	105
7.2	Recommendations for future research . . . . .	107
<b>Appendix A</b>	<b>Mathematical details</b>	<b>109</b>
A.1	Equivalence of RMSE and $R^2$ . . . . .	109
<b>Appendix B</b>	<b>kNN feature weight optimization</b>	<b>111</b>
B.1	Methods . . . . .	111
B.1.1	Requirements weighting methods . . . . .	111
B.1.2	Data . . . . .	112
B.1.3	Considered weighting methods . . . . .	113
B.2	Results . . . . .	119
B.2.1	Predictive quality . . . . .	120
B.2.2	Running time . . . . .	120
B.2.3	Interpretation . . . . .	121
B.3	Conclusion and discussion . . . . .	122
<b>Appendix C</b>	<b>Model specific results</b>	<b>125</b>
C.1	SciSkill case study . . . . .	125
C.1.1	Ordinary least squares . . . . .	126
C.1.2	Lasso . . . . .	129
C.1.3	Linear mixed effect models . . . . .	132
C.1.4	Decision tree . . . . .	134
C.1.5	Random forest . . . . .	137
C.1.6	XGBoost . . . . .	140
C.1.7	kNN . . . . .	144
C.1.8	kNN with Mahalanobis distance . . . . .	144
C.1.9	kNN with RReliefF weights . . . . .	145
C.2	Estimated Transfer Value case study . . . . .	146
C.2.1	Ordinary least squares . . . . .	146

C.2.2	Lasso . . . . .	149
C.2.3	Linear mixed effect model . . . . .	152
C.2.4	Decision tree . . . . .	154
C.2.5	Random forest . . . . .	156
C.2.6	XGBoost . . . . .	159
C.2.7	kNN . . . . .	163
C.2.8	kNN Mahalanobis . . . . .	163
C.2.9	kNN RReliefF . . . . .	164
<b>Appendix D Visualizations of hyperparameter tuning process</b>		<b>167</b>
D.1	SciSkill case study . . . . .	168
D.2	ETV case study . . . . .	182
<b>Appendix E Descriptions of data sets</b>		<b>197</b>
E.1	SciSkill case study . . . . .	197
E.2	Estimated Transfer Value case study . . . . .	202
<b>References</b>		<b>207</b>

# Introduction

In the evolving landscape of football analytics, the exploration of player performance has gained momentum, driven by an increased influx of data. Improvements in data-capturing technologies resulted in large data sets containing in-game data about football players, which provide the possibility to obtain more complex variables on player performance (Herold et al., 2019; Rein and Memmert, 2016). This increased amount of available data is reflected in the performed research as there has been a surge in the number of studies on player attributes and football performance analysis since 2012 (Principe et al., 2021; Wakelam et al., 2022). Therefore, the introduction of improved technologies has enlarged the research in football analytics in the last decade.

Due to the increased amount of data and research, new challenges emerged in football analytics. For example, Behravan and Razavi (2020) describe that the predominant challenge at present time lies in working with the complexity and huge amount of data itself. Next to this, football has a complex and dynamic nature compared to sports such as baseball which can easily be studied by making use of the simple and discrete nature of the games (Szczepański and McHale, 2015). The complex in-field nature of football makes it hard to determine the value of players for a team in a game (Yigit et al., 2020). These challenges create a field of study where more complex mathematical models can be utilized to deal with the abundance of data and the complex nature of the game.

Mathematical models have been created to obtain insight into different parts of the game of football. Models that value an on-the-ball action by estimating the differences in scoring ability are xThreat (Roy et al., 2020; Rudd, 2011) and VAEP (Decroos et al., 2019). Another model called xReceiver was introduced by Stöckl et al. (2021) to predict the likelihood of a player receiving the ball next. Such models result in numerical values describing in-field behavior or characteristics of football players. These numerical

values are key performance indicators (KPIs) and can be used to analyze the behavior and quality of football players and teams.

A possible use-case of these KPIs is aiding in transfer decisions for football clubs. As Szczepański and McHale (2015) and Aydemir et al. (2021) describe, recruitment can be distilled into two fundamental tasks: evaluating players' skills and estimating their value. Using knowledge about a player's performance, a club can estimate the worth of a player to their club and assess whether the estimated transfer value is worth the costs. With this knowledge, a club can make better-informed transfer decisions. Therefore, KPIs about both the quality of players and their costs are important to determine possible beneficial transfers.

However, player careers are often nonlinear and subject to randomness (Bergkamp et al., 2019; Wolf et al., 2021), and, therefore, it is not enough to have insight into the current values of these KPIs. In order to make a well-considered transfer decision it is insufficient to know the current values, but also needed to have insight into the future values of these KPIs. This is emphasized by He et al. (2023), Szczepański and McHale (2015), and Leifheit and Follert (2021) who state that it is necessary to obtain insight into the future values of these type of KPIs.

Nevertheless, predicting future football performance is a topic that has not received much attention thus far. Some studies exist about predicting future values of KPIs which studied the future values of the number of goals and assists (Apostolou and Tjortjis, 2019), the tier in which the player will be active in the English football system (Barron et al., 2018), and the market value (Baouan et al., 2022). But to the best of our knowledge, no prediction of future values of model-based KPIs has yet been conducted in the existing literature. This means that there is a lack of studies on the prediction of future model-based KPI values involving player performance and player values. The aim of this research, therefore, is to create models that predict future KPI values of model-based metrics.

It would be a straightforward choice to measure the quality of such a model using the predictive accuracy of a model. Nonetheless, this is not a comprehensive method to determine the quality of a model in the domain of football analytics as the domain of football is relatively expert-based. Because of this, explanations of the inner workings of models and explanations of their resulting predictions are important factors as these can give trainers and coaches more confidence in the model (Rossi et al., 2021). This means that the model should be explainable.

Moreover, the competitive environment of football has made transfer decisions increasingly important for both financial and athletic goals (Yigit et al., 2020). Simulta-



neously, progressively more money is being involved in transfers while large sums are spent on a small group of assets with high risks (McHale and Holmes, 2023). Because of the increased importance of transfer decisions with high risks, it is important to obtain insight into the uncertainty of the predictions of models.

A good prediction method is, therefore, defined in this thesis as a model with a high accuracy, good explainability, and methods to quantify the uncertainty of the predictions. The gap in the existing knowledge regarding the prediction of future model-based KPI values will be addressed by studying the research question of the current thesis: what is the best prediction model to predict KPI values one year ahead considering the accuracy, possibilities for uncertainty quantification of predictions, and the explainability of the models?

This study is an external thesis project at SciSports, a sports analytics company with a focus on football. Their main goal is to provide data-driven insights into football, and for this purpose, they have created several KPIs. One of their main branches is the Recruitment Center which provides data-driven insights to football clubs and player agencies on players for scouting and recruitment purposes.

As described, KPIs of the quality of a player and their financial values are essential to determine possible beneficial transfers when KPIs are used for scouting purposes. The SciSports Recruitment Center contains the model-based KPIs SciSkill, a player performance metric, and ETV, the Estimated Transfer Value. The SciSkill is a KPI for player performance based on match sheet data. Because it does not need the more detailed event data or tracking data, it can be calculated for a large amount of players. The ETV model estimates the expected value of a transfer fee given that a transfer would occur. The SciSkill and ETV together provide insight into the two most important things for recruitment: the quality of players and their transfer values. The research question will, therefore, be studied by doing case studies on the SciSkill and ETV.

This thesis consists of several chapters. First, the existing literature is studied in chapter 2. This will provide insights into the problem and will provide knowledge about the explainability and uncertainty quantification methods of the models. The predictive quality of the models will be studied using the case studies on the the SciSkill and Estimated transfer value. The methods of this case study will be described in chapter 3 and the corresponding results will then be given in chapter 4. Consequently, the results will be discussed in chapter 5, and the conclusion of the thesis will be given in chapter 6. Finally, applications and recommendations will be discussed in chapter 7.



## Literature review

The literature study first discusses literature on existing player ratings to obtain insights into the context, possible problems, and attributes. Thereafter, the literature on player valuation methods will be reviewed. Finally, possible statistical or machine learning methods will be discussed to obtain insight into corresponding uncertainty quantification methods and explainability of the models. It also provides knowledge of the workings and potential performance of this prediction problem.

### 2.1 | Player ratings

In recent years, the interest of the scientific community in football analytics has grown (Principe et al., 2021). As a part of this, a novel line of research was introduced that utilizes the increasing amount of data on football to rate individual players (Arntzen and Hvattum, 2020). New methods for quantification of player performance have been introduced such as VAEP (Decroos et al., 2019), PlayeRank (Pappalardo et al., 2019), and the SciSkill (SciSports, 2020). The application of these methods made it possible to create databases of player ratings, which then can be used to find interesting players for football clubs.

The determination of player performance was, historically, based on the expert judgment of the video data and statistics describing the frequency of in-game events (Memmert and Raabe, 2018, Ch. 1). The subjective expert judgment is, however, considered to be biased and inaccurate (Hvattum and Gelade, 2021; Pappalardo et al., 2019). With the introduction of these data-driven methods, the opportunity to reduce this bias is available. It is impossible to obtain a method with absolutely no bias, because, for example, biased decisions by managers are present in the division of playing time and

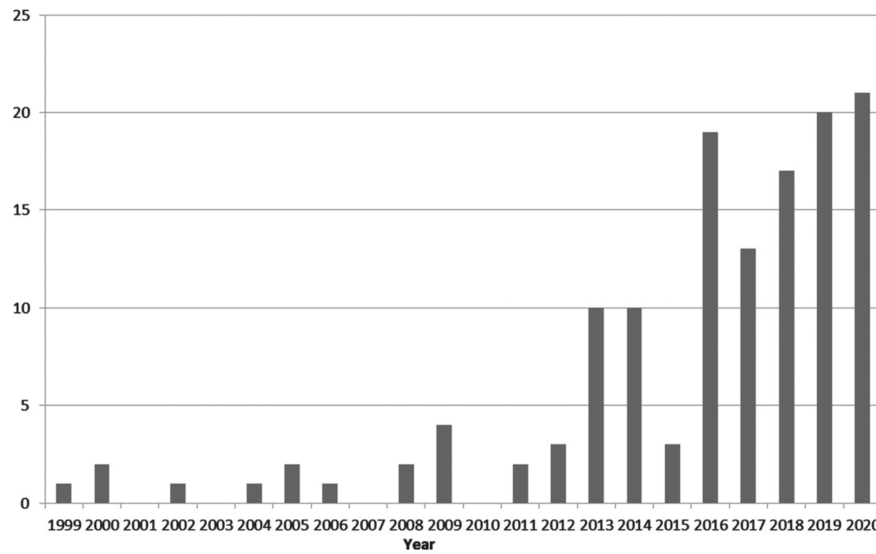


Figure 2.1: The number of papers published between 1999 and 2020 in the field of football analytics with a focus on eleven-a-side competitive professional football. (Adapted from Wakelam et al. (2022))

will influence the available data. Still, the introduction of data-driven methods provides the possibility to have more consistent scouting results.

The main problem of modeling player performance in football is the absence of one ground truth (Aydemir et al., 2021). This is due to the fact that a player can have different roles and that the performance of the player is dependent on the context (He et al., 2023). For instance, the player performance is dependent on the dynamics of the team. Moreover, defenders are less involved in goal scoring, a metric that is easily studied, than attackers. Depending on the playing style of a coach, a certain player can be a good fit or a terrible one. This makes it challenging to train comprehensive supervised models on the performance of players to obtain a model.

Several studies (Cefis and Carpita, 2022; Chazan-Pantzalis and Tjortjis, 2020; Matano et al., 2018; Nsolo et al., 2019; Sařabun et al., 2020) exist in which the ratings of experts were studied. Sařabun et al. (2020) and Nsolo et al. (2019) tried to apply supervised learning by considering existing ratings of players as ground truth. (Cefis and Carpita, 2022) and (Matano et al., 2018) are examples of such studies that use ratings from the video game FIFA/EA FC, which are based on ratings of a large number of scouts. Chazan-Pantzalis and Tjortjis (2020) tried to predict team performance based on player ratings obtained from the game Football Manager, which is also based on expert ratings. These studies try to create models that provide similar ratings as domain

experts. Nonetheless, biases in player ratings of expert-based ratings are continued by considering expert ratings as the ground truth.

With the introduction of data-driven methods, the opportunity to reduce this bias is presented. Additionally, data-driven models provide the possibility to have more consistent scouting results as expert ratings in football can be inconsistent (Chawla et al., 2017). This study will, therefore, only consider data-driven player rating models. In this section, the individual existing models are first discussed followed by a discussion of these methods.

### 2.1.1 | Existing models

As there is no ground truth for player performance, there exist different interpretations of player performance. Link et al. (2016) created a model describing player performance based on a model obtained via domain knowledge. Pappalardo et al. (2019) considered expert opinion as ground truth and Aydemir et al. (2021) inspected the quality of their model using player transfer values and the quality of teams in transfers. As the ultimate goal of a football game is to win by scoring goals, other models consider the influence of a player on the scoring probability (Decroos et al., 2019; Kharrat et al., 2017) or winning probability (Hvattum and Gelade, 2021) as the dependent variable. A good player is then defined as one that increases the scoring or winning probability. In these ways, it is possible to estimate the quality of a player over a match or season.

#### 2.1.1.1 | Dangerousity approach

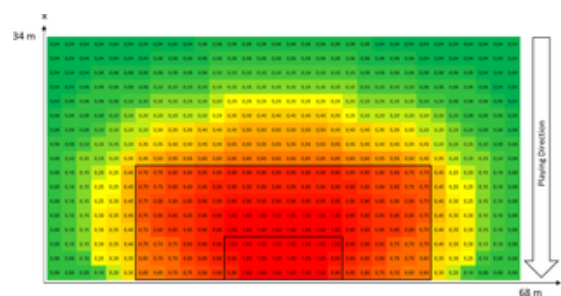


Figure 2.2: The Zone (ZO) in the Dangerousity approach in a grid of  $2 \times 2$  meters beginning 34 meters from the goal line. (Adapted from Link et al. (2016))

The Dangerousity approach was introduced by Link et al. (2016) and describes offensive behavior. The value of the Dangerousity was derived via the mathematical description of domain knowledge. The authors define a value for the Dangerousity at a

moment  $t$  by

$$DA(t) = ZO(t) \times \left( 1 - \frac{1 - CO(t) + PR(t) + DE(t)}{k} \right), \quad (2.1)$$

where  $ZO(t)$  is the zone, which represents the danger of a goal being scored from the position of a player in ball possession as shown in Figure 2.2. The other quantities are the Control ( $CO$ ), Pressure ( $PR$ ), and Density ( $DE$ ), and a constant  $k$ . This danger of the zone is then multiplied by a factor that reduces it with at most a factor of 0.5 based on the other quantities. The Control describes the extend to which the player in ball possession can control the ball. The Pressure is a measure of to what extent the defending team applies pressure to the ball-possessing player. The Density consists of a weighted average of the Shot Density, which describes the ability of defenders to block a shot, and the Pass Density, which describes the extent to which defenders can block a pass aimed at the area in front of the goal called the Interception Zone. The idea behind the definition of Dangerousity in Equation 2.1 is that the danger of scoring can be decreased by applying pressure and the ability of defenders to block the ball. A good ball control, on the other hand, gives a higher chance of performing the desired action and increases the multiplication factor. In this way, Link et al. (2016) defined principle of Dangerousity.

An important part of the Dangerousity approach is that the factors like Control, Pressure, and Density mathematically describe concepts of the domain knowledge. The Control, for instance, is defined using the relative velocity of the ball with respect to that of the player in ball possession using  $CO = 1 - k_2 \times v_{rel}^2$ , where  $k_2$  is a constant and  $v_{rel}$  the relative distance between the ball and the ball playing player. In similar ways, the other factors like Control and Pressure were defined.

The resulting model describing the Dangerousity can be used to determine the Dangerousity before and after an action. The value of an action can be determined by calculating the increase in Dangerousity as a good action increases the Dangerousity. The player's performance can then be determined by summing over the values of the actions performed by a football player.

#### 2.1.1.2 | Risk-reward approach

Power et al. (2017) introduced a combination of two logistic models to assess the quality of passes in football. First, they trained one model to estimate the probability of a pass being successfully completed. To this end, they included features created to describe the situation with principles that are considered important based on domain knowl-

edge, such as the speed of a player. As it is often unknown what the intention was of an unsuccessful pass, they needed to determine the expected receiver. For each teammate of the player in ball possession, the relative distance ('Distance') and the relative angle ('Angle') were calculated. The expected receiver was then taken as the one with the maximal value of  $\frac{\text{Distance}}{\text{Min Distance}} \times \frac{\text{Angle}}{\text{Min Angle}}$ , where 'Min Distance' and 'Min Angle' are the minimal values of 'Distance' and 'Angle' respectively. Using the resulting data, they trained the model to describe the probability of the pass being completed, called the Risk model.

Additionally, they trained a model to define the possible reward of a pass. This was done by estimating the probability that the pass made would result in a shot within the next 10 seconds. As a shot does not often occur, their training set was unbalanced. This was also reflected in the fact that their model only slightly improved the RMSE compared to naively assigning the average probability of a shot occurring. This resulted in a model that was called the Reward model as it described the probability of scoring soon after a pass.

In their research, Power et al. (2017) trained both models on multiple feature sets and they found that features derived from tracking data of the football players during a match are of added value. Combined with information about the formation of the teams describing, for instance, a high defensive block, the tracking data resulted in the best loss values on the test set.

These two models were then used to introduce new metrics such as Passing Plus Minus, which describes the difference between the expected number of passes completed using the Risk model and the actual number of passes completed. A player with good passing ability can be described as completing more passes than expected. They also described a player performance metric that describes the quality of a player to receive passes. This is done by taking  $1 - P(\text{pass completed})$  and summing over these values for the passes received. Similarly, the pass executor and pass receiver metrics were introduced counting the number of dangerous passes, which are passes that are within the top 75% of the passes with the most reward according to the model. In this way, they used the two models describing the risk of a pass and the reward of a pass to determine player quality.

### 2.1.1.3 | VAEP

The Valuing Actions by Estimating Probabilities (VAEP) framework was introduced by Decroos et al. (2019). It describes a model that estimates the probability of scoring

using supervised learning methods like the initially implemented CatBoost algorithms. A VAEP model considers the current state of the game,  $S_i$ , as the last 3 actions. It then estimates the probability that a goal is scored or conceded in the next 10 actions and denotes this by  $P_{scores}(S_i)$  and  $P_{concedes}(S_i)$ . The offensive and defensive values are then defined as the difference in scoring probability caused by a given action  $x$ ,  $\Delta P_{scores}(S_i, x) := P_{scores}(S_i, x) - P_{scores}(S_{i-1}, x)$  and  $\Delta P_{concedes}(S_i, x) := P_{concedes}(S_i, x) - P_{concedes}(S_{i-1}, x)$  respectively. The value of an action is then defined as the sum of the offensive and defensive value,  $V(a_i, x) := \Delta P_{scores}(S_i, x) + (-\Delta P_{concedes}(S_i, x))$ . The final player rating is obtained by summing the action values of a player and rescaling it for the number of minutes played. In this way, a VAEP model obtains a performance KPI by assigning value to each action in a game.

Several studies have been conducted to improve the quality of the model. Decroos and Davis (2020) studied the accuracy and interpretability of logistic regression, XGBoost, and GAMs. They found that XGBoost and GAMs outperform logistics regression, but that the difference in performance between XGBoost and GAMs is relatively small. They conclude that it is better to use GAMs because these are generally better interpretable. Van Haaren (2021) subsequently tried to improve the explainability of the model. He did this by using fuzzy assignments to pitch zones which are used by practitioners. In this way, he tried to make it more explainable to football staff. These changes improved the explainability and interpretability of a black-box model.

#### 2.1.1.4 | xThreat

A similar and more explainable model exists in the form of xThreat models (Roy et al., 2020; Rudd, 2011). An xThreat model estimates the probability of scoring similarly to VAEP but does this in a more simple and explainable way. xThreat models the game of football as a Markov chain. This means that it assumes that the future is only dependent on the current state of the game and not on what happened in the past. This is realized by defining  $S_i$  as the current situation, where the current state is defined as the position of the ball-possessing player on the field. Using this, it is possible to estimate the probabilities  $P_{scores}(S_i)$  using an iterative method, where  $P_{scores}(S_i)$  describes the chances of scoring before losing ball possession instead of scoring within the next 10 actions. Roy et al. (2020) compared the xThreat and VAEP models and stated that, although the xThreat model is more interpretable to practitioners, it can only take into account the position of an action and excludes contextual information such as the position of defenders from its model. This makes it more interpretable, but less accurate. van Arem



and Bruinsma (2024) extended the xThreat model by including variables describing the defensive situation and height of the ball beside the position of the ball-possession on the field. This extended xThreat model can better differentiate between more different situations, although VAE models can take into account more contextual information. Dependent on the context one model can be preferred over the other.

### 2.1.1.5 | Plus-minus rating

Plus-minus ratings are a different type of approach that was extensively studied in ice hockey and basketball (Kharrat et al., 2017) before being applied to football by Hvattum and Sæbø (2015). The idea behind the method is that every game of football is partitioned into segments with every change in lineup creating a new segment. In this way, a segment is a period for which the lineup was the same. For each segment, it is denoted which players are actively playing by defining the variable  $x_{tj}$  for segment  $t$ , and player  $j$  as

$$x_{tj} = \begin{cases} 1 & \text{if player } j \text{ plays for the home team in the segment} \\ -1 & \text{if player } j \text{ plays for the away team in the segment} \\ 0 & \text{if player } j \text{ does not play in the segment} \end{cases}.$$

Regularised linear regression is then applied with  $x_{tj}$  as independent variables and the score difference in the segment as the dependent variable. The value  $\beta_j$  of the coefficient corresponding to each player can then be viewed as a value of the quality of a player. If a player's coefficient is positive, the player is associated with a positive goal difference, whereas a negative coefficient would mean that the team of the player generally concedes more goals than it scores with this player in the field.

In contrast to basketball and ice hockey, the number of substitutes is limited in football, and a goal is a relatively rare event. There are, therefore, fewer segments and the changes in goal differences during a segment are often the same. This causes the models trained on football data to be less accurate and several solutions have been studied for this problem. Kharrat et al. (2017) changed the dependent variables for the PM ratings. They trained two models, one with the expected goals (xG), an estimation of  $\mathbb{E}[\text{Number of Goals}]$ , created in the segment, and one with the expected points in a competition (xP), an estimation of  $\mathbb{E}[\text{Points per game}]$ , obtained during the segments. Both xP and xG provide more distinct values for the different segments compared to the goal difference, which only has integers as possible values. This means that there is more differentiation between the quality of play of a segment, making it possible

for the model to catch more fine distinctions in player quality. Hvattum and Gelade (2021) similarly used VAEP-values created in one segment as the dependent variable, but this did not provide improvement over the normal PM ratings. Pantuso and Hvattum (2020) presented a model that improved the normal PM ratings by taking age, red cards, and home advantage into account. Their model had such an increased complexity that it could not be viewed anymore as regularised regression but as a quadratic programming problem. Hvattum (2020) discussed how the ratings in this model could be split up into a defensive and offensive part. In these ways, the PM ratings have been adapted to the sport of football.

Kharrat et al. (2017) highlighted that PM ratings have problems with distinguishing the quality of players that often play together. This is caused by these players having similar segments, which results in similar values in the performance values. This inability to distinguish players who often play together is a downside of the PM ratings.

#### 2.1.1.6 | Elo rating

The Elo-rating is a model originally made for one-versus-one setting, attributed to chess Grand Master and physicist Arpad Elo (Sullivan and Cronin, 2016). It was later adapted for application in team ratings by Hvattum and Arntzen (2010), Hubáček et al. (2019), and Sullivan and Cronin (2016). The Elo-ratings were later adjusted to obtain performance ratings for individual players by Wolf et al. (2021) and Aydemir et al. (2021).

The main assumption behind Elo-ratings is that a better team is more likely to win the game. According to Sullivan and Cronin (2016), the Elo rating defines the outcome value  $S$  of a match as

$$S = \begin{cases} 1, & \text{for a win} \\ 0.5, & \text{for a draw} \\ 0, & \text{for a loss} \end{cases} \quad (2.2)$$

Let the Elo-rating of team  $T$  be denoted as  $R$ . The player rating is then updated after every match using the formula

$$R_{\text{post game}} = R_{\text{pre game}} + K(S - \mathbb{E}[S]), \quad (2.3)$$

where  $K$  is a constant describing the weighting of the new results. This means that the rating is adjusted by the difference between the outcome and the expected outcome. If a team wins, while the team is expected to win, the rating is not changed much. On the other hand, if a team has lost when it was expected to win with high probability, the

new rating is decreased more firmly. The expected outcome of team  $A$  winning against team  $B$  is computed by

$$\mathbb{E}[S_A] = \frac{10^{R_A/400}}{10^{R_A/400} + 10^{R_B/400}}.$$

In this way, the Elo-rating for teams is computed based on the differences between the expected outcome and the real outcome.

Wolf et al. (2021) revised the existing algorithm to obtain Elo-ratings for individual players. Their model considers the segment of the game that a player was on the pitch similar to PM ratings. For this segment, the rating of the team ( $A$ ) was calculated using

$$R_A = \frac{\sum_{i=0}^{n-1} R_{A_i} \cdot M_{A_i}}{\sum_{i=0}^{n-1} M_{A_i}},$$

where  $A_1, \dots, A_{n-1}$  are the  $n$  players that played in the time segment,  $M_{A_i}$  is the number of minutes played by  $A_i$  and  $R_{A_i}$  is the rating for player  $A_i$ . This quantity can be seen as the average player ranking on the field weighted by the number of minutes played. Also the result  $S$  is defined slightly differently to adjust it for the application on segments. They divided the game into segments similar to those of the PM ratings to adjust Equation 2.2 to estimate the individual player quality. They then added weights in their new expression for  $K(S - \mathbb{E}[S])$ , scaled the differences for the number of minutes played by a player, and included the rating of the team itself. Additionally, they included the difference between the rating of a player with the rating of the team as an extra weight. This means that a bad player will be punished less when the team performs badly. In this way, the new Elo-rating was adjusted for the application in football.

Elo ratings have a downside similar to that of the PM ratings. According to Wolf et al. (2021), the Elo ratings punishes good players who play at clubs performing below average. As it is of interest for scouting purposes to find good players at clubs performing below average, this is a considerable downside and the player Elo ratings might not be well-suited for scouting these types of players.

### 2.1.1.7 | PlayeRank

Pappalardo et al. (2019) introduced the algorithm PlayeRank, which consists of both supervised and unsupervised methods to obtain a ranking model. First, they detected 8 different player roles by applying a k-means algorithm with soft clustering to the average positions of players. They then created 76 different features that they extracted from event data such as the number of accurate shots and number of cross-passes that

were an assist. The importance of each feature was subsequently extracted by training a linear support vector machine on the problem of predicting whether a match is won or not. The ratings obtained by their model are of a linear form. For player  $u$  in match  $m$ , it defined as  $r(u, m) = \frac{1}{R} \mathbf{w}^T \mathbf{x}$  where  $\mathbf{x}$  is a vector containing all features,  $\mathbf{w}$  is a vector of weights and  $R$  is a scaling factor such that all values are in  $[0, 1]$ .

After training the model, they then tested the quality of the model with a data set labeled by scouts. In this way, they were able to compare it with Flow Centrality (FC), a KPI introduced by Duch et al. (2010) defined by the fraction of the times a player is part of a passing chain that ends in a shot, and Pass Shot Value (PSV) (Brooks et al., 2016), a KPI that is obtained via a machine learning model that predicts whether a pass will generate a shot. They compared the methods based on the agreement with expert opinions of scouts, by letting scouts and the different models determine the quality of a pass. The concordance between the scouts and the models is shown in Figure 2.3. The results show that PlayeRank achieves better agreement with the scouts' opinions. Additionally, they found that experts do not always agree with each other highlighting the subjectivity of these ratings and they found that they often provide biased and inaccurate estimates. Still, this validation method might increase the confidence of scouts in the PlayeRank method.

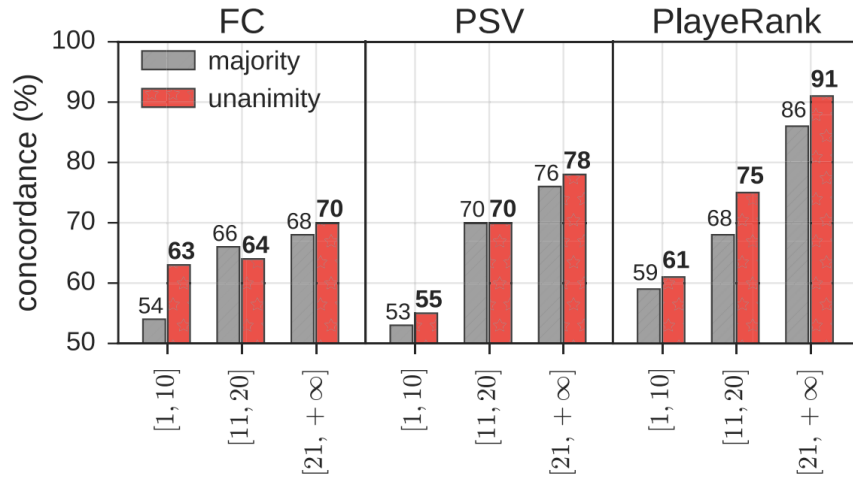


Figure 2.3: Majority and unanimity concordance between FC and scouts, PSV and scouts, and PlayeRank and scouts. (Adapted from Pappalardo et al. (2019))

### 2.1.1.8 | Dimension reduction approach

Another unsupervised player performance method is the dimension reduction approach introduced by Aydemir et al. (2021). Their method was applied to identify the quality of 3,500 fullbacks. They created features based on player match statistics, game difficulty, and competition quality based on Elo ratings of teams. After rescaling the features, they applied cosine kernel PCA to obtain a player ranking method. The cosine kernel was selected to eliminate the orthogonality of the components as this would make the PCA less suited for ranking problems. In this way, they applied the unsupervised PCA method to obtain player ratings.

Although Aydemir et al. (2021) admit that the absence of a ground truth makes the validation of methods nontrivial, they attempt to validate their model by implementing other methods for multiple-criteria decision-making (MCDM) problems called COMET, earlier introduced in football on a small scale by Sařabun et al. (2020), and CPP-Tri, a probabilistic sorting method introduced by Sant'Anna et al. (2015). Additionally, they considered random allocation as a baseline method. These methods were then compared by selecting the market valuation of a player as the ground truth. They concluded based on visual interpretation of results and a two-sample Kolmogorov-Smirnov test that their dimension reduction approach was a better indicator of the market value. Next to that, they validated the method by studying transferred players with the Elo ratings of the destination club. They found a significant relationship that showed that players with a good PlayeRank transferred to clubs with a high Elo rating. With these two relationships, they tried to validate the workings of PlayeRank.

Because of the unsupervised nature of the PlayeRank algorithm, it is hard to interpret the results of the model. This makes the intuition behind the algorithm less explainable to practitioners, while this is a crucial part of football analytics. Next to that, the application of PCA transforms the meaningful features that were used as input to several, for which the interpretation is unclear. Therefore, the unsupervised nature of PlayeRank algorithm makes it hard to interpret what it describes.

### 2.1.1.9 | SciSkill

As a last method, the SciSkill is discussed, a player rating algorithm provided by SciSports (2020). The SciSkill is a player rating algorithm with online updates similar to Elo ratings as shown in Figure 2.4. There are three main differences between the SciSkill and the Elo algorithm. First, the SciSkill model predicts the number of goals for each team in a match, whereas the Elo algorithm predicts the winning team. Secondly, SciSkill

makes use of a defensive and attacking rating to predict the expected number of goals in a match, similar to the split in the defensive and offensive PM ratings obtained by Hvattum (2020). Third, the prediction algorithm for the expected number of goals is an expectation-maximization algorithm which is more powerful than the rough estimate as defined in Equation 2.3. With these differences, the SciSkill algorithm can obtain more detailed rankings.

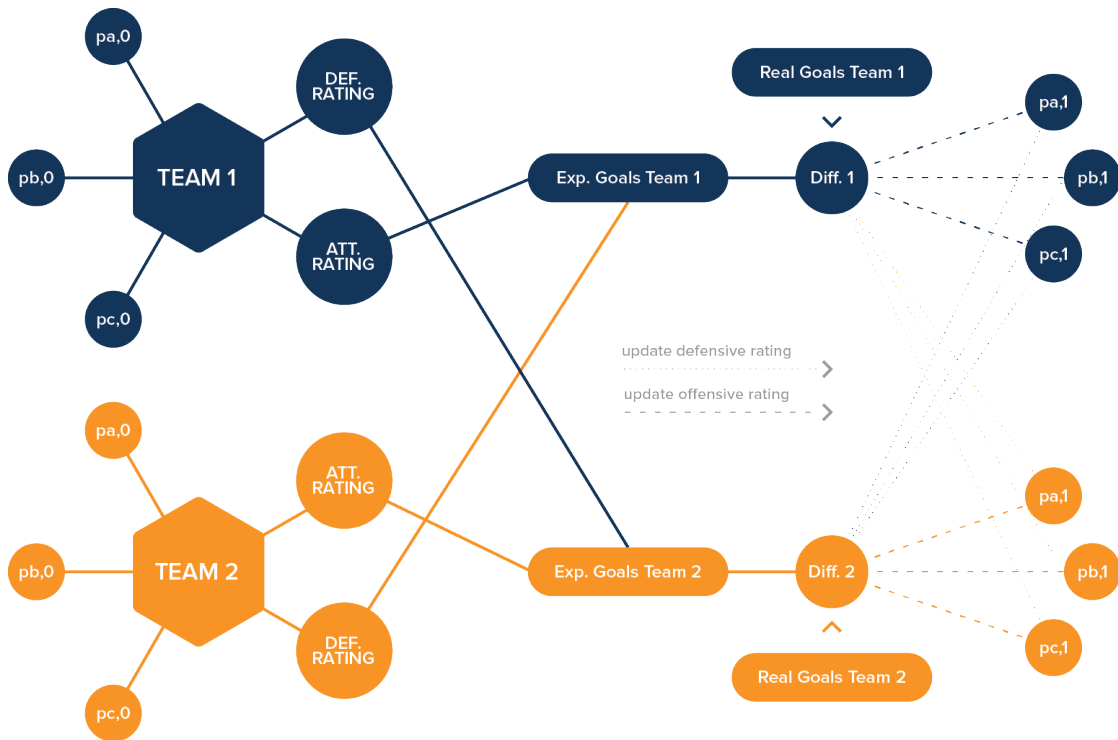


Figure 2.4: A graphical representation of the SciSkill algorithm. (Adapted from SciSports (2020))

## 2.1.2 | Discussion of methods

Now that we have discussed important existing methods for player rating, it is possible to consider some overarching findings from the literature that provide extra insight or context to the project.

### 2.1.2.1 | Comparison of methods

Due to the lack of ground truth, it is a challenge to compare different player rating methods. Still, different studies did perform comparisons to validate their methods.

For instance, Arntzen and Hvattum (2020) compared Elo ratings of teams and PM ratings of individual players. They found that PM ratings have more predictive value than team Elo ratings by using a linear model and a competing risk model. They found that their PM ratings contained better predictive quality. Hvattum and Gelade (2021) similarly studied the predictive qualities of VAEP and PM ratings by considering the predictive accuracy of a logit regression model that predicts the outcome of a match with the player ratings as input. Additionally, they studied the robustness by splitting the data set into smaller parts and training the model on these parts. A model is then considered robust if the ratings do not differ much between the different parts, and were, therefore, not susceptible to randomness in the data.

Their results showed that the PM ratings were more robust and had a better predictive performance. However, the VAEP model is a nonlinear model and, therefore, might favor different variables due to nonlinearities. On the other hand, the PM model is a linear model and approximates the real relation with the best-fitting linear relation. This best-fitting linear relation provides the information in a pattern that is the assumed relation for the model that measures the predictive quality because this is also a linear model. Consequently, the PM ratings can be expected to perform better, independent of the informational quality, which means that this comparison method has a bias towards linear models.

Instead of studying predictive information in the KPIs, Pappalardo et al. (2019) compared the labels with the concordance with scouts. Aydemir et al. (2021) compared ratings with market values of players to determine the quality of their model. Additionally, they studied the Elo ratings of destination clubs of player transfers. These studies used substitute quantities for the player quality to examine the model quality, despite the fact that it is debatable what substitute quantities should be used. It can be concluded that there is no unbiased and objective method to compare the quality of different player performance models.

### 2.1.2.2 | Lack of studies on future KPI values

The existing models in this literature study were constructed to estimate the current quality of a player. As stated in chapter 1, there is a potential in the usage of future values as it provides insight into how a possible player might develop over time, which is an important factor in transfer decisions. To the best of our knowledge, there are only two studies that studied predicting future values of player performance KPIs.

In the first study, Apostolou and Tjortjis (2019) tried to predict the number of goals

in the next season of two attackers: Lionel Messi and Luis Suárez. Next to that, they also predicted the number of shots in the next match for only Lionel Messi. Due to the limited amount of players and the fact that these players have been consistent in player performance, it is not possible to generalize results from this study.

The second study was performed by Barron et al. (2018) who tried to predict whether at what tier in the English football system football players would be active next season using artificial neural networks. Although the tier of a player gives more information about the level of an average player than the number of goals, it is a KPI with minimal knowledge gained as it attains only three values.

This means that the first study does not have generalizable results due to the limitations of their method and the other study considers a KPI that cannot differentiate between players in the same league. It can, therefore, be concluded that there is a lack of knowledge of the future values of more complex and informational KPIs in the existing literature.

### 2.1.2.3 | Biases in current studies

As discussed in chapter 1, expert judgment in football was found to be biased, but there also exist biases in current studies, both in the type of studies but also in the data set on which the studies are performed.

The current models are often better at describing offensive quality. This is caused by the fact that the number of goals or xG are widely used metrics, but these mostly give direct information on attacking quality. This is also reflected in the fact that KPIs concerning assists, number of key passes, and accuracy of shots were the most important in Pappalardo et al. (2019), which are all KPIs describing attacking quality. The same holds for xThreat and VAEP models which quantify the quality of on-the-ball actions, which are often offensive actions. Next to that, some studies only consider attacking players such as Saġabun et al. (2020) and Apostolou and Tjortjis (2019). These studies contain a bias towards attacking players. It can be concluded that the current knowledge about the quality of football players is biased towards attackers as also described by Chazan-Pantzalis and Tjortjis (2020).

This bias might lead to a worse valuation of more defensive players. Due to the simplicity of the input data, the bias towards attackers is less apparent in Elo ratings and SciSkill ratings. A study on such KPIs would, therefore, be important to reduce the current bias in the knowledge about football rating systems.



#### 2.1.2.4 | Trade-off between detailed data and player set size

The last finding in the existing literature is the relation between the complexity of an algorithm and the type of data as input. The methods discussed use event data or match sheet data. Event data contains annotations of all actions and events that occur during a football game. Match sheet data on the other hand is much more high level and only contains basic information about the goals, cards, and substitutions.

Methods such as VAEP, xThreat, PlayeRank, and the dimension reduction approach use event data to train the models and predict values. With this, more detailed information can be used to assess the quality of a player. This makes it probable that these methods can model more subtle differences in player qualities. Nonetheless, this extra data might enhance the bias towards attacking players as it mainly contains on-the-ball actions. Methods such as PM ratings, Elo ratings, and SciSkill ratings only use match sheet data. Gelade and Hvattum (2020) tried to combine match sheet and event data in PM ratings, but found minimal improvement in the model. The methods can, therefore, oftentimes be categorized as either event-based or match sheet-based.

Whereas event-based methods can assess the quality of players with more details, match sheet data is generally available for significantly more games. This makes it possible to include many more competitions when using match sheet-based methods. As assistance in player scouting is the main aim of rating methods, the inclusion of more players is an important advantage of match sheet-based methods. Next to that, more complex prediction methods for future values can be applied to the player rankings, if more players are included as more training points are available. Because of the focus of the current thesis on the study of prediction methods for player performance, it is beneficial to have larger model coverage resulting in a larger data set for the research. Thus, match sheet methods such as SciSkill are better suitable as a player performance metric for the research in the current thesis.

#### 2.1.2.5 | Summary

To summarize, there exist different player performance models and it is still an open challenge to rigorously compare the quality of the different models. The models are based on match sheet or event data of which the first is preferred for this study. There is a bias in the existing models as the event-based models can better describe offensive actions than defensive actions. And most importantly, there is a lack of knowledge on the prediction of future values of player performance KPIs.

## 2.2 | Transfer fee prediction

Another KPI that is considered in this thesis is the expected transfer value (ETV) of a player. This KPI can be used to make transfer decisions. It can, for instance, be used to determine whether a player could be bought and whether the transfer fee demanded by a football club is a reasonable price. This section covers studies related to the prediction of the transfer fee of football players.

First, some necessary context is provided. In the transfer system, clubs are allowed to buy or sell players subject to regulations imposed by football associations (Payyappalli and Zhuang, 2019). Such regulations include that players can only be transferred during transfer windows and that a transfer can either be permanent or temporary, in which case it is called a "loan". With a permanent transfer, it is common that a transfer fee is paid by the buying club. The monetary value of this transfer fee is based on the value of the player for both clubs and historical transfers of similar players (Poli et al., 2021).

### 2.2.1 | Transfer and market value

Whereas there is no data on the ground truth in player performance, there is data on the ground truth for transfer fees. When a player is bought, a transfer sum is paid by the buying club. The value of these fees is often not disclosed and is often estimated by media such as Transfermarkt.de. These estimated values are samples of the distribution describing the the ground truth, which makes it possible to train models to predict these values. These monetary values are, for instance, used in transfer and salary negotiations according to Herm et al. (2014).

The values resulting from these predictive problems are called transfer values in this research. Poli et al. (2021) defined these transfer values as the fee that an engaging team is willing to agree with the releasing team as compensation for breaching the contract of a player. This value also takes into account the situation of a player such as his current club and the contract length. The transfer value, thus, gives the monetary value given the occurrence of a transfer of the player at that specific moment.

The market value is a similar KPI describing the monetary value of a football player. Herm et al. (2014) defined this as an estimate of the fee a club would be willing to pay to sign a player independent of the actual transaction. This means that the current club and the contract length are not taken into account for this KPI. Although the transfer value and market value have this subtle difference, market values are often used as a

substitute for transfer fees (Franceschi et al.).

Franceschi et al. also studied the relation between the crowd-sourced transfer values obtained from Transfermarkt.de and the real transfer fees that were released by the Bundesliga. They found that there is a strong linear correlation with a significant slope of 0.95 and an  $R^2$  of 0.90. Therefore, market values have a strong relation with transfer fees and studies on transfer values will be included in this review.

In this section, several types of models used for transfer fee prediction will be discussed for an overview of the existing methods and the behavior of the problem. Thereafter, important overarching findings in the literature are discussed to consider differences within existing literature and obtain insight into possible problems for this research.

## 2.2.2 | Existing models

The estimation of the value of a player has been an important line of research that has developed over the last two decades. Frick (2007) states that increasingly more information about transfer fees has become available and many studies have been performed on this data. The first studies on player value were performed building on economical principles (Franceschi et al., 2023). These were often based on basic analytical methods. Later studies followed a more mathematically oriented approach with many studies using linear regression models. More recently, predictive modeling using machine learning methods has been emerging (Aydemir et al., 2022).

### 2.2.2.1 | Linear models

In an overview article, Franceschi et al. (2023) considered 29 studies containing 111 models on the transfer fee or market value estimation. A significant amount of these models, 94 out of 111, were ordinary least squares (OLS) models. The OLS models were mainly used to find which variables have a significant linear dependence on the value of a player. The differences in the linear models can mainly be found in the chosen independent variables and the inclusion of possible interaction terms. Many different sets of independent variables have been used in the existing literature, which can be described by the categories time, labor, performance, club characteristics, player characteristics, and popularity (Franceschi et al., 2023). Figure 2.5 and Figure 2.6 show the significance levels and the sign of the coefficients of the 10 most used variables.

The most frequently used variable in linear models is the use of age-related variables. A young player is generally worth more than an older player with the same

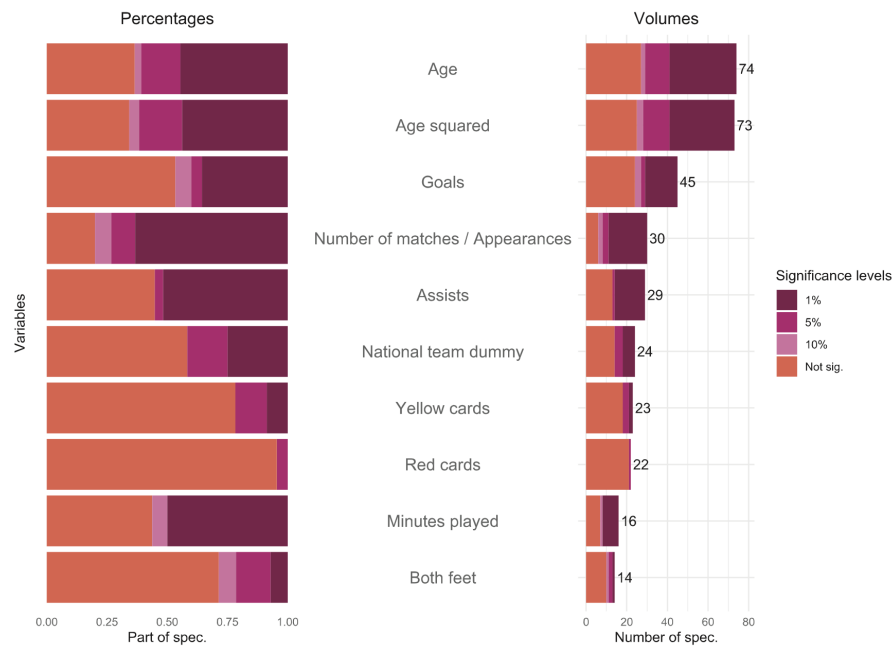


Figure 2.5: Distribution of the significance levels for the 10 most tested independent variables. (Adapted from Franceschi et al. (2023))

quality because they have more potential to increase their quality and they can play more seasons before retiring. As the quality of a football player is widely believed to be an approximately concave function of age, a peak is attained during the player's career and the quality of the player is consequently nonlinear. As can be seen in Figure 2.5, most studies that included age also included the squared value of the features. It is also one of the features that is most often found to be significant. Figure 2.6 additionally shows that  $\text{age}^2$  generally has a negative sign, which indeed means that the average value of a player is approximately concave with respect to age and, therefore, attains a maximum. The age of a player seems to be an important variable that influences the transfer fee in a nonlinear way.

Other variables that are often significant are the number of appearances of a player and the minutes played. One can see that minutes played are less often used than the number of appearances. As it is reasonable to assume that the number of appearances and number of minutes played are highly correlated, one of these variables is probably left out to increase the quality of a model. It can be seen however that both variables are often significant and both have a positive relation with the value of a player. Variables containing information about the amount of playing time, therefore, seem to be



Figure 2.6: Distribution of the sign of the coefficient for the 10 most tested independent variables when significant at 5% threshold. (Adapted from Franceschi et al. (2023))

important.

As can be seen in Figure 2.5, a variable indicating national team experience is often used. Experience at the national level is also always found to increase the player value as can be seen in Figure 2.6. However, this variable is found to be significant less than half of the time. This might be because a dummy variable does not contain much information. It might be interesting to include a feature that describes the international experience in more detail.

Figure 2.5 shows that the number of goals and assists are variables that are often studied. This might be due to the case that these KPIs are better available. Nonetheless, these KPIs mainly focus on attacking quality as discussed in subsubsection 2.1.2.3. Although these variables provide extra information about the quality of players, it might be better to exclude them to reduce the bias of the models towards the attackers.

The number of yellow and red cards are two other variables that are often included. Nevertheless, these variables are often found to be of insignificant influence on the market value. The inconclusive results about these variables can be due to differences in playing styles and interactions. On the one hand, a defender could receive many cards, because he enters many duels which could have a positive influence. On the other hand, receiving a card is a disadvantage in a game, which could lead to a negative

influence. In general, it can be concluded that the number of yellow cards and red cards is a variable that adds minimal information to a transfer value model.

As can be seen in Figure 2.5 and Figure 2.6, time-dependent variables are not often used. (Poli et al., 2021) is an example of a study that did include a variable modeling the inflation by averaging the value of the 100 largest transfer fees in the last four transfer windows. They found this variable to be significant with a positive slope, which means that higher inflation gives higher transfer fees as would be expected. However, only information is known on the historical inflation and the prediction of future inflation numbers is a challenge. Such macro-economic developments are outside the scope of this thesis.

The OLS models were generally constructed as descriptive models. The research performed by Poli et al. (2021) and Al-Asadi and Tasdemir (2022) were exceptions on this. In the descriptive studies, the explainability of OLS models in combination with the significance testing makes it possible to obtain insight into what factors influence the transfer fees. As the current thesis aims at constructing predictive models, predictive quality will be assessed via out-of-sample losses instead of in-sample losses.

#### 2.2.2.2 | Estimated Transfer Value

In contrast to these descriptive studies that used OLS models, there also exist predictive studies using machine learning models. These methods are often able to find more complex patterns in data by finding nonlinear relations and possibly interactions. Additionally, these models are more similar to the models considered in this thesis.

#### 2.2.2.3 | Estimated Transfer Value

First, the Estimated Transfer Value is a model created by SciSports (2024). They trained the model on historical transfers to estimate the transfer value of a player using features such as the league strength, age, international experience of a player, and the contract situation. The values of this model were used in the research performed in the current thesis. The literature on this type of models in the literature will be discussed now.

#### 2.2.2.4 | Added value of complexity

Al-Asadi and Tasdemir (2022) studied the predictive performance of several machine learning models on the market value using data from the video game FIFA 20. They used the market values as an independent variable with the quality indicators of FIFA

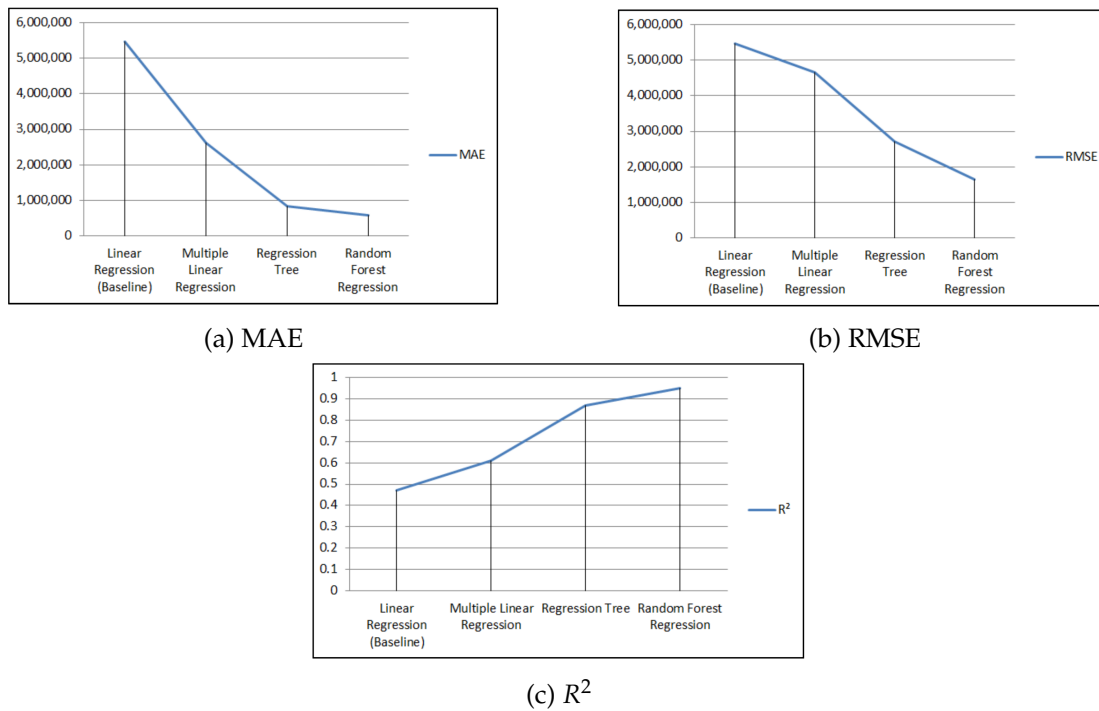


Figure 2.7: The performance of the models in Al-Asadi and Tasdemir (2022) for different metrics with the models sorted on complexity. (Adapted from Al-Asadi and Tasdemir (2022))

as predictors. These features include an overall rating and a potential rating which predicts the quality of a player in the future, which was obtained using expert judgment. The considered models were linear regression, regression trees, and random forests and the results are shown in Figure 2.7.

A regression tree is a relatively simple model, but the estimated loss values indicate that it outperforms the linear regression models. As a regression tree is better able to catch possible nonlinearities and interactions, this implies that the prediction of transfer values seems to be a problem with nonlinearities or interactions. Methods that can incorporate nonlinearities and interactions are, therefore, more likely to succeed in this problem.

The models in Figure 2.7 are shown with increasing complexity of the models. It can be seen that the random forest outperforms the other methods. As it is a more complex and flexible method than linear regression and regression trees, it is better able to capture certain patterns in the data. This means that more complex models seem to perform better on this problem. As the random forest is the most complex method

used in the study by Al-Asadi and Tasdemir (2022), it would also be interesting to see how even more complex methods such as XGBoost and neural network-based models would perform. These could potentially provide an improvement in accuracy.

#### 2.2.2.5 | Study using artificial neural networks

A study with artificial neural networks was performed by Steve Arrul et al. (2022), who studied the predictive performance of artificial neural networks (ANNs) for the prediction of market values. They obtained the market values and player performance KPIs from the video game FIFA 19, similar to Al-Asadi and Tasdemir (2022). One of the baseline models they implemented is an ANN with a linear activation function, which is a linear regression model trained with back-propagation. They found an  $R^2$  value of 0.477, which is lower than the value of 0.61 found in Al-Asadi and Tasdemir (2022), which might be caused by back-propagation not being the right optimization method for the fitting of a linear model.

During the training of their first model with a Relu activation function, the initial model was found to be overfitting. To solve this problem, they applied  $L^1$ -regularization which can prevent overfitting, and found this significantly increased the predictive performance. This indicates the possible necessity of including regularization in order to prevent overfitting.

Their final model managed to have an  $R^2$  of 0.95, which is a significant improvement on the baseline method. The  $R^2$  value is similar as shown in Figure 2.7 for Random Forest Regression. Due to the different data sets and the different performances of the baseline methods it is not possible to compare the quality of the random forest regression of Al-Asadi and Tasdemir (2022) and this ANN. Nonetheless, the result of this research showed that ANNs might be a method suited for this type of prediction problem.

#### 2.2.2.6 | Interpretable modeling

Similarly, Yang et al. (2022) also applied random forest regressors to predict the transfer fees that clubs pay for footballers. Next to a random forest, they also trained generalized additive models (GAMs) and quantile additive models (QAMs). QAMs were applied as these take the quantiles of the data into account which gives better results for a skewed dataset. They used the logarithm of the transfer fee as the dependent variable to make the distribution of the dataset less skewed. The authors included numer-



ous variables for their models including player characteristics, performance indicators, club information, and time effects.

The GAMs and QAMs are methods that provide the possibility for the visualization of relations between the dependent and independent variables as visualized in Figure 2.8. For instance, the results show that the number of appearances shows nearly linear relations with the market value, but that the slope is dependent on the quantile of the market value. This indicates that different patterns occur for different groups of players, which makes it hard to model. If more simple models such as OLS are applied, this could be dealt with by fitting multiple models on disjoint subsets of the data.

On the other hand, several features have nonlinear relationships with the dependent variable. Yang et al. (2022) showed that GAMs and QAMs can be used to do significance testing on nonlinear relations. As an example, the remaining contract and the age can be seen to have a nonlinear relation with the estimated transfer fee. In this way, the use of QAMs and GAMs can provide insights such as the presence of nonlinearities and differences in slopes dependent on the size of the transfer fee.

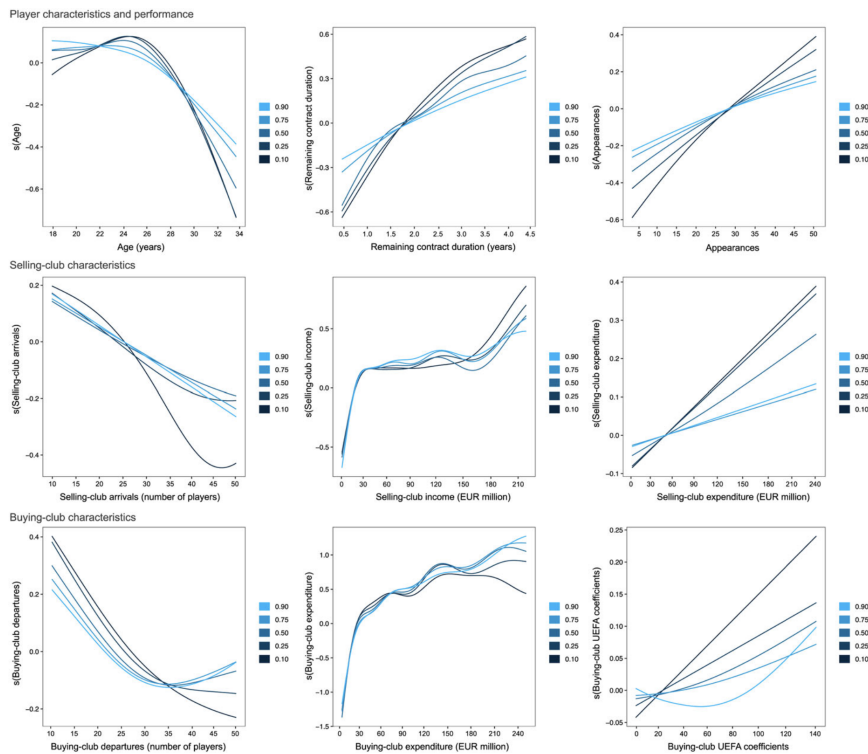


Figure 2.8: The selected quantile additive smooth effects of non-linear predictors of transfer fees by Yang et al. (2022). (Adapted from Yang et al. (2022))

Next to that, their random forest model also provided insights into the feature importance as shown in Figure 2.9. It can be seen that the expenditure of the buying club and the income of the selling club are the most influential features. This is something that our model for future transfer values cannot take into account as it is not known what teams would be interested in a player in the future. However, the method they use gives insight into how feature importance can be investigated for tree-based methods.

They found that the random forest estimator performed best with the  $R^2 = 0.67$ . With this final model, they studied the performance before and during COVID-19 and found that the predictive performance of the models was significantly less for transfers during COVID-19. This implies that transfers during COVID-19 behaved differently.

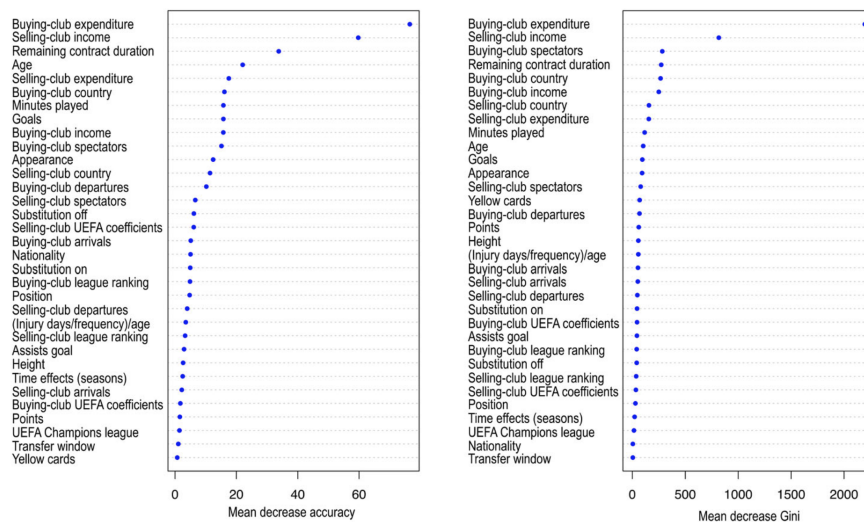


Figure 2.9: The feature importance according to the random forest estimator of Yang et al. (2022). (Adapted from Yang et al. (2022))

Figure 2.9 shows that information about the buying club is the most influential feature in the random forest model. This can be expected as the fee of a transfer is likely to be higher if the selling club knows that the buying club has a large budget. This is in line with the findings of Depken and Globan (2020) who found that clubs in better competitions, that have more money, tend to pay more for their players. This could imply that it is important to involve information about the buying club in a model for transfer fee estimation.

However, including this information does not follow the definition of either a transfer value or a market value. More importantly, the addition of these features requires

the user of the model to specify a buying club, which could reduce the use case of a model for transfer fee estimation. Next to that, it should be noted that Yang et al. (2022) only used basic player performance KPIs like appearances and goals of a player. It could well be possible that the buying-club expenditure is a feature that is highly correlated to player performance as a well-performing player is more likely to make a transfer to a high-level club, which generally has a higher expenditure. With the inclusion of a player performance metric in a player valuation model, the need for information on a buying club is less. Because of these limitations, the information about a buying club was not included in the research of this thesis.

### 2.2.2.7 | Prediction with advanced player performance metrics

McHale and Holmes (2023) also carried out a predictive study on the estimation of transfer fees. They used overall ratings and potential ratings from the FIFA video game combined with several other predictors and KPIs such as playing minutes, height, position, PM ratings based on xG, and VAEP ratings. On this data, they applied an OLS model, a mixed effects linear model, elastic-net regression, xgbDART, and xgbTree (default XGBoost) models.

The results of McHale and Holmes (2023) are depicted in Table 2.1 and show that for all summary statistics, the xgbTree model performed best. The xgbDart model perform slightly less well. The fact that these models perform best could be an indicator that the problem has nonlinear relationships that are not considered by the other models that only consider linear relations.

Model	MAE	MAPE	$R^2$
xgbTree	3.60	67.47	0.77
xgbDART	3.64	68.90	0.76
mixed linear effects	4.11	69.05	0.74
elastic-net	9.93	90.57	0.50
OLS	10.34	91.81	0.50

Table 2.1: The accuracy results in the study performed by McHale and Holmes (2023) with the loss functions mean absolute error (MAE), mean absolute percentage error (MAPE), and  $R^2$ . (Adapted from McHale and Holmes (2023))

The mixed linear effects model surprisingly outperforms the elastic-net and OLS estimates although they are all linear models. The performance of the mixed linear effects model even comes near the performance of the gradient boosting algorithms. This might be because the mixed linear effect model compensates for the random effects

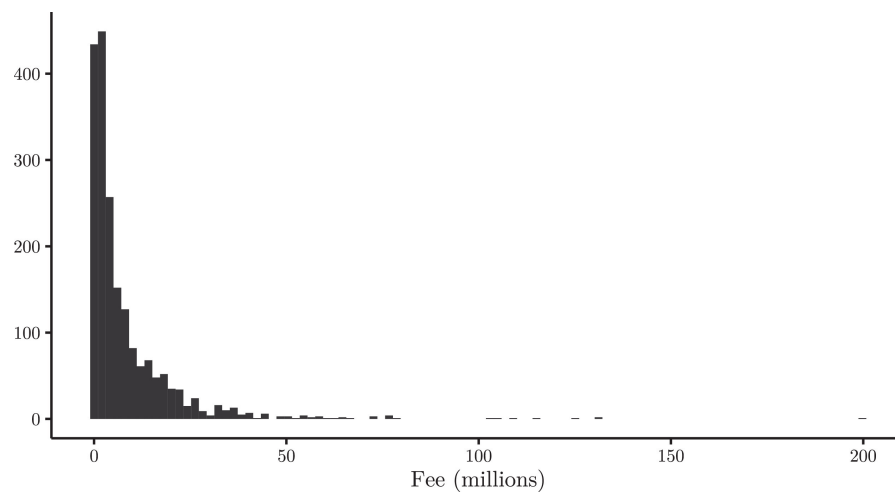


Figure 2.10: A histogram with the distribution of the transfer fee values in the data set of McHale and Holmes (2023). (Adapted from McHale and Holmes (2023))

of buying and selling clubs, which are important as shown in the research of Yang et al. (2022). This shows the potential of mixed linear effects models combined with the importance of club information in transfer fee estimation.

Next to that, the authors also compare the results of their xgbTree model with the Transfermarkt's market values. They find that their model outperforms these market values in estimating the actual transfer fees on average. They also studied the performance of the model for transfer fees with different fee values. In contrast to the average, the Transfermarkt values outperform their model for transfers higher than 20 million pounds. This could be caused by skewed data as their data set is highly skewed as shown in Figure 2.10. It might be beneficial to implement a second model on data containing the upper segment of the transfer fees or to create a hybrid model that gives the final prediction based on the predictions of other models.

#### 2.2.2.8 | Modified models

A combined model was obtained by Yigit et al. (2020), who tried to predict the market values from Transfermarkt. They used a regularized linear model, a decision tree, a random forest, and XGBoost model to predict the market values using performance indicators of the Football Manager 2018 video game. Although they do not give extensive consideration to the estimated losses of the models, they found that regularized linear model outperformed the random forest model. They finally considered the XGBoost and regularized linear models as the best methods and concluded that a weighted av-

erage should be used where the results of XGBoost provide 70% of the outcome and the regularized linear model 30%. This study presents an example of an approach where multiple methods are combined to obtain one joint model.

Behravan and Razavi (2020) also introduced a method that contains a nontraditional machine learning approach to predict market values based on FIFA 20 performance indicators. They extensively used particle swarm optimization (PSO) as the first to use a PSO-based clustering method to cluster players. On these clusters, they applied support vector regression (SVR) and used PSO to optimize the prediction accuracy. In the optimization of SVR, the PSO algorithm also applied feature selection. The paper lacks a baseline method to compare the quality of their model with other models, so no conclusions can be drawn about the quality of the used methods. Nonetheless, their research shows that evolutionary algorithms can be used for the optimization of algorithms.

#### 2.2.2.9 | Summary

To summarize, different types of models were applied in the studies discussed. The studies in general showed that machine learning models can outperform linear methods in predicting the transfer values. For all studies, the performance improved for more complex models, which indicates that the prediction task seems to be of a complex nature. As the prediction for future values of transfer fees involves more uncertainty, the prediction of transfer fees might, therefore, prove to be a difficult challenge for which the type of model should be able to capture complex patterns present in the data.

### 2.2.3 | Discussion of the studies

As the existing studies for transfer fee estimations have been discussed, overarching topics can be discussed to obtain insight into the current state-of-the-art available in academic literature. These insights provide supplementary understanding and context to the project.

#### 2.2.3.1 | Lack of studies on future value estimation

As discussed, there exist numerous studies on the transfer fees of players and influential factors for this. This could give rise to the expectation that there are studies on future values of expected transfer fees. However, to the best of our knowledge, Payyap-

palli and Zhuang (2019) and Baouan et al. (2022) performed the only studies considering future values of player valuations at the time of this literature review. Payyappalli and Zhuang (2019) used a moving average window to obtain predictions for future values, but they did not cover this in detail.

Baouan et al. (2022) studied the important indicators for future market values. They trained a lasso regression and random forest model to predict Transfermarkt market values based on player performance statistics from Wyscout. They trained a different model for each player position and using the feature importances of these models, they studied what features were influential for the development of the market value. They, for instance, found that the inclusion of the average market value of a league was an important indicator. A similar method could be applied in the research of this thesis to infer the important variables.

In their study, they performed 5-fold cross-validation to tune the hyperparameters, and their results for the lasso model and random forest models showed similar performance. It should be noted that they included several higher-order polynomials and success-ratio-based statistics in their feature set. In this way, they accounted for nonlinearities by performing specific feature engineering.

They found that the cross-validation estimates for the  $R^2$  after hyperparameter tuning were between 0.55 and 0.60, which is relatively high. However, it should be noted that for this estimate, the cross-validation estimate was applied. This means that data leakage probably has occurred and, additionally, the hyperparameter tuning was also applied using this same cross-validation method. This led to an overly positive estimate of their test losses. Because the authors did not include any estimations of the losses on a separate test set, it cannot be said what the actual predictive quality of their models on possible unseen future values is.

Baouan et al. (2022) and Payyappalli and Zhuang (2019) conducted the only research on future values of transfer or market values of football players. Payyappalli and Zhuang (2019) applied a moving average to forecast the market value, salary, overall rating, and potential to use it in an optimization problem. They stated that the forecasting of these attributes could be a topic of research itself. Baouan et al. (2022) carried out a descriptive study to identify important player characteristics to forecast future market values. The results indicated that the age and the minutes on the field are important variables in the development of a football player. This means that no study has yet analyzed the predictive accuracy of forecasting future player transfers. To conclude, there is a lack of predictive studies on the forecasting of future player transfer values.

A possible reason for this is the complexity of the estimation of expected transfer fees itself. He et al. (2015), for instance, found that their model was less accurate for the best footballers. This was called the superstar phenomenon by Herm et al. (2014), and is caused by the fact that the best football players are outliers of the general population. As these football players draw the most attention, the accuracy on these players is important for the model to be accepted by practitioners. This creates extra complexity for the modeling task.

Next to that, Herm et al. (2014) showed that public attention and performance are influential factors on the value of a player. As both of these factors can have fluctuations, the resulting transfer fees of a model can be prone to have relatively large fluctuations as well. The fact that the current values can be very volatile makes it a complex task to predict future values as there is already much uncertainty involved.

In short, important factors in the development of future market values have been studied. Still, research on predictive models to forecast the transfer values of football players has not yet been carried out. An important factor in this is the fact that this is a complex modeling problem due to, for instance, the superstar phenomenon and temporal fluctuations in the estimates of a player's value. This shows that the task at hand might be a complex one and stresses that it is important to study the uncertainty of the model.

### 2.2.3.2 | Bias in current studies

As discussed in subsection 2.1.2.3, there exists a bias in the current studies on player performance because there is more knowledge on quantifying the quality of offensive actions than on defensive actions. In contrast with player performance where models were often only constructed for offensive performance, studies on the estimated value of a football player are generally applied to all types of players. This bias in the existing models for player valuation is, therefore, less apparent.

Nonetheless, player valuation methods are often trained using player performance metrics. For instance, Yang et al. (2022) mainly take attacking KPIs into account which creates an indirect bias in the model. This bias is caused by the fact that the features in the model provide more knowledge about the attacking quality of a player which leads to a better estimation of the value for offensively oriented players. This means that the bias in the knowledge about player performance towards offensive players causes a higher uncertainty for the player value estimation of defensive players.

Additionally, models estimating the transfer fee are trained on football players who

make a transfer. Thus, no data is known about the players not making a transfer, which creates a selection bias in the data (Franceschi et al., 2023). Therefore, models that estimate the expected transfer fee should be used precariously when used for player valuation of players that do not make a transfer. Franceschi et al. (2023) state that market values of, for instance, Transfermarkt make it possible to avoid this selection bias.

In short, there exist biases in the current research on monetary football player values, such as a bias towards attackers and a selection bias. These biases make the model less useful for practitioners. The research in this thesis is performed on the data of the Estimated Transfer Value, which is a model with a reduced bias toward attackers due to the underlying SciSkill model. Consequently, there might be some bias present in this research, although it is likely to be limited.

### 2.2.3.3 | Inconsistent definitions of market value

Additionally, it was discussed at the beginning of this section that the models for player value estimation generally estimate the expected transfer fee of a player or the market value of a player. However, according to the recent study of Franceschi et al., there did not exist one clear definition of market value.

Nonetheless, many different studies discussed in this section have different sources for their transfer market values. For instance, market values estimated by FIFA were used by Al-Asadi and Tasdemir (2022), Steve Arrul et al. (2022), and Behravan and Razavi (2020), whereas Yigit et al. (2020) used market values from Transfermarkt. As the different sources might have different definitions, these studies might examine different quantities because of the different underlying definitions.

The main goal of the current thesis is the estimation of future transfer values. As discussed, many studies consider the market value of a player as a proxy variable. This, combined with the different definitions of market values, creates a field of research that uses several different variables to study the same concept. This gives rise to the need in the current thesis to define the concepts involved in this study explicitly.

### 2.2.3.4 | COVID-19

Most of the models studied were trained on data containing seasons that were subject to COVID-19 regulations. In these seasons, factors such as home advantage and quarantine regulations influenced the football transfers. Luo (2023) described the influence of these factors and predicted that the consequences of the COVID-19 pandemic will also be present in the years after the pandemic.



Yang et al. (2022) confirmed this by showing that their models behaved differently on data from the seasons influenced by the COVID-19 pandemic. Their research shows that the fees for high- and medium-priced players were underestimated by their model despite the fact that they included temporal variables that accounted for inflation. This implies that the transfer fees for these transfers have increased. The COVID-19 pandemic, therefore, makes the modeling more difficult, which means that it might be interesting to consider COVID-19-related features for the models in this research. However, the model used for the research in this thesis is time-homogeneous, and temporal features are, therefore, not included.

## 2.2.4 | Economical context

As previously discussed, the early studies on influential factors for transfer fees were based on economic approaches and principles. Franceschi et al. (2023) stated that the focus of the field has shifted towards predictive modeling, even though the economic approaches give important insights into the problem. Therefore, some economic principles will now be discussed to be able to take this knowledge into account for this research.

### 2.2.4.1 | Football transfer market as a labor market

The current football transfer market has been shaped by the Bosman ruling in 1995 (Frick, 2007). Before the Bosman ruling, football clubs could demand a transfer fee for players whose contracts expired, but the Bosman ruling meant that a player could leave transfer-free when their contract had expired. This change in rules caused an increase in average player salary and contract length (Frick, 2007). In the new situation, the leverage of football clubs over players was reduced. This change in regulations created a transfer market that became similar to the general labor market.

There is much more data available on football player performance and player values compared to a normal labor market according to McHale and Holmes (2023). This in combination with the transfer market being similar to the general labor market makes the transfer market an opportunity to study the valuation of human capital and labor markets (Leifheit and Follert, 2021). This means that analyses of the transfer market are interesting for labor market economists as some principles might be generalizable to other labor markets.

### 2.2.4.2 | Challenges of the football transfer market

However, compared to a normal market, the player market is continually changing which makes it difficult to adequately model it (Yigit et al., 2020). In addition to this, an increasing amount of money is involved in transfers, and high transfer fees with a relatively high risk are paid on a small group of investments (McHale and Holmes, 2023). Moreover, some character traits are considered less important in the football industry compared to other labor markets with, for instance, cognitive functions and motivation being considered less important for footballers (Wakelam et al., 2022). Thus, the transfer market differs from a general labor market, which has to be accounted for when trying to generalize conclusions from analyses of the transfer market.

The fact that football clubs are willing to invest large sums in football players, which generally involves large investment risks indicates the determination of football clubs to perform well. Several researchers like (Sloane, 1971) and Dobson and Goddard (2001) argued that football clubs should not be considered to be profit maximizers, but utility maximizers. This means that athletic results generally have priority over financial results. Leifheit and Follert (2021) tried to obtain a more general concept of player value by taking kit sales, influences on other players' development, and financial aspects concerning athletic success into account for a player valuation. In this way, they tried to obtain a more economic approach to the utility maximization of football clubs. In general, it can be concluded that football clubs try to optimize the athletic performance of the team subject to financial constraints. It is, therefore, important that the models about player values are accurate and that the uncertainty is quantified such that this can be taken into account by football clubs when assessing whether their transfer plans satisfy their financial constraints.

To summarize, the football transfer market can be considered as a labor market with a wide availability of data. Even though there exist some differences with the general labor market, studies on the transfer market might obtain insight into the general labor market. A main difference with the general labor market is the fact that football clubs are utility maximizers instead of profit maximizers, which causes the focus of a football club to be different than that of a normal company. Still, analyses of the football transfer market might provide interesting insights into labor markets.

## 2.3 | Predictive models

In this section, possible models from statistical learning for the construction of a model for the prediction problems in the current thesis are discussed. With the knowledge about the existing studies on player performance KPIs and player valuation, these models can be placed into context. For the different types of models, advantages and disadvantages will be discussed with a focus on explainability and uncertainty quantification. Explainability is considered in the current thesis to be the extent to which it can be explained why the model gives certain values. Uncertainty quantification is the extent to which the uncertainty of predictions from a specific type of model can be quantified. In this section, linear, tree-based, neural network-based, and time series-oriented methods will now subsequently be discussed.

### 2.3.1 | Linear methods

The first type of method is the linear method. The linear workings of these models make it possible to interpret the results of the model and explain the values of predictions. On the contrary, linear models cannot find complex patterns due to their simple nature. Because of this, linear models are often used as baseline methods to compare the improvement of the performance for more complex methods.

#### 2.3.1.1 | Ordinary least squares

Ordinary least squares (OLS), sometimes called multiple linear regression (MLR), assumes a linear relation between the dependent and independent variables as described by Hastie et al. (2009) in Section 3.2. This results in the formulation  $y = X\beta + \varepsilon$  where  $\varepsilon \sim N(0, \sigma^2 I)$ . With this formulation, OLS minimizes the residual sum of squares defined as

$$\text{minimize}_{\beta} \quad RSS(\beta) := (y - X\beta)^T (y - X\beta),$$

which has an unique global minimum at

$$\hat{\beta} := (X^T X)^{-1} X^T y$$

if  $X$  is of full rank.

As the predictions are obtained linearly, the explanation of the results of the model is straightforward. Next to that, the linear model makes it possible to interpret the meaning of parameters in the model. Consider the case where  $Y = \sum_i \beta_i X_i$ . If a  $\beta_i$

corresponding to  $X_i$  is positive, it means that a higher value of  $X_i$  is associated with a higher value of  $Y$ . In addition to this, the relatively simple structure of OLS allows for the construction of statistics for testing significance tests with additional assumptions such as the normality of the features and labels. Although these assumptions do not generally hold, they can be used as a heuristic to select features. In this way, OLS can be used for feature selection based on statistical tests with a theoretical foundation.

The Gauss-Markov theorem also provides a theoretical analysis of the OLS method as it proves that OLS has the smallest mean square error of all unbiased linear models (Hastie et al., 2009, p. 51). Nonetheless, the underlying principle of the bias-variance trade-off in statistical learning implies that introducing a bias in the model may outperform the unbiased case. A high number of variables leads to a high variance and causes overfitting, and these problems can be addressed by regularization, which introduces bias into the model. Moreover, the linearity assumption makes it harder to accurately construct a model as nonlinearity and feature interactions can only be introduced in feature construction. It is infeasible to include all possible interaction terms of higher order or nonlinear transformations in a scenario with a large number of features as the number of interaction terms blows up in this case. For the above reasons, the OLS often has a low accuracy on prediction problems.

A benefit of the OLS model is the fact that it has natural uncertainty quantification methods. As shown by Neter et al. (2004) in Chapter 2, the true coefficients of OLS follow a t-distribution after ‘studentization’, which is subtraction by the estimate and division by the estimated standard deviation. Similarly, they describe how to obtain a prediction interval for unseen observations. It should be noted, however, that these proofs have assumptions that are not met in the case studies of this research, like normality in the response variable. Although these intervals cannot be used as true prediction intervals for the potential application in this use case, they can be interpreted as some measure of the uncertainty of the prediction. This means that the OLS model has a naturally arising way of quantifying the uncertainty of the model, both in the coefficients and in the predictions.

### 2.3.1.2 | Regularized least squares

To reduce the variance, a regularization term can be added to the OLS model which introduces bias in the model. This is done by reformulating the optimization task by adding a penalty term for the size or number of estimated coefficients or the number of

nonzero coefficients. The general problem that is optimized can be described as

$$\text{minimize}_{\beta} \quad RSS(\beta) + P(\beta) = (y - X\beta)^T(y - X\beta) + P(\beta).$$

Because models with large or many coefficients are penalized, smaller values for the coefficients are selected or fewer coefficients are selected. This means that the addition of the penalty term shrinks the coefficients of the optimal  $\beta$  towards zero which decreases the variance of the model at the cost of some bias.

The penalty term is often chosen as  $P(\beta) = \lambda \|\beta\|_q^q$  or the sum of multiple  $\ell_q$  penalties. The two well-known examples are lasso regression ( $\ell_1$  penalty) and ridge regression ( $\ell_2$  penalty) as described by (Hastie et al., 2009, pp. 61-73). The ridge regression was popularized by Hoerl and Kennard (1970) and has a closed-form solution to the optimization problem in the form of

$$\hat{\beta}^{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T y$$

similarly to OLS.

Lasso regression was described by Tibshirani (1996) and is obtained by introducing the  $\ell_1$  penalty resulting in the objective function in the form of  $RSS(\beta) + \lambda \sum_{i=1}^p |\beta_i|$ . The estimated coefficients can be described using the coefficients of OLS using

$$\hat{\beta}_j^{\text{lasso}} = \hat{\beta}_j^{\text{OLS}} \max \left( 0, 1 - \frac{\lambda N}{|\hat{\beta}_j^{\text{OLS}}|} \right).$$

This means that the coefficients could be calculated via the OLS coefficients, but this would give computational problems due to the near-singularity of matrices in the case in which noise features are present. To solve this, it is generally optimized using a gradient descent algorithm.

Lasso and ridge penalties shrink the coefficients towards zero. However, the  $\ell_1$  penalty of lasso regression actually forces the coefficients to be equal to zero, whereas the ridge penalty forces coefficients to be close to zero. In these ways, regularization can be used to introduce bias and improve the predictive accuracy of linear models. Still, the lasso model also applies feature selection by itself as it forces values to be zero.

Due to the introduction of regularization, regularised least squares loses the possibility to construct significance tests for coefficients and prediction intervals. These are obvious drawback of the regularized linear models.

### 2.3.1.3 | Mixed effect linear model

A linear mixed effects model is a linear model that can be used to predict transfer fees as shown by McHale and Holmes (2023). In contrast to ordinary least squares and regularized regression, a linear mixed effects model distinguishes between fixed and random effects. As described by Lindstrom and Bates (1988), this is done by assuming that

$$y_i|b_i \sim N(X_i\beta + Z_ib_i, \sigma^2\Lambda_i),$$

where  $X_i$  is the fixed effect of a group  $i$  and  $Z_i$  is the random effect of the group  $i$ . The distribution of the random effects is assumed to be  $N(0, \sigma^2 D)$ , which gives that  $y_i$  are marginally independent with mean  $X_i\beta$  and covariance matrix  $\Sigma_i = \sigma^2(\Lambda_i + Z_i D Z_i^T)$ . The values for  $\Lambda_i$  and  $D$  are then estimated by estimating the maximum likelihood estimator using a Newton-Raphson method. With these estimated matrices, the estimated parameters can be calculated by the formula

$$\hat{\beta}(\theta) = (X^T V^{-1} X)^{-1} X^T V^{-1} y,$$

where  $V = \Lambda + Z D Z^T$ .

In a linear fixed effects model, the fixed effects correspond to the features that are studied for their systematic influence on the prediction, while random effects correspond to variables that might influence the result in random ways like subjects in a study where different measurements are performed on one subject. In this way, a linear mixed effect model can take randomness in distribution the of the data into account.

Due to its linearity, a linear mixed effect model is still an interpretable model with explainable results. On the other hand, feature selection cannot be applied and there does not exist a regularization hyperparameter to help in the bias-variance trade-off. Still, by distinguishing between fixed and random effects, more factors can be taken into account by the linear model. This means that the linear mixed effects model might perform better or worse depending on the context. As it was applied by McHale and Holmes (2023) with promising results, this is an interesting model.

### 2.3.2 | Tree-based methods

More flexible prediction models can be found in the form of tree-based methods. These are often based on fitting one or multiple decision trees to the data. As they partition the feature space into a set of rectangles, decision trees can take nonlinearities and interactions into account which makes them more suitable for more complex prediction problems compared to linear models.

### 2.3.2.1 | CART Decision Trees

The most simple tree-based method is a single decision tree, generally fitted using the CART-method (Hastie et al., 2009, pp. 305-308). A CART decision tree divides the feature space into rectangles and then takes the average value on each rectangle as the predicted value. This is visualized in Figure 2.12. The CART method hierarchically constructs the splits by empirically finding the split that would minimize the sum of squares for each variable. The variable with the best split is then chosen. This process can be repeated multiple times to produce splits. Each split can be represented as a choice in a decision tree as visualized in Figure 2.11. This creates a model for which it is easily explained how it works. Although the model is not linear, the splits give a clear interpretation. Figure 2.12, for instance, shows that high values for  $X_1$  are generally associated with a higher value of the dependent variable.

The CART method itself works by testing all possible splitting points  $s$  for variables  $j$  in the features as described by Hastie et al. (2009) in Section 9.2. Define

$$R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\}.$$

The CART algorithm then defines the split that minimizes the optimization problem as

$$\underset{j, s}{\text{minimize}} \left[ \min_{c_1} \sum_{x_i \in R_1(j, s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j, s)} (y_i - c_2)^2 \right]. \quad (2.4)$$

The inner optimization is solved by taking the averages of the points in  $R_1(j, s)$  and  $R_2(j, s)$ . The CART algorithm uses this by scanning through all input points to check all possible splitting points  $s$  and variables  $j$ . In this way, the CART algorithm optimizes the problem in Equation 2.4.

The hyperparameters of CART trees are the maximum depth and minimal leaf size. A deeper decision tree creates a more flexible model decreasing the bias of the model. On the other hand, it can be chosen to require more points per leaf, which decreases the variance of the model but increases the bias. These hyperparameters can be tuned to deal with the bias-variance trade-off. However, the decision trees generally suffer from a high variance, fail to capture smooth surfaces, and have difficulties in capturing additive structures (Hastie et al., 2009, pp. 310-313). A decision tree, therefore, generally has a bad predictive performance.

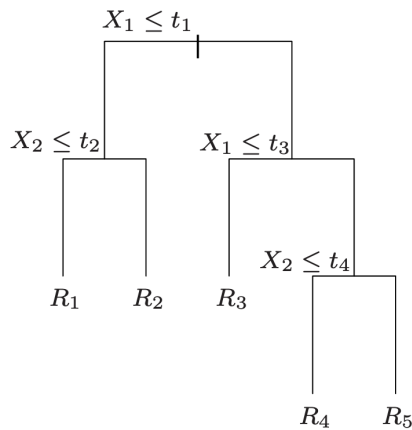


Figure 2.11: A visualisation of a decision tree. (Adapted from Hastie et al. (2009))

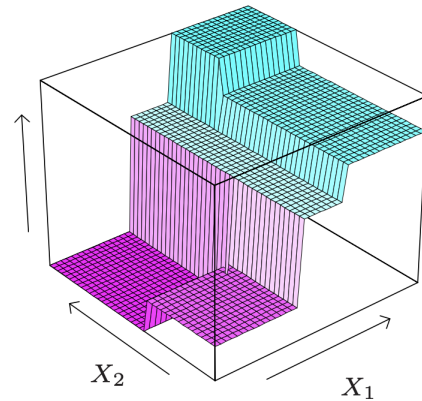


Figure 2.12: A perspective plot of the prediction surface of a decision tree. (Adapted from Hastie et al. (2009))

### 2.3.2.2 | Random Forest

To solve this, bagging or bootstrap aggregation can be applied to trees. With bagging, a model is repeatedly trained on resampled data obtained via the bootstrap method. A new, bagged model is then obtained by taking the average resulting values of the individual models, which reduces the variance of the model. Bagging especially works well for methods with a high variance and low bias such as decision trees (Hastie et al., 2009, Ch. 15). A random forest is a model obtained by applying bagging to decision trees as this reduces the variance of a decision tree.

The decision trees in a random forest are fitted on the same data up to differences due to the bootstrap, which creates strong correlations between the decision trees. Random forest deals with this problem by taking a random subset of the features for the construction of each decision tree, which improves the quality of the predictions.

The bagging method introduces a new type of hyperparameter to the tree-based learner which is the number of trees fitted. With a high number of trees, the bias of the method reduces, with a minimal increase in the variance and a large increase in computational time. This hyperparameter can be taken into account when tuning the models.

Because a random forest uses numerous different decision trees, it is no longer straight-forward to explain how a model came to certain values as with decision trees and the interpretation of the model itself is less straightforward. To obtain insights



into the explainability of results, an implementation of Shapley values can be applied to show how certain values differ from the mean and what the influence is of certain variables (Lundberg and Lee, 2017). Next to this, it is possible to measure the feature importance as shown in Figure 2.9. In these ways, random forests can still be an explainable method.

The random forest model consists of many decision trees that are obtained by the bagging procedure. These individual decision trees all make their individual predictions, which means that they provide a set of estimations. If the decision trees have similar predictions, the model can be said to be certain of the final prediction and the other way around. The method described by Wager et al. (2013) makes use of this bagging procedure by applying an infinitesimal jackknife which approximates the effect of small perturbations to the data to estimate the variance of the predictions. In this way, it is possible to leverage the bagging procedure to obtain a model-specific uncertainty qualification method for the random forest model.

### 2.3.2.3 | XGBoost

Another way of improving the predictive quality of tree-based learning is by applying a boosting algorithm. Similarly to a random forest, numerous different decision trees are fitted on the data to obtain a new model. In contrast with random forests, a boosting algorithm fits these decision trees sequentially and reweighs the samples with the worst predictive performance to improve the predictive quality on these data points. This provides another way of making sure the predictions of the individual decision trees are not highly correlated.

A commonly used boosting method is the XGBoost model introduced by Chen and Guestrin (2016). They defined the objective function of the  $t$ th tree for the  $i$ th iteration as

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t) = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \gamma T + \frac{1}{2} \lambda ||w||^2,$$

where  $l$  is a differentiable convex loss function. The algorithm then uses a second-order approximation to optimize this expression where  $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$  is the first order gradient statistic and  $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$  is the second order gradient statistic on the loss function. Assume that a decision tree is fitted to the data called  $q$ . If  $I_j$  is then the instance set of leaf  $j$ , the optimal weight of leaf  $j$  can be derived and equals

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}.$$

This results in the optimal value with respect to the leaf weights that is

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T.$$

This value can be viewed as a measure of the quality of the tree structure  $q$ .

The CART algorithm used in decision trees and random forests would now check the value at each splitting point. The XGBoost algorithm reformulates the value as

$$\mathcal{L}_{\text{split}} = \frac{1}{2} \left[ \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma,$$

where  $I_L$  and  $I_R$  are the instance sets of the left and right nodes after a split and  $I = I_L \cup I_R$ . The XGBoost algorithm then estimates feature quantiles and uses this to find suitable split candidates to check. This algorithm approximates the best value and also takes into account possible sparse values of features.

Whereas random forests can fit decision trees in parallel, a gradient boosting algorithm cannot due to the sequential algorithm which significantly increases the computation time. However, the approximate algorithm of the XGBoost model drastically decreases the computational time as shown in Figure 2.13 and results indicated minimal costs in prediction performance.

To prevent overfitting, the XGBoost algorithm also introduces a regularization penalty on the number of trees and the leaf weights. With several parameters for the decision trees, the weighing function, and the regularization, the XGBoost algorithm has many hyperparameters that need to be tuned. This takes more time, but also makes the model more flexible which makes it possible to fit the model to many different problems. With this, the XGBoost algorithm is a high-performance boosting algorithm with a good computational time (Chen and Guestrin, 2016); (Brownlee, 2018, pp. 16-17).

Similarly to the random forests, XGBoost calculates the predicted values by taking an average of the prediction of multiple trees although the XGBoost uses a weighted average. Because of this, the feature importance and Shapley values can be calculated analogous to the way these are calculated for random forests. Therefore, the explainability of the model is still satisfactory. The Python implementation also provides possibilities for quantile regression, which provides uncertainty quantification. However, this method needs a new model to be trained, which is a downside compared to the uncertainty quantification methods of the OLS and random forest models.

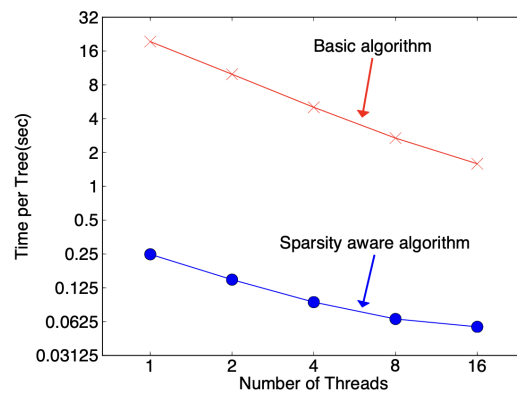


Figure 2.13: The computation time per tree in seconds plotted against the number of threads used for the training time of a single tree in a boosting algorithm for a basic algorithm and a sparsity-aware algorithm used in XGBoost. (Adapted from Chen and Guestrin (2016))

### 2.3.3 | Neural networks

Another type of model is that of an artificial neural network (ANN). Neural networks are based on combining linear combinations of features with nonlinear activation functions, called hidden layers, as shown on the left in Figure 2.14. The universal approximation theorem introduced by Cybenko (1989) proved that with this construction it is possible to asymptotically approximate any arbitrary continuous function on a subset of the Euclidean space  $R^m$ . For this, the width of the ANN should be able to go towards infinity.

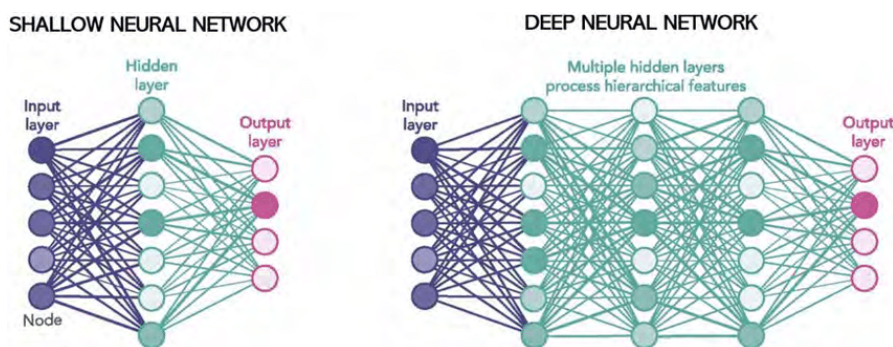


Figure 2.14: Schematic of a shallow neural network (left) and a deep neural network (right). (Adapted from Han et al. (2020))

### 2.3.3.1 | Deep neural networks

It is computationally infeasible to have a wide enough ANN to fully capture most functions but it was found that an ANN can estimate functions more accurately by adding extra hidden layers. A visualization of this is shown on the right in Figure 2.14. Neural networks with multiple hidden layers are called deep neural networks (DNN). The weights in the linear combinations of each layer can be locally optimized by applying methods based on gradient descent. The gradient of each linear combination can be computed using the backpropagation algorithm which makes use of the chain rule for taking derivatives. In this way, a DNN can be obtained that estimates a function.

There exist several methods that use gradient descent to optimize neural networks. Stochastic gradient descent uses a stochastic approach to estimate the gradient by tackling random batches of the samples to apply backpropagation. Several convergence theorems exist for this method providing a theoretical background (Garrigos and Gower, 2023; Zhang et al., 2021). The limited memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) method, on the other hand, is a quasi-Newton method with low iteration costs that can be used for the optimization of DNNs (Liu and Nocedal, 1989). The settings of the optimization methods can be used to prevent overfitting, but also a regularization penalty can be added. With different choices for the optimization methods, the regularization, the depth, and the width of a deep neural network, this method provides a flexible method that can be applied to many problems.

However, DNNs generally need a large data set to perform well (Zhang et al., 2021). Next to that, it is hard to explain how resulting values from a DNN are constructed. It is possible to do this with Shapley values like with the tree-based methods but this is more time-consuming for neural networks. Although a DNN does have methods to quantify uncertainty such as utilizing the architecture or applying a Bayesian approach (He and Jiang, 2023), the explainability of these methods in laymans terms is a problem. To summarize, DNNs provide a method with a good predictive performance on large data sets but with limited possibilities for explaining results and uncertainty prediction of the model. As this type of model was not considered sufficiently explainable, they were not implemented in the research of the thesis.

### 2.3.3.2 | LSTM networks

Up to now, all models had a vector of a fixed size as input to calculate a predicted value. The prediction problem of the current thesis contains data of a fixed length such as player age, height, and minutes played in the last season. In addition to this, it

contains historical data on the KPIs of interest which can be viewed as a time series. By viewing the problem as a classical prediction problem with a fixed length vector, information is lost that might increase predictive accuracy.

Recurrent neural networks (RNNs) are neural networks that can handle time series as input. In contrast with a classical ANN, the RNN contains nodes that have a link with itself as shown in red in Figure 2.15. However, training of RNNs is found to be problematic due to the vanishing gradient problem as described by Chollet (2017). They also write that Long short-term memory (LSTM) networks were introduced in order to solve this problem. With a special construction of the sigmoid and tanh activation functions, both a short-term and long-term memory information stream is obtained by an LSTM network. Because of this, the LSTM networks do not suffer from the vanishing gradient problem according to Chollet (2017). Next to that, it was found that LSTM networks are good at analyzing the global, long-term structure of sequences and that hyperparameter tuning and regularization methods are important for the performance of LSTM networks.

Because of the ability of LSTM networks to take a time series as input, LSTM networks offer a method that potentially outperforms the others. As an LSTM network can have a higher output dimension, the quantiles of the distribution can also be included during training similar to the methods of Fan et al. (2023), which makes it possible to perform uncertainty quantification. Nonetheless, LSTM networks are models that are hard to explain, due to the difficulties of a neural network combined with the complex construction of the LSTM network. Because LSTM networks are not sufficiently explainable, they are not implemented in the research of this thesis.

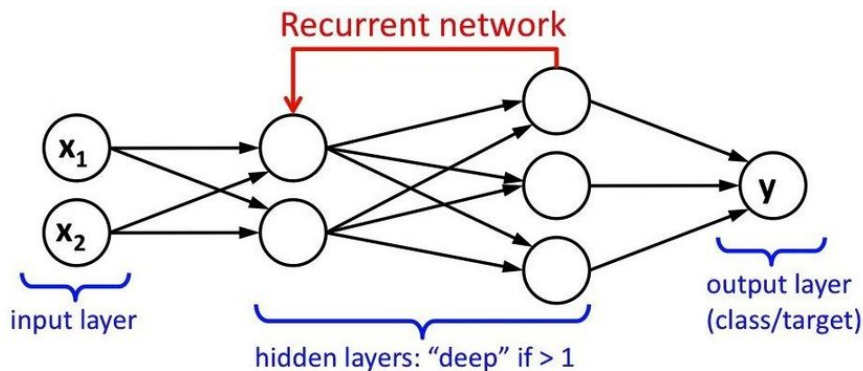


Figure 2.15: Visualisation of a recurrent neural network. (Adapted from Mishra et al. (2018))

### 2.3.4 | $k$ nearest neighbors

The last model considered in this research is the  $k$  nearest neighbor (kNN) model. As described by Hastie et al. (2009) in paragraph 2.3.2, this model calculates the distance of input features  $X \in \mathbb{R}^p$  with the features of the data points in the training set  $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$ . It then takes the  $k$  points with the least distance, called the neighbors, and predicts the value based on the labeled values. For classification, voting is performed and an average is taken for regression. This can be described as  $f(X) = \frac{1}{n} \sum_{i=1}^n Z_i(X) Y_i$  where  $Z(X) = (Z_1(X), \dots, Z_n(X))$  is the solution to the integer linear optimization problem

$$\begin{aligned} & \underset{Z(X) \in \{0,1\}^n}{\text{minimize}} && \sum_{i=1}^n Z_i(X) \|X - X_i\|, \\ & \text{s.t.} && \sum_{i=1}^n Z_i(X) = k. \end{aligned} \tag{2.5}$$

Dudani (1976) introduced an improvement of the kNN model by giving weights to the different neighbors. To each neighbor, they assigned the weights  $w(i) := \frac{d_{\max} - d_i}{d_{\max} - d_{\min}}$ , where  $d_i = \|X - X_i\|$ ,  $d_{\min} := \min\{d_i | i = 1, \dots, n \text{ and } Z_i(X) = 1\}$ , and  $d_{\max} := \max\{d_i | i = 1, \dots, n \text{ and } Z_i(X) = 1\}$ . Equal weights were assigned in the case where all distances  $d_i$  are equal. They also provided the alternative weights  $w_i = \frac{1}{d_i}$  where it is assumed that  $d_i \neq 0$ . The prediction of a neighbor-weighted kNN is obtained by  $f(X) = \frac{1}{n} \sum_{i=1}^n w_i Z_i(X) Y_i$ . Later, Bicego and Loog (2016) introduced a new view of the kNN model by considering it as a combined classifier with each neighbor being an expert prediction. They showed that their method and the method introduced by Dudani (1976) improved the predictive quality in experiments carried out in the research.

Although kNN algorithms do not have feature importances, Kononenko (1994) introduced a feature importance measure based on the distances of a kNN classifier called the ReliefF algorithm. This method was later adapted by Robnik-Šikonja and Kononenko (1997) to be applicable to regression problems. As described by Robnik-Šikonja and Kononenko (2003), this algorithm was called the Regressional ReliefF (RReliefF) method and determines the feature importance  $W[A]$  of feature  $A$  in a kNN model by defining

$$\begin{aligned} W[A] := & P(\text{different value of } A | \text{nearest instances}) \\ & - P(\text{different prediction} | \text{nearest instances}). \end{aligned}$$

These probabilities in this expression were approximated using the pseudo-code in Fig-

ure 2.16. In this pseudo-code, the term *diff* is defined as

$$\text{diff}(A, I_1, I_2) := \frac{|value(A, I_1) - value(A, I_2)|}{\max(A) - \min(A)}$$

where  $value(A, I)$  is the value of feature  $A$  for data point  $I$ . Next to this,  $d(i, j)$  should be a measure that indicates the similarity between data point  $i$  and data point  $j$ . This  $d(i, j)$  can be chosen with similar formulas as the  $d_i$  in the neighbor weighting and should be normalized to sum to 1 for each data point.

*Algorithm RReliefF*  
*Input:* for each training instance a vector of attribute values  $\mathbf{x}$  and predicted value  $\tau(\mathbf{x})$   
*Output:* vector  $W$  of estimations of the qualities of attributes

1. set all  $N_{dC}$ ,  $N_{dA}[A]$ ,  $N_{dC\&dA}[A]$ ,  $W[A]$  to 0;
2. **for**  $i := 1$  **to**  $m$  **do begin**
3.     randomly select instance  $R_i$ ;
4.     select  $k$  instances  $I_j$  nearest to  $R_i$ ;
5.     **for**  $j := 1$  **to**  $k$  **do begin**
6.          $N_{dC} := N_{dC} + \text{diff}(\tau(\cdot), R_i, I_j) \cdot d(i, j)$ ;
7.         **for**  $A := 1$  **to**  $a$  **do begin**
8.              $N_{dA}[A] := N_{dA}[A] + \text{diff}(A, R_i, I_j) \cdot d(i, j)$ ;
9.              $N_{dC\&dA}[A] := N_{dC\&dA}[A] + \text{diff}(\tau(\cdot), R_i, I_j) \cdot$   
 $\text{diff}(A, R_i, I_j) \cdot d(i, j)$ ;
11.         **end;**
12.     **end;**
13. **end;**
14. **for**  $A := 1$  **to**  $a$  **do**
15.      $W[A] := N_{dC\&dA}[A] / N_{dC} - (N_{dA}[A] - N_{dC\&dA}[A]) / (m - N_{dC})$ ;

Figure 2.16: Pseudo code of the RReliefF algorithm (Adapted from (Robnik-Šikonja and Kononenko, 2003))

### 2.3.5 | Summary

To summarize, several modeling methods have been discussed. Linear methods such as ordinary least squares, regularized least squares, and mixed effect linear models have good explainability, but they can be expected to have a bad performance. Tree-based learning methods such as decision trees, random forests, and XGBoost provide more complex methods that keep some interpretability but have better performance. The deep neural networks and LSTM networks are not explainable enough for this research but might perform better on complex problems with large samples. kNN is a very explainable model, which can take nonlinearities and interactions into account. Uncertainty quantification can be implemented for OLS, random forest, XGBoost, kNN,

DNN, and LSTM models, but the corresponding methods have very different explainability. Table 2.2 also briefly summarizes the discussed methods.



Model	Model type	Advantages	Disadvantages
OLS	Linear	Explainable model Explainable results Feature significance testing	Only linear relations Limited interactions Prone to overfitting
Regularised LS	Linear	Explainable model Explainable results Feature selection with regularization	Only linear relations Limited interactions
Mixed effect model	Linear	Explainable model Explainable results Possibly improved predictive performance	Only linear relations Limited interactions No feature selection method
Decision trees	Tree-based	Explainable model Explainable results	High variance Lack of smooth surface representation Bad for additive structures
Random Forest	Tree-based	Good predicted performance Explanation via Shapley values Interpretation via feature importance	Inner workings less explainable
XGBoost	Tree-based	Good predicted performance Explanation via Shapley values Explainable via feature importance	Inner workings less explainable
DNN	Neural network	Excellent performance on large data set Limited explanation via Shapley values	No layman's explainability of model Results not explainable
LSTM	Neural network	Potential for time series input	No layman's explainability of model Results not explainable
kNN	kNN	Explainable model Explainable results via examples Uncertainty prediction method	Basic model that might be unable to describe all patterns

Table 2.2: Summary of the models discussed in this section.



## Methods

To find the best prediction method for future model-based KPI values, we studied the application of nine predictive machine learning models. These models were implemented for the values of the SciSkill and Estimated Transfer Value (ETV) as case studies. The training of the models was mainly aimed at improving predictive performance, but the explainability and uncertainty quantification were also important for the case studies and were considered when determining the best model.

### 3.1 | Data

The data used for this research was split into two datasets for the two case studies. The two datasets contain player information and the development of the player KPIs in the subsequent year. The datasets will be used to train models that predict the development of these football player KPIs in the subsequent year.

#### 3.1.1 | SciSkill

In the first case study, a dataset concerning the SciSkill, a metric for general player performance, was obtained. The dataset was obtained from a temporal data set that consisted of approximately 13 million data points corresponding to each game of a player. These data points were irregularly placed in time. To obtain monthly data, these data points were transformed into monthly data points. This was done by taking the average value within a month if multiple values were available. Some months contained no games due to seasonable breaks, injuries of a player, or players not being

selected for games. For these months, the previous available SciSkill value was filled in if the previous game was less than 6 months earlier.

To increase the quality of the data points, the monthly data set was then filtered on players who played at least 20 games and on whom at least 2 years of data was available. These filtering actions were used to ensure that the SciSkill values of these players had converged to their actual values. Data points that contained missing values were either deleted or filled in by an estimate depending on the feature. This resulted in the data set used for the case study to predict the future values of the SciSkill. A full list of features can be found in section E.1 in Appendix E.

The resulting data set contained information on 80,568 football players in the years 2012 up to 2023, which is on average 47.6 data points per player, corresponding to approximately 4 seasons of data per player. The distribution of the data points over the years is shown in Figure 3.1. The data available for this study consists of 3,834,539 data points each representing the state of one player in one month with 86 different features and the dependent variable. The dependent variable is the development of the SciSkill in the subsequent year. This is defined as the difference in SciSkill between the SciSkill value one year in the future minus the current value.

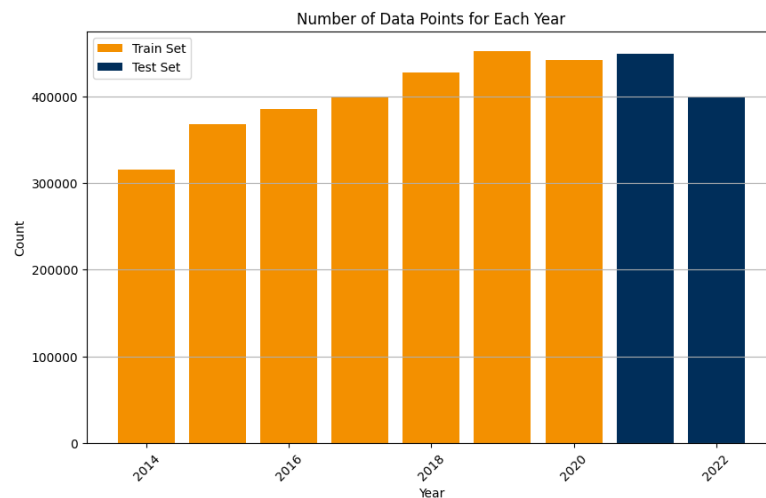


Figure 3.1: The distribution of the data points for the SciSkill case study over the years.

### 3.1.2 | Estimated Transfer Value

In the second case study, the future values of the Estimated Transfer Value of football players were investigated. This data set was obtained by combining the half-yearly

data points with Estimated Transfer Value with the data set for the SciSkill case study. A subset of features was used and some features were combined. In this way, many highly correlated variables were avoided. By reusing the data set of the SciSkill problem, the filters on the number of games and the minimal number of years of data were kept the same. A full list of features can be found in section E.2 in Appendix E.

The data set consists of 413,177 half-yearly data points for 60,175 football players containing 58 features and the dependent variable. The data set covers the years 2016 up to 2021 with on average 6.87 data points per player. This corresponds to approximately 3.5 seasons of data. For this case study, the dependent variable is the development of the Estimated Transfer Value in one year. Similarly as in the SciSkill case study, this is defined as the the ETV value one year in the ahead minus the current ETV value.

The distribution of the data points over the years is visualized in Figure 3.2. The number of data points is slightly increasing throughout time. Whereas the SciSkill problem consisted of data starting from the year 2014, the first data point from the ETV model was in the year 2015. As some necessary features are based on information on the ETV values from one year ago, the first data points to consider were those of the year 2016. Next to that, the data of the ETV model was different for the year 2022 due to a change in the model. That year was, therefore, not considered in this case study. This means that the data for the ETV case study contained data points from the years 2016 up to 2021.

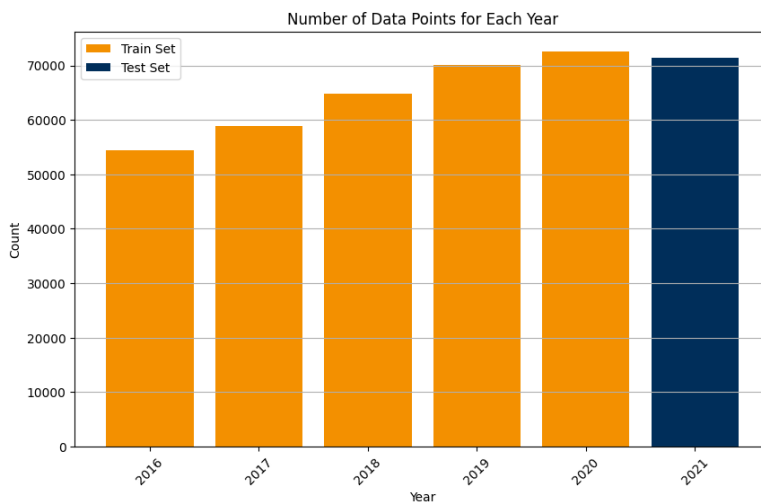


Figure 3.2: The distribution of the data points for the ETV case study over the years.

## 3.2 | Validation

A good prediction model was defined in chapter 1 as a method with a high accuracy with explainable predictions, an interpretable model, and with an uncertainty quantification method. In order to assess the predictive accuracy of the models, the following validation methods have been applied.

### 3.2.1 | Train/test splits

The models were trained on the training set and the test set was used to determine the quality of the model. Two types of train/test splits were applied in the current thesis. The main test set of this study was obtained by taking a train/test split as visualized in Figure 3.3. This split was carried out to estimate the quality of the predictions of the different models and was not done using random sampling but by applying a time-dependent split. The data points corresponding to the years up to 2020 were considered as a training set. The data points corresponding to the later years were considered to be the test set. These test sets were used to compare the predictive quality of the models.

A second test set was defined to study how the loss estimates develop through time. This split was performed using stratified sampling on the SciSkill values, age, starting date of the career, length of the available data window, player position, and the number of matches played for each player. In this way, a representative test set with unseen data of the whole population of football players was obtained. For each case study, the loss estimates were calculated for the best two models. It also contains the RMSE score when no change in SciSkill is predicted. This indicates the total change and might help in explaining patterns. It gives information about the development of loss estimates through time that can be used to put the estimates of the other test set into context.

### 3.2.2 | Choice of loss functions

Three interesting loss functions are the root mean square error (RMSE), the mean absolute error (MAE), and the  $R^2$ . Note that the  $R^2$  is not a loss function itself, but can be transformed into one by taking the negative  $R^2$ . The considered loss functions can be

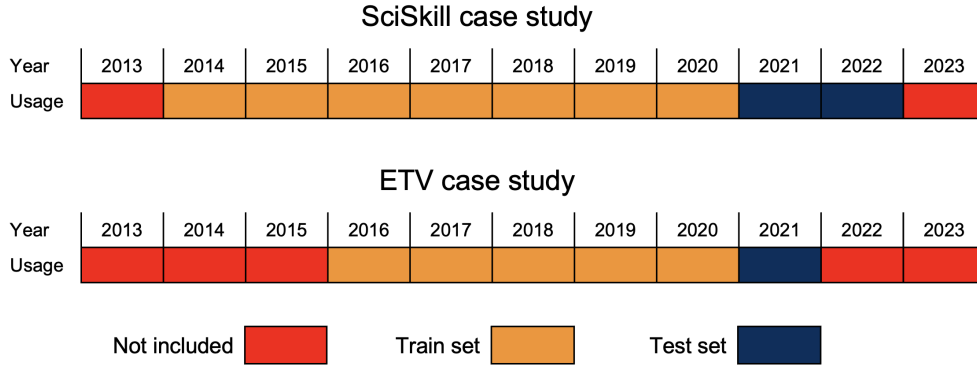


Figure 3.3: A visualization of the train and test set for both use cases.

defined as follows.

$$\text{MSE}(Y, \hat{Y}) := (Y - \hat{Y})^2 \quad (3.1)$$

$$\text{MAE}(Y, \hat{Y}) := |Y - \hat{Y}| \quad (3.2)$$

$$R^2(Y, \hat{Y}) := 1 - \frac{(Y - \hat{Y})^2}{(Y - \bar{Y})^2} \quad (3.3)$$

In this case,  $\hat{Y}$  is the estimate of the true value  $Y$ , and  $\bar{Y}$  is the sample mean. The root mean square error is the root of the estimated or expected value of the mean square error.

For the use cases of the current thesis, players with good values for the KPIs are the most interesting as these will be world-class football players. It is, therefore, desirable that the models perform best on these data points which can be considered as outliers of the data set. The error on these data points will be larger and should be punished more. As the RMSE punishes large deviations from the actual value more than the MAE, the RMSE is favored. The MAE was still calculated as the comparison with the RMSE might provide useful insights in the performance of the models.

It was proven in A.1.1 that there exists a strictly monotone function between the RMSE and the  $R^2$  provided that they are applied to the same dataset. These loss functions therefore contain the same information, but present it in a different way. Because differences can be more clear in the  $R^2$  and both loss functions have different interpretations, both were estimated in this thesis.

## 3.3 | Models

For each model, feature selection was applied using a method suited to the specific model. The hyperparameters of the models in this research were tuned with a Bayesian optimization algorithm as described in section 3.4.

### 3.3.1 | Linear models

The first model implemented was the ordinary least squares regressor (OLS). It was implemented using the 'OLS' function from the statsmodels package in Python by Seabold and Perktold (2010). Feature selection was performed by applying backwards feature selection. As not all features are normally distributed, the assumptions for significance testing in the OLS model do not hold. This means that a significance threshold of 0.01 does not correspond to a 1% probability of the  $H_0$  hypothesis being incorrectly rejected. Nonetheless, it provides a threshold of the minimal required importance of the features.

Via trial and error on the training set, the best significance threshold found was 0.0001 for the SciSkill case study and 0.001 for the ETV case study. These values are chosen smaller than the more classical values of 0.01 or 0.05 because of the large data sets, the large number of features, and the fact that the assumptions of normality were not met. No additional interaction or polynomial terms were considered because of the high computational costs.

The second model implemented was the linear lasso model, which is a regularized linear regression method as described in subsection 2.3.1.2. The implementation used the coordinate descent implementation of the function 'Lasso' in the scikit-learn package by Pedregosa et al. (2011) which follows the algorithms presented in (Friedman et al., 2010; Kim et al., 2007). The feature selection was applied by training a lasso model to the training data and selecting the nonzero features. Subsequently, a second lasso model was fitted to obtain the final model. Similarly, as with OLS, no interaction terms were considered for the lasso models.

The last linear model was the linear mixed effect (lme) model. It was implemented using the 'mixedlm' function of the statsmodels package by Seabold and Perktold (2010), which follows (Lindstrom and Bates, 1988). This model can contain both fixed effects, normal features, and random effects, variables that describe the random influence on the relation between the dependent variables and the fixed effects. The random effects are frequently used to describe the effects of different subjects on the relationship. In this study, the first nationality of a football player was used as the random effect for



both the SciSkill use case and the ETV use case. This was done because it provides the possibility to identify possible discrimination based on nationality by the models and because trial and error found that it resulted in good predictive performance compared to other considered quantities. The feature selection was performed by taking the 20 features that had the highest feature importance for the lasso model.

### 3.3.2 | Tree-based models

In addition to the linear models, tree-based models were applied to the prediction problems. First, a decision-tree model was implemented using the function ‘DecisionTreeRegressor’ in the scikit-learn package by Pedregosa et al. (2011) which follows the methods as presented in (Breiman et al., 1984). Feature selection was performed by adding noise variables as features and training the models. For each feature, the feature importance was subsequently computed, and features that had a higher feature importance than the noise variables were selected. A final model was then trained on the selected feature subset.

In a similar fashion random forest regressors were trained. The function ‘RandomForestRegressor’ from the scikit-learn package by Pedregosa et al. (2011) follows the algorithms presented in (Breiman, 2001; Geurts et al., 2006). As with the decision tree, feature selection was performed by adding noise variables as features and training the data, computing the feature importance, and selecting features with higher feature importance than the noise variables. The final model was then trained on the feature subset.

The last implemented tree-based model considered in this thesis was XGBoost. This is a gradient boosting algorithm introduced and implemented by Chen and Guestrin (2016). For hyperparameter tuning, the rule of thumb described by Brownlee (2018) was applied, where the number of trees is first fixed at a value around 100 and the other hyperparameters are tuned. The number of trees was afterward tuned keeping the other hyperparameters fixed. In this way, the hyperparameter tuning was split into two parts. Similar to the other tree-based methods, feature importance can be calculated using an XGBoost model, which can be used for feature selection when noise variables are added. Feature selection is thus performed by training the model on the features with added noise features and by selecting the features with higher feature importance than the noise variables.

### 3.3.3 | kNN models

After the implementation of the linear and tree-based methods, the question was raised whether it was beneficial to view the predictive problem as a time series or a stochastic process. Several time-series methods were considered for the use case of the current thesis such as the Prophet model by Taylor and Letham (2017), the vector autoregressive processes (VAR) by Lütkepohl (2005), and a time-series-based kNN implementation. The most promising method seemed to be a kNN implementation that takes the time series into account next to several additional features. This was due to the explainability of the method, the fast computation, and the limited ability of the other methods to generalize patterns of football players' development.

In order to improve the predictive quality of the kNN model, distance-weighted rules as described by Dudani (1976) were implemented. These methods take a weighted average over the  $k$  neighbors and assigns stronger weights to neighbors that are closer to the feature vector for which a prediction should be made. The weights were calculated using the inverse distance or the distance after min-max scaling within the selected neighbors. In this way, it assigns heavier weights to data points that are more similar, which improves the quality of prediction.

Using this adaptation, a study was performed to introduce feature weights to the kNN model. The goal of this study was to introduce feature weights that improve the quality of the predictions by assigning more importance to more important features. This study and its results are described in Appendix B.

Based on the results of this study, three implementations of kNN regression models were applied. All of the kNN models for the SciSkill case study contained the SciSkill values of the last 12 months, the age, the month, and the number of months since the last game of a player as features. These added features were selected because of their simplicity in computing their values. This makes it more suitable for application outside of this research.

For the application of the kNN models on the ETV models, the last values of the ETV, the SciSkill, and the SciSkill potential were used including the differences with the historical data points of 6 and 12 months ago. The SciSkill potential is a metric obtained after applying a more basic model to estimate the Sciskill 6 months in the future iteratively. The largest value obtained can then be considered as the SciSkill potential as it is the highest SciSkill that the player potentially is expected to attain. Additionally, the age, the month, and the number of months without a game were taken as features.

All kNN models were implemented using the faiss package of Douze et al. (2024) with hierarchical navigable small world indexing which builds a graph based on the similarity between data points within the data set. This package provides a significant speedup for predicting values compared to the kNN implementation of scikit-learn Pedregosa et al. (2011) which follows Goldberger et al. (2004).

The first kNN that was implemented was the classical kNN model. The features are weighted equally in this algorithm. The number of neighbors  $k$  was tuned as a hyperparameter using the Bayesian optimization algorithm described in section 3.4.

Second, a kNN was implemented which used the Mahalanobis distance metric as described by McLachlan (1999) instead of the traditional Euclidean distance. This implementation was chosen due to the fast computing time and the high predictive performance that were found in Appendix B.

The Mahalanobis distance can be computed using

$$\|x_i - x_j\|_M = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}$$

which is equivalent to the Euclidean distance in the transformed space obtained by the transformation  $x' = (x - \hat{\mu})\hat{\Sigma}^{-1/2}$ . To implement this algorithm, the covariance matrix  $\Sigma$  was estimated using the element-wise estimation of the covariance matrix,  $\text{cov}(X_i, X_j) = \frac{1}{n-1} \sum_{l=1}^n (X_{l,i} - \bar{X}_i)(X_{l,j} - \bar{X}_j)$ . The calculation of  $\hat{\Sigma}^{-1/2}$  was then performed by applying a Choleski decomposition and calculating the inverse of this decomposition matrix. Note that the true underlying  $\Sigma$  can be chosen as positive definite by removing constant features and assuming that no linear dependencies between the features occur. This makes it likely that the estimation  $\hat{\Sigma}$  was also positive definite if enough data points are available. The number of neighbors  $k$  of this obtained kNN algorithm was then determined using hyperparameter tuning with the method described in section 3.4.

Lastly, a kNN algorithm with feature weights obtained by the RReliefF algorithm (Robnik-Šikonja and Kononenko, 2003) was implemented. This method was implemented for classification purposes, but there was a lack of available Python implementations for regression purposes. Therefore, a new vectorized implementation of the algorithm was created in Python following the algorithm described in Figure 2.16. The measure of similarity between two data points  $I_i, I_j$  was chosen to be  $d(i, j) := \frac{\|I_i - I_j\|_2}{\sum_l \|I_i - I_l\|_2}$  where  $\|I_1 - I_2\|_2$  denotes the Euclidean distance between two data points  $I_1$  and  $I_2$ . The number of neighbors  $k$  was again viewed as a hyperparameter. A slightly more elaborate explanation of the implementation of the RReliefF algorithm can be found in subsubsection B.1.3.4 in Appendix B.

## 3.4 | Hyperparameter tuning

Most of the models considered for this study have hyperparameters that need to be tuned when applying the model to the predictive problems in the case studies. These hyperparameters were tuned using a Bayesian optimization algorithm which used an adjusted time series cross-validation method as the objective function.

### 3.4.1 | Adjusted time series cross validation

Similar to the year-based train/test split described in subsection 3.2.1, the cross-validation methods for the hyperparameter tuning were changed to prevent data leakage. A visualization is shown in Figure 3.4. This method prevented data leakage by splitting the data based on the year it belongs to. The first training set was then obtained by the first year while the second year was the test fold corresponding to the same first fold. The next fold was then obtained by adding the second year to the training set and taking the third year as the test set. In this way, the other folds could be created incrementally. The final error estimate was then obtained by taking a weighted average over the folds, where the weights are the number of data points considered in each training set.

	2014	2015	2016	2017	2018	2019	2020
Fold 1	Train set	Test set					
Fold 2	Train set	Train set	Test set				
Fold 3	Train set	Train set	Train set	Test set			
Fold 4	Train set	Train set	Train set	Train set	Test set		
Fold 5	Train set	Train set	Train set	Train set	Train set	Test set	
Fold 6	Train set	Train set	Train set	Train set	Train set	Train set	Test set



Train set 
 Test set 

Figure 3.4: A visualization of the division of the train and test set for the adjusted time series cross-validation method for the SciSkill case study.

This adjusted version of cross-validation could be expected to be more conservative with respect to overfitting because some folds with a small amount of training data were considered. This means that a small bias favoring hyperparameters that work well for smaller data sets was introduced by using this adjusted time series cross-validation.

On the other hand, the variance of the test error estimate was reduced by taking the weighted average over the different folds.

In order to study the influence of this choice, a small study was performed in section 4.3 in which test losses were calculated using a normal train/test split, a year-based train/test split, a cross-validation method, and a time series adjusted cross-validation method. To this end, the RMSE estimates were resampled using bootstrap for the OLS and decision tree models. 20 bootstrap samples were taken for the SciSkill case study and 50 for the Estimated Transfer Value case study. The mean and standard deviation of these samples were then calculated to analyze the behavior of the different validation methods.

### 3.4.2 | Bayesian optimization algorithm

For the optimization of the hyperparameters of the models, a Bayesian optimization algorithm was implemented to minimize the estimated loss values. This method was chosen as the Bayesian optimization algorithm with a Gaussian process generally provides the best results for hyperparameter tuning (Hutter et al., 2014). Next to that, the Bayesian optimization algorithm provided possibilities to estimate the values of the objective function, which gives visualization possibilities to explain the chosen hyperparameters.

The Bayesian algorithms were implemented using the Scikit-Optimize package provided by Louppe and Kumar (2016). The lower and upper bounds for the hyperparameters chosen for the models are shown in Table 3.1. The number of function calls was determined by trial and error with the requirement that the computing time remained feasible. The obtained number of function calls is shown per model in Table 3.2. The Bayesian algorithm used for the minimization of the loss values had an acquisition function that was chosen to be a function that iteratively alternates between minimizing the lower confidence bound, maximizing the expected improvements, and maximizing the probability of improvement.

Model	SciSkill		ETV	
	Feature selection	Final model	Feature selection	Final model
OLS	-	-	-	-
Lasso	25	50	50	100
LME	-	-	-	-
Decision tree	25	50	50	100

Table 3.2: (continued)

Model	SciSkill		ETV	
	Feature selection	Final model	Feature selection	Final model
Random forest	19	25	50	50
XGBoost	50	50	50	100
XGBoost (n_trees)	25	25	50	75
kNN	-	25	-	50
kNN Mahal.	-	25	-	50
kNN RReliefF	-	35	-	50

Table 3.2: The number of function calls performed for the hyperparameters for each model for the different case studies.

Model	Hyperparameter	Lower bound	Upper bound	Type	Transformation
OLS	-	-	-	-	-
Lasso	$\alpha$	$10^{-6}$	$10^2$	float	$x \mapsto 10^x$
LME	-	-	-	-	-
Decision tree	max depth	1	1,000	int	-
	min. samples per leaf	1	$2^{20}$	float	$x \mapsto 2^x$
Random Forest	n_trees	20	250	int	-
	max. depth	1	25	int	-
	min. samples per leaf	1	$2^{20}$	int	$x \mapsto 2^x$
	max. features	0.05	1.0	float	-
	max. samples	0.05	1.0	float	-
XGBoost	n_trees	20	1,000	int	-
	$\eta$	$2^{-10}$	1	float	$x \mapsto 2^x$
	subsample	0.01	1	float	-
	max. depth	1	15	int	-
	min. child weight	1	7	int	-
	colsample by tree	0.01	1.0	float	-
	colsample by level	0.01	1.0	float	-
	$\lambda$	$2^{-10}$	$2^{10}$	float	$x \mapsto 2^x$
	$\alpha$	$2^{-10}$	$2^{10}$	float	$x \mapsto 2^x$
kNN	$k$	1	50	int	-
kNN Mahal.	$k$	1	50	int	-
kNN RReliefF	$k$	1	50	int	-

Table 3.1: The specification of the considered values of the hyperparameters of each model.





## Results

In this chapter, the results of the model training for the two case studies are discussed. The results concerning the training of the individual models were studied to assess whether the models were correctly trained. As these do not provide direct implications for the main research question, these are described in Appendix C.

### 4.1 | SciSkill case study

#### 4.1.1 | Predictive performance on test set

The values of the loss functions of the different models on the test set are visualized in Figure 4.1. It shows that the RSME is generally between 4.4 and 4.8, the MAE between 3.1 and 3.4, and the  $R^2$  between 0.25 and 0.4. The figure does not show large differences between the RMSE and MAE values. However, the differences in the  $R^2$  are more evident. As the differences are better visible and the  $R^2$  is equivalent in this situation to the RMSE as proven in section A.1, these are used interchangeably.

Figure C.4 shows the distribution of the 1000 bootstrap samples of the loss values on the test set for the OLS model. These bootstrap samples were used to calculate a bootstrap estimation of the standard deviation of the test loss values. These models show that the distribution of the points appears to be symmetric. Next to that, they seem to have a single mode around the mean value. The distribution of bootstrap samples of the other models showed similar behavior. Therefore, the heuristic assumption is made in the current thesis that a difference in the performance of two models of more than two estimated standard deviations can be considered a significant difference.

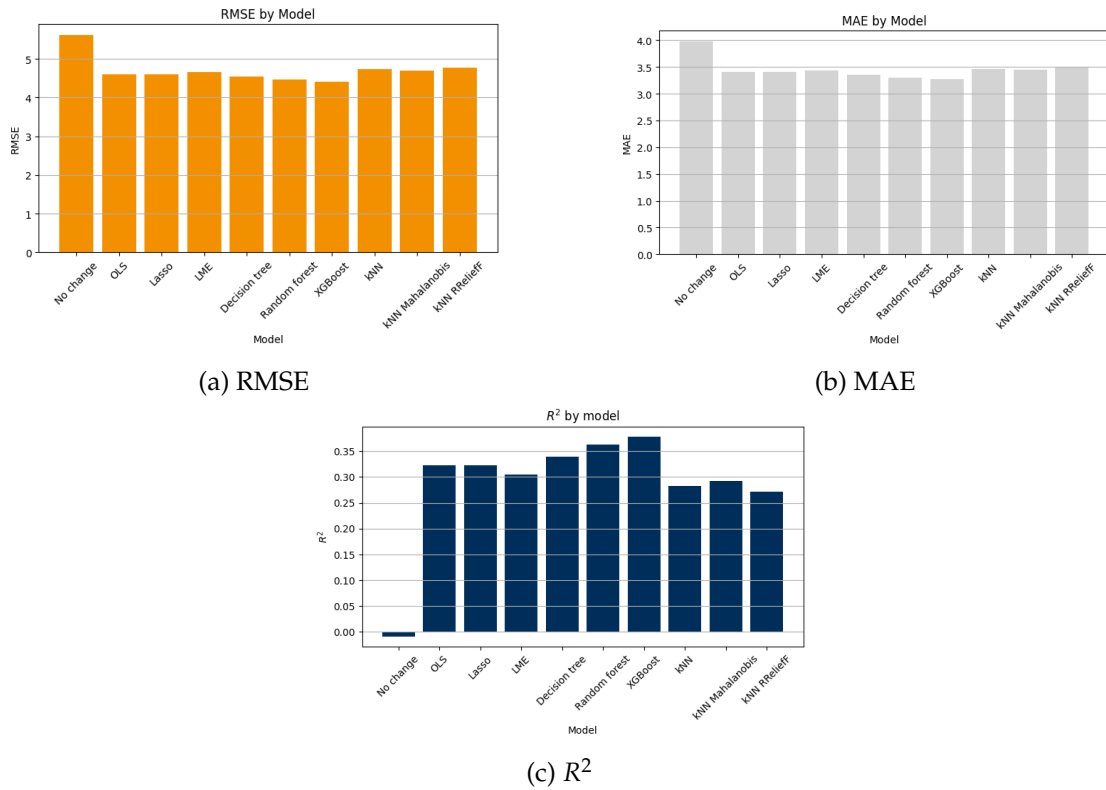


Figure 4.1: The values of the test loss estimates for the different models in the SciSkill case study.

The largest standard deviation of the RMSE losses is 0.00607, while for the MAE and the  $R^2$  it is 0.00359 and 0.00122 respectively. This means that differences in the RMSE should be around 0.012 to be significant, for the MAE 0.0072, and for the  $R^2$  0.0024. This research will use these thresholds to consider two loss values as different.

It can be seen in Figure 4.1c that all models provide more insight than assuming that every football player is constant in their SciSkill value, indicated with 'No change'. Moreover, all  $R^2$  values are significantly larger than zero. This means that all models show improvement over assuming no change or assuming the average change of the whole sample.

The loss values in Figure 4.1c are the worst for the kNN-based models for all loss functions. This is probably due to their simple and local nature. The kNN-based models are, therefore, not considered the best models for this prediction problem with respect to predictive accuracy.

The results also indicate that the tree-based models outperform the linear models.

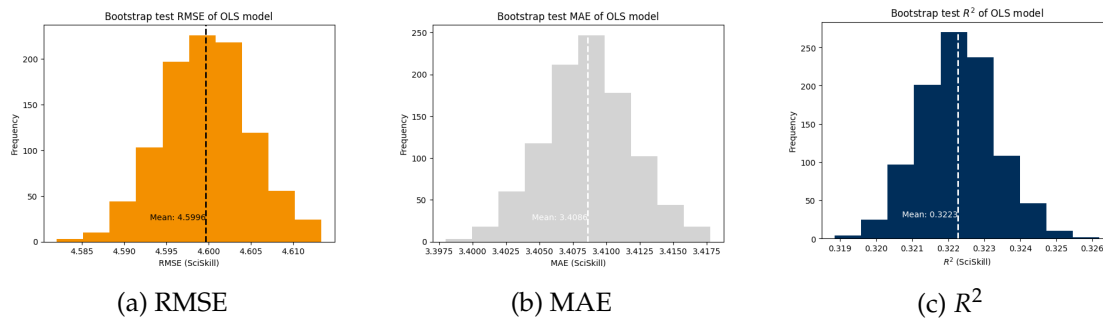


Figure 4.2: Histograms of the distributions of the 1000 bootstrap samples for the different estimated loss functions of the OLS model in the SciSkill case study. The bootstrap mean is indicated by a vertical line.

This is probably due to the fact that the true relation includes non-linear or interaction terms with the features, which are not taken into account by these models. Consequently, the linear models are not considered the most accurate models for this prediction problem.

This means that the tree-based models have the best predictive performance. The decision tree has the worst predictive accuracy, which could be expected as it is a simple method. It is, however, surprising that it outperforms all linear models. This is probably due to a large amount of data and the large size of the tree, combined with the presence of nonlinearity or interactions.

The best-performing models are the XGBoost and the random forest models. The difference between the RMSE for these methods is 0.052038, which is significant. The same holds for the difference between the random forest and decision tree models, which is 0.081519. As this difference is also significant, it can be concluded that the differences within the tree-based models are significant. The same also holds for the MAE and the  $R^2$ . This means that the model with highest predictive accuracy is the XGBoost model, followed by the random forest model. The decision tree model is the third best.

### 4.1.2 | Predictive performance on age groups

For the use-case of SciSports, the performance of the models on certain subsets of the players is interesting. It is, for instance, important that the models perform well on young players, as the estimates of the resulting model of the current thesis are expected to be used the most for this subset.

The RMSE of the players of each age were estimated on the test set. This resulted in the visualizations in Figure 4.3. The XGBoost and random forest models perform better for all ages. For ages above 25, the XGBoost clearly seems to have better loss values compared to the random forest model, whereas the differences are less evident for the lower ages. The better general RMSE loss of the XGBoost model is probably caused by the better performance on the subset of the better players.

The results also show that the bad performance of the kNN models are mostly caused by the larger RMSE on the lower ages. Figure 4.4 gives the distribution of the data points per age. There are less data points corresponding to the lower ages. Because the kNN models are local models, the small amount of data points make it harder for the models to infer the right patterns as there are not enough data points to do this. The kNN models, therefore, perform worse because of the low number of data points available for the younger ages.

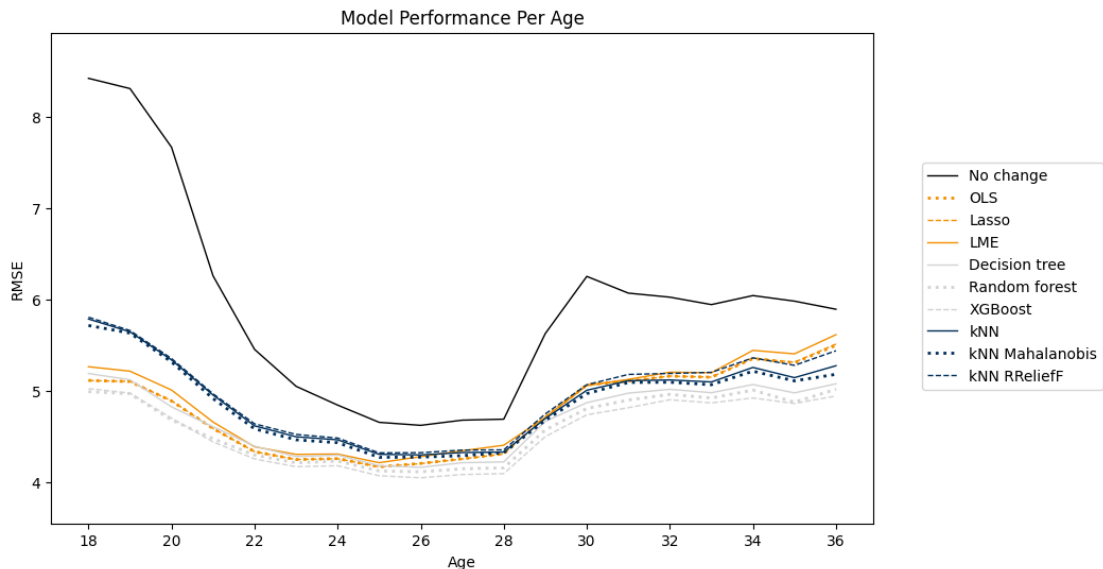


Figure 4.3: The estimated test RMSE for each age group for the different models.

In general, Figure 4.3 shows that all models perform best in the age group of 24 up to 28. Figure 4.4 provides the number of data points in the data set for each age. The group of players with ages from 24 up to 28 is also the group of players with the most data points. The estimations of the younger and older ages are, therefore, based on fewer data points. On top of that, the performance of a football player is often more stable for ages from 24 up to 28, which makes it easier to estimate. This explains why the performance of these age groups is worse than for the age group of 24 up to 28.

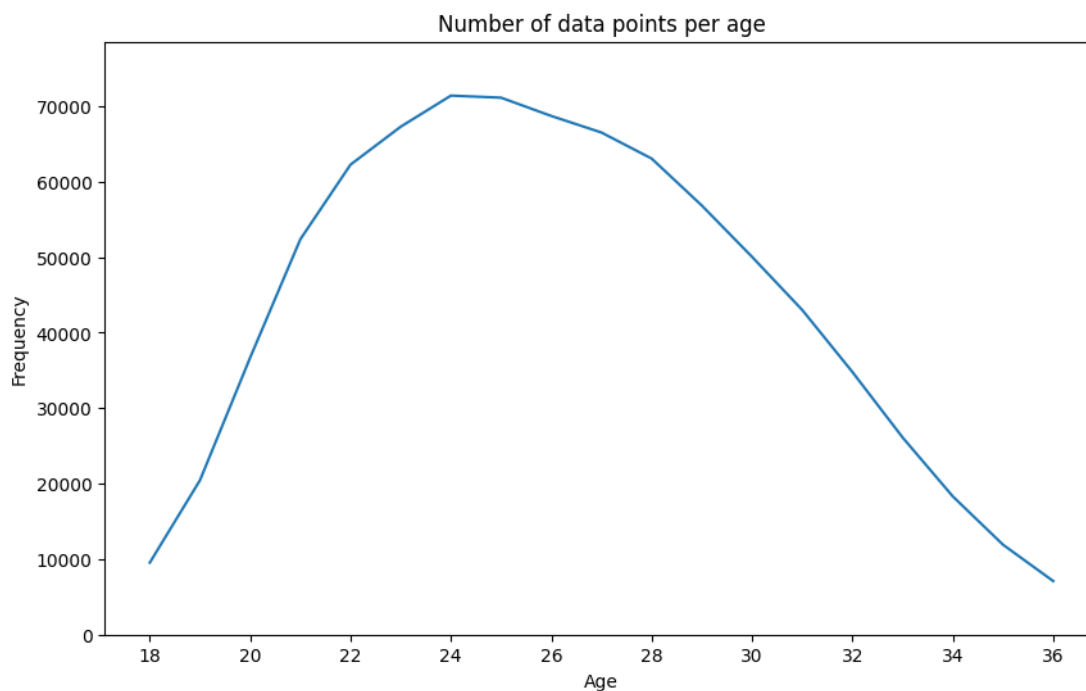


Figure 4.4: The number of data points for each age group in the training and test set.

When the performances of the individual models are assessed, they clearly show that the performances of the XGBoost and random forest models is better for the young ages compared to the other models, whereas the XGBoost model performs better for the later ages. Because the performance on the younger ages is the most important for the use-case, the XGBoost and random forest are the favorable models.

### 4.1.3 | Predictive performance on subsets of outliers

Additionally, it is important to look at the model performance for the subsets of players with a high SciSkill value, a large decrease in SciSkill, and a large increase in SciSkill over a year. These cases are the most interesting for the users of the model but are outliers of the dataset at the same time. This creates a challenging prediction problem.

The performance of the models on the data points with SciSkill values larger than 100 was studied, as well as the data points with a decrease or increase of at least 10 in their SciSkill within a year time. These results are shown in Figure 4.5 and indicate that the XGBoost and random forest models perform best on the data points corresponding to players with a high SciSkill or a large decrease. The XGBoost and random forest models are also best at predicting large increases, although the XGBoost appears to

predict more accurately.

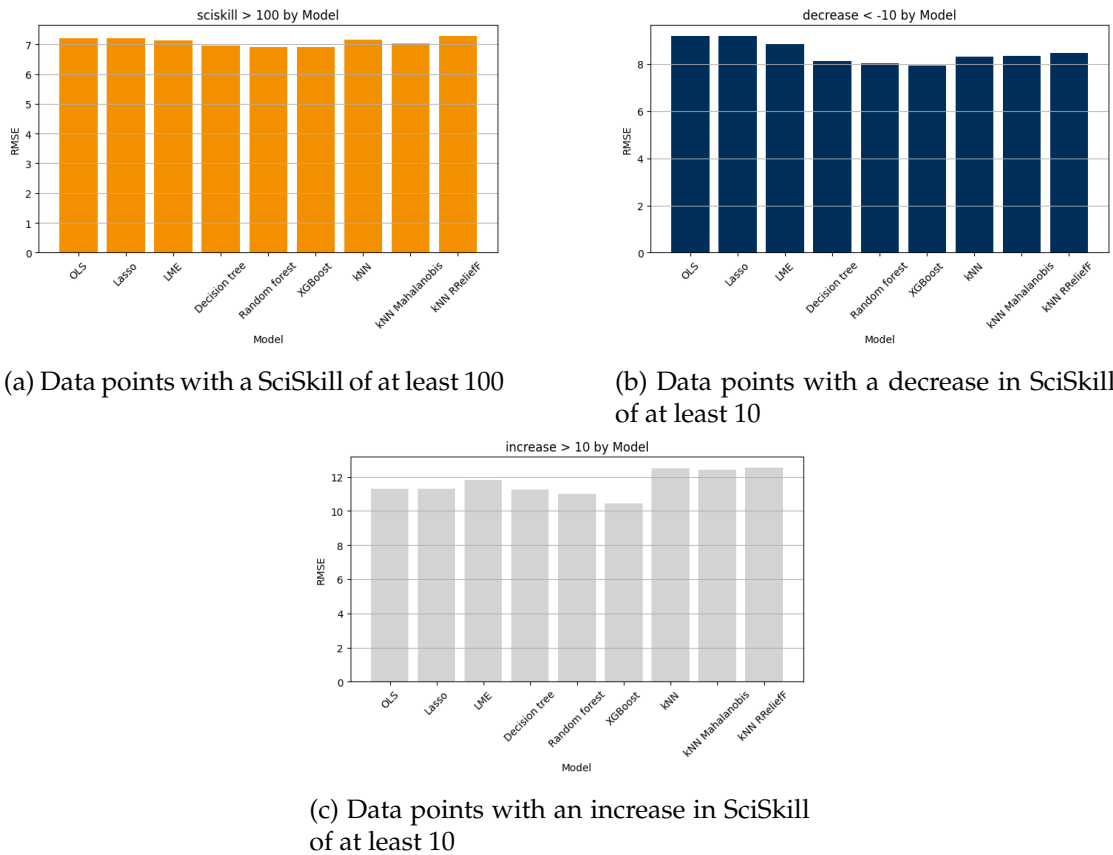


Figure 4.5: The RMSE estimates on different subsets of the test data in the SciSkill case study.

The XGBoost and random forest models perform well on the subsets of players with a large decrease or increase. These correspond to the extreme values for the dependent variable. This could be expected as the XGBoost model had good values for the RMSE, which is dominated by the large errors that typically occur at the extreme values of the dependent variable. Consequently, the performance on these subsets is in line with the results shown in Figure 4.1.

#### 4.1.4 | Summary predictive performance

To summarize the results on the predictive performance, the XGBoost model predicts the most accurate with respect to the RMSE, MAE, and  $R^2$ , followed by the random forest model. It was found that the XGBoost and random forest models performed best

for the younger players and that the XGBoost model had the most accurate predictions for the older players. A study of interesting groups of outliers showed that the XGBoost model performs best for predicting large decreases or increases in SciSkill. For the data points with a high SciSkill or a large decrease, the random forest and the XGBoost models had the lowest loss values. These results mean that the XGBoost and random forest models are the best models for this case study with respect to predictive accuracy, with the XGBoost being slightly favorable.

### 4.1.5 | Predictions per year

The 5% test set of this study contained a set of players that remained unseen by the models. The two most accurate models were the XGBoost and random forest models. Additionally, the loss values were determined on this data set when no change in the SciSkill was predicted. This can be viewed as a baseline model. Figure 4.6 gives the loss estimates of these two models and the case where no change in SciSkill was predicted. The results indicate that both models have a better predictive accuracy than predicting no change. Moreover, the random forest and XGBoost models had similar performance, although the XGBoost model seemed to have performed slightly better. These results, therefore, do not provide new insights into the performance of the models.

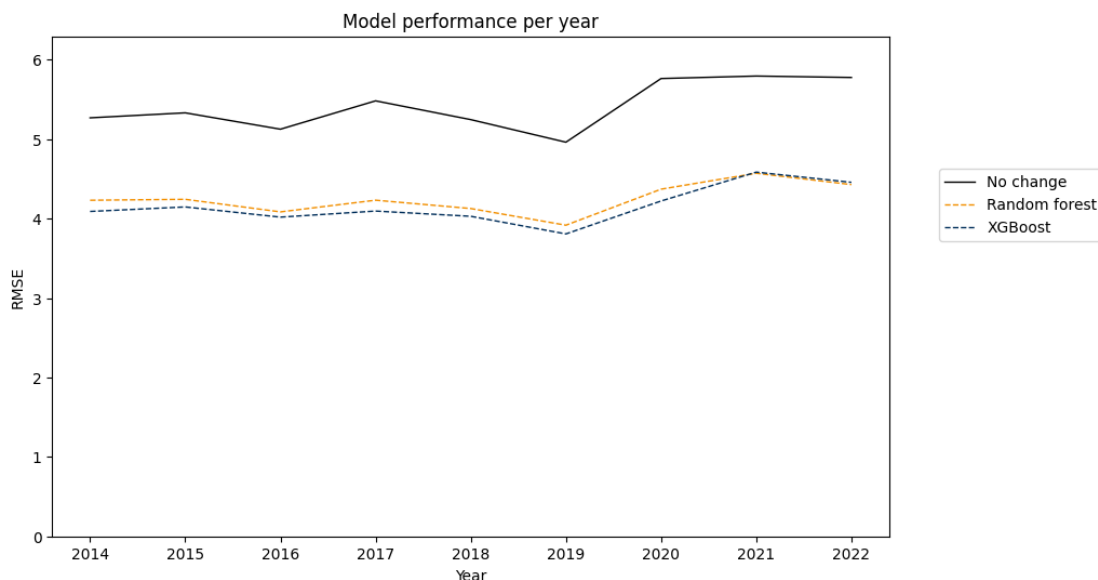


Figure 4.6: The RMSE estimates for each year on the stratified 5% test set in the SciSkill case study.

However, the RMSE scores of the models seem to be relatively the same but are a bit higher for the last three years. The line representing no change indicates an even larger difference in RMSE. This means that the distribution of the data in the last years has changed, which might be due to the addition of new players or competitions for instance. This change in distribution implies that there are temporal effects in the data, which were not included in this study. It might be interesting to include features that might influence the distribution through time, like the number of new players in the data set in a year or the total number of players. This might improve the predictions of models.

#### 4.1.6 | Feature importances

All linear and tree-based models in the current thesis had possible interpretations of feature importances. These feature importances can be studied in order to obtain insight into the most influential features for estimating how football players will develop. In order to do this, the feature importances were scaled using min-max scaling and visualized as in Figure D.14. Note that the kNN model with RReliefF features also had an interpretation of feature importance. Because the RReliefF weights only were applied to the time series features, these feature importances were not taken into account.

The results show a heatmap in which only a few features have a large importance. Additionally, it indicates that the models mostly agree on the most important features. One model that differs is the OLS model, where only age-related features seem to be important. This might be due to strong correlations between these age-related features. The XGBoost model is another model with different behavior in feature importance. Although it gives more importance to the features that are deemed most important by the other models, it gives, in general, a relatively high importance to the features that were considered unimportant by the other methods. This might be because of the ability of the XGBoost model to decorrelate features by using subsampling. The random forest also performs subsampling, but the hyperparameters selected by the Bayesian algorithm turned this off, which might have limited the working of this property. In this way, Figure D.14 provides interesting insights into the features importances of the models.

The figure also shows that age-related features are deemed important by all models. As discussed in the literature study, domain knowledge indicates that the development of a player is often dependent on age. Therefore, this is in line with what was expected.

Next to that, the current SciSkill value was an important feature. It can be expected



that players with a very high SciSkill do not increase in quality as they will barely increase their SciSkill by performing well, but will decrease steeply if they perform badly. It is, therefore, not surprising that this variable influences the estimates.

Another important variable is the number of months since the most recent game of a player ('previous\_zero\_months'), resistance-related features ('player\_resistance'; 'minutes\_x\_resistance\_domestic'; 'minutes\_x\_resistance\_domestic\_last\_6m'; 'mean\_resistance\_last\_6m') and the difference in the SciSkill between the player and the average of the whole team ('sciskill\_diff\_mean\_team'). The importance of these variables is not surprising as these factors are known to be of influence on the underlying SciSkill model itself. It is, therefore, logical that these features have an influence on future values.

The features concerning the difference in SciSkill with the historical SciSkill values ('sciskill\_diff\_1m\_ago'; 'sciskill\_diff\_6m\_ago'; 'sciskill\_diff\_12m\_ago') show a relatively large feature importance. However, the kNN models with time series features did not perform well. Models combining information of normal variables and historical values could, therefore, be expected to predict best.

On the other hand, it might also be the case that adding features from the linear and tree-based models to the kNN models improves the predictive quality of the kNN models. Features with possible added value are, for instance, those related to player resistance.

In short, the feature importances of the different models can be used to conclude which features are important for the prediction of the development of the SciSkill values in the next year. In this case study, age-related features, features connected to the SciSkill model, and time series information are important features. Additionally, comparing the feature importances of the models provides insight into different behaviors of the model.

## 4.2 | Estimated Transfer Value case study

### 4.2.1 | Predictive performance on test set

The loss values on the test set are visualized for the different models in Figure 4.7. The values of the RMSE are generally between 1.7 million euros and 1.8 million euros the values of the MAE are between 440,000 and 540,000 euros, and the values of the  $R^2$  are between 0.1 and 0.2. The figure does not show large differences between the RMSE, but the differences in the  $R^2$  and the MAE are more evident.

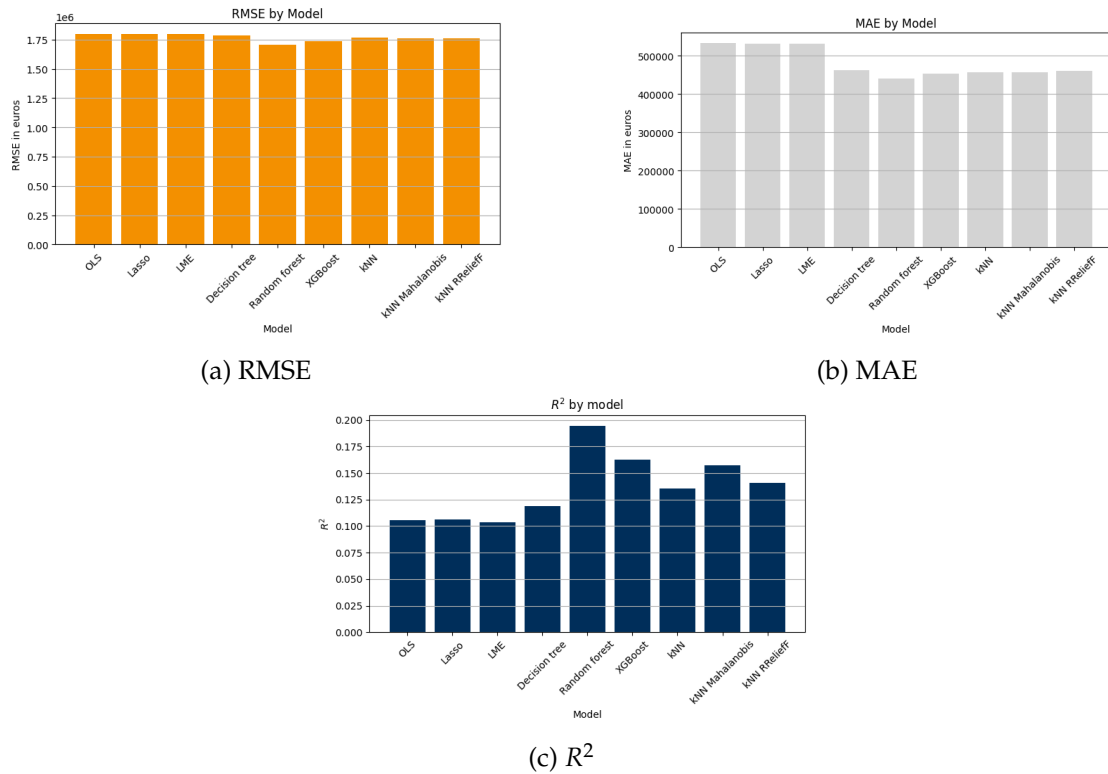


Figure 4.7: The values of the test loss estimates for the different models in the Estimated Transfer Value case study.

For each model, the bootstrap estimation was applied to approximate the standard deviation of the test losses. The distribution of these estimates is visualized in Figure 4.8, which appears to be symmetric and unimodal. Similar to the SciSkill case study, differences are assumed to be significant if they are larger than two standard deviations, corresponding to the 95% two-sided interval of the normal distribution. The largest standard deviation of the RMSE losses is 44,580 euros, while for the MAE and  $R^2$  it is respectively 6,509 euros and 0.02202. This means that differences in the RMSE should be around 89,000 euros to be significant. For the MAE this is around 13,000 euros and for the  $R^2$ , it is 0.044.

The largest differences in the RMSE are between the linear mixed effect model (1,797,849 euros) and the random forest model (1,704,637 euros). This difference is 93,212 euros, which means it is a significant difference. Similarly, the RMSE of the random forest model is significantly different with the RMSE values of the lasso and OLS models. This means that all differences in RMSE of the random forest model and the linear models are significant. However, the differences with all other models in

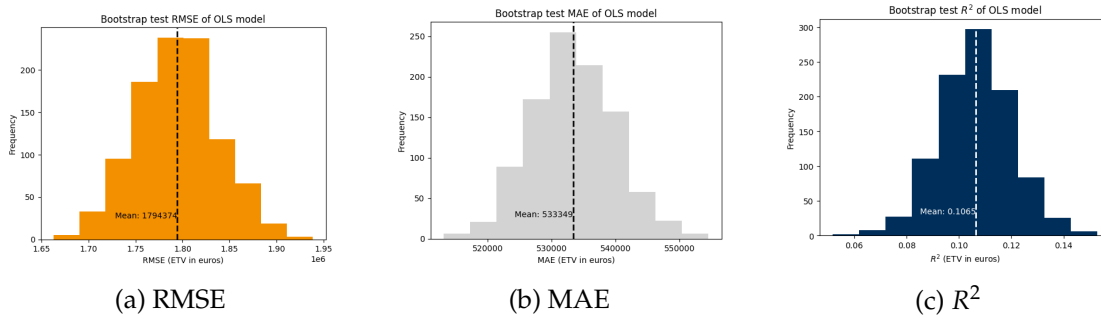


Figure 4.8: Histograms of the distributions of the 1000 bootstrap samples for the different estimated loss functions of the OLS model in the ETV case study. The bootstrap mean is indicated by a vertical line.

the RMSE are smaller than 89,000 euros and can thus not be considered as significant differences.

For the MAE on the other hand, the differences are more obvious. The random forest has the best MAE value (441,403 euros), with the second-best MAE value corresponding to the XGBoost model (453,343 euros). The difference between these MAE values is less than 13,000 and the difference is, therefore, not significant. As the differences with the models other than the XGBoost model are larger, the random forest model performs significantly better than the other models.

The differences between the linear models are less than 2,500 euros, which makes these differences insignificant. On the other hand, the differences between any linear models and any tree-based or kNN-based model are at least 68,726 euros, making these differences significant. This means that linear models perform significantly worse than the tree-based and kNN-based models with respect to the MAE.

The lasso model has a similar RMSE value to the decision tree model. At the same time, the decision tree has a lower loss for the MAE. Large errors are penalized more by the RMSE and small errors have, therefore, more influence on the MAE. Therefore, the smaller MAE loss for the decision tree means that the decision tree makes relatively better estimates on the data points with smaller errors. It is reasonable to assume that these are the average points of the data set as the largest errors are made on the outliers. It can, therefore, be concluded that the decision tree performs better on those data points.

On the other hand, the RMSE is similar for both the lasso and decision tree models. The RMSE is heavily influenced by the large errors of the model and it can, therefore, be concluded that they perform similar on points with a large error. In this context, these

points can be assumed to be the players with a high ETV and changes in ETV. It can be concluded that the lasso model and the decision tree perform similarly on the players with a high ETV or large changes in ETV.

### 4.2.2 | Performance on age groups

The RMSE for each player's age was calculated for the different models on the test set. This resulted in the visualizations in Figure 4.9. It shows that the error of the model decreases with age. Older players generally have a smaller transfer value and smaller errors will generally be made on players with smaller transfer values. Moreover, their value could be expected to be decreasing, which makes the development in the next year more predictable. It could, therefore, be expected that the error is smaller for older players.

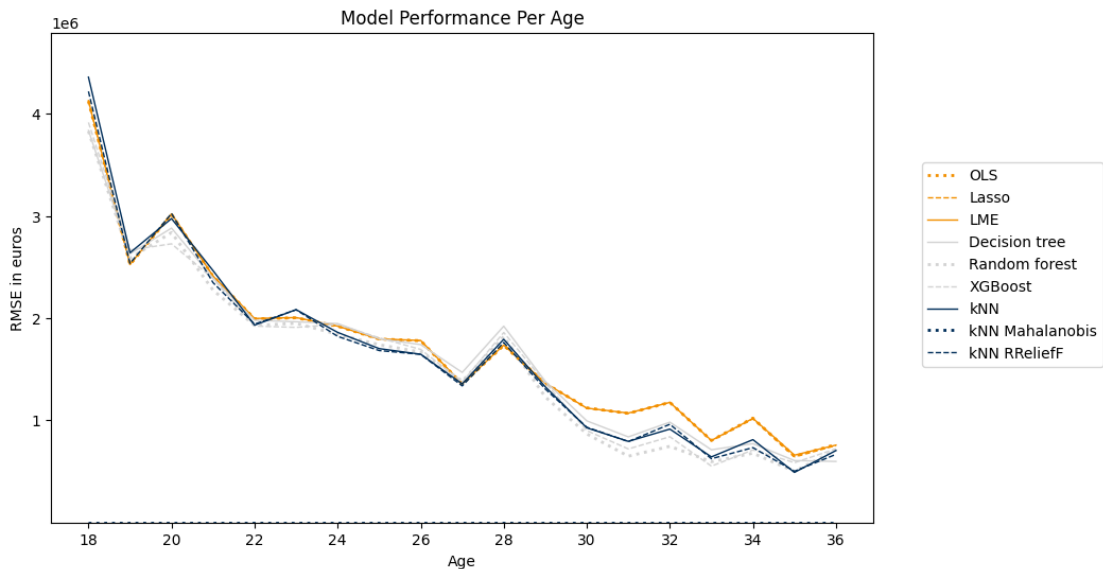


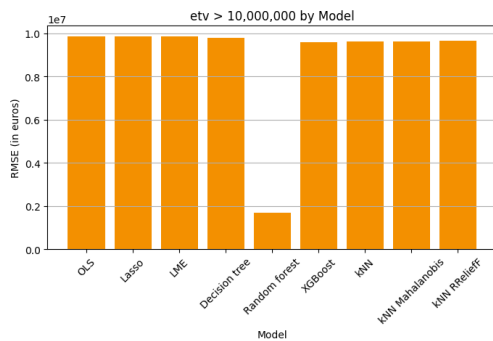
Figure 4.9: The estimated test RMSE in euros for each age group for the different models for the ETV case study.

Figure 4.9 shows that the linear models perform worse than the tree-based and kNN-based models for the ages of 30 and older. The errors on these data points are generally smaller and the linear models thus perform worse on this subset of data points with relatively small errors. The bad performance on the older players could be an explanation for the fact that the linear models perform worse for the MAE, but not much worse for the RMSE as described in subsection 4.2.1.

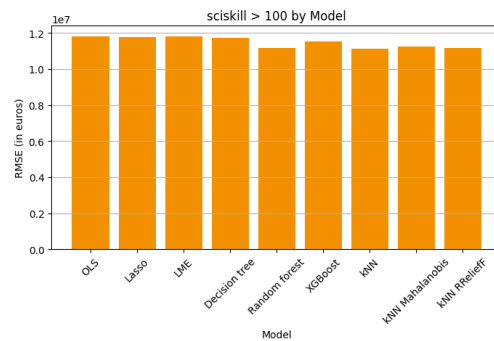
Figure 4.9 has different models being the best for many different age groups. Only the random forest model appears to be slightly better for the ages 31 up to 34. In general, the performances of the models per age do not show obvious patterns in performance.

### 4.2.3 | Performance on subsets of outliers

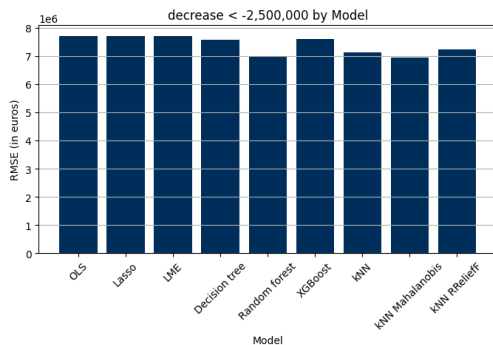
The performance of the models was also studied on different groups of outliers in the test set. These outliers were the players with an ETV of at least 10 million euros, players with a SciSkill of at least 100, players with a decrease in ETV in the next year of at least 2.5 million euros, and players with an increase in ETV in the next year of at least 2.5 million euros. The results are visualized in Figure 4.10.



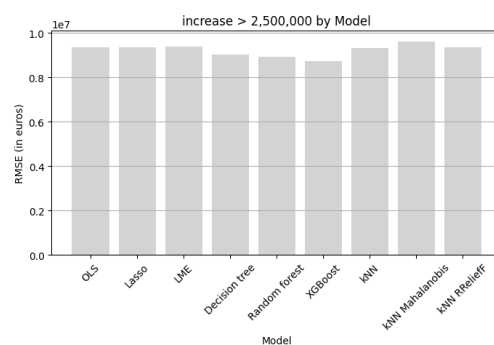
(a) Data points with a ETV of at least 10 million euros



(b) Data points with a SciSkill of at least 100



(c) Data points with a decrease in ETV of at least 2.5 million euros



(d) Data points with an increase in ETV of at least 2.5 million euros

Figure 4.10: The RMSE estimates in euros on different subsets of the data of the ETV case study.

Figure 4.10a indicates that the random forest model performs much better on the

players with a large ETV value. The error is around 5 times smaller than the other models. The players with a large ETV value are expected to also have larger changes in the ETV which are more likely to have larger errors. This would explain why the random forest model has the best test RMSE estimate on the general test set.

The random forest model has the property that it can deal with highly correlated features and data points due to the subsampling of features and data points. The XGBoost is the only other model that performs column subsampling. The tuned hyperparameter for the XGBoost model selected a larger fraction of the columns than the equivalent hyperparameter in the random forest model. This makes the random forest deal better with highly correlated features. Moreover, the random forest model is the only model that performs data point subsampling to break down dependencies within the data points. An explanation for this surprising result could, therefore, be that the random forest model was able to deal with the correlations of the features and data points that are stronger within this subset of data points.

Figure 4.10b indicates that the loss values on the subset of players with a high SciSkill tend to be large compared to the general loss values as the RMSE values are between 10 million and 12 million euros. This could be expected as this subset contains players with larger changes, which are harder to predict. It also shows that the random forest and the kNN-based models tend to perform best on the players with high SciSkill values.

A similar pattern is visible for the players with a large decrease in the ETV value albeit with smaller RMSE values and more clear differences. This might be caused by the fact that the patterns in these subsets are similar. This is reasonable because players with a high SciSkill have a high Estimated Transfer Value, which can generally only decrease over time. These subsets could, therefore, be expected to behave somewhat similarly.

A surprising result is the fact that the XGBoost model performs best on the subset of players with a large increase in ETV values as given in Figure 4.10d. The decision tree and random forest models also appear to perform relatively well on this subset. It can thus be concluded that tree-based models perform better in predicting large increases in ETV. This result might be caused by the fact that these models are able to describe nonlinear relationships and interaction terms within variables, which linear models cannot do. The differences with the kNN models can be described by the fact that the tree-based models might be able to infer more of these relationships with fewer data as the tree-based models are less flexible than the kNN models and this might help to infer patterns better on smaller data sets.

### 4.2.4 | Summary predictive performance

To summarize, the predictive performance in terms of the test RMSE loss was the best for the random forest model closely followed by the XGBoost model. The random forest was also the best model with respect to the test MAE loss. There did not appear to be a large difference in performance for certain age groups. The XGBoost model performed best at predicting large increases in estimated transfer values, although the random forest and kNN model with Mahalanobis distance performed well on the subsets of players with a large decrease in player value or a high SciSkill. Lastly, the random forest clearly outperformed the other models on the subset of players worth more than 10 million euros. Consequently, the model with the best predictive performance is the random forest and the second-best predictive performance was achieved by the XGBoost model.

### 4.2.5 | Predictions per year

A 5% test set was created described as in subsection 3.2.1 that contained players that remained unseen by the model. The two most accurate models were the random forest and XGBoost models. Figure 4.11 indicates the loss estimates of these two models and the RMSE when no change in Estimated Transfer Value was predicted. The results show that both models provide interesting information as they have lower loss values than predicting no change. Moreover, the random forest and XGBoost models performed similarly.

The results also show that the RMSE values increased after the year 2018. This trend was present both for the losses of the models and the losses of predicting no change. The increase of the loss of predicting no change indicates that more changes in the ETV were present in the later years. This could be explained by differences in the feature distributions. It is known that the distribution of, for instance, contract length changes over time. This is due to the fact that more detailed information was available about contract lengths for later data points. This resulted in less estimated contract lengths, which have an important influence on the Estimated Transfer Value. As the contract length is dependent on time, it can also be assumed to be influential for the Estimated Transfer Value one year later in time. Although the distribution of this feature can be expected to become more stable, the results indicate that the RMSE values are likely to increase in the years after 2021.

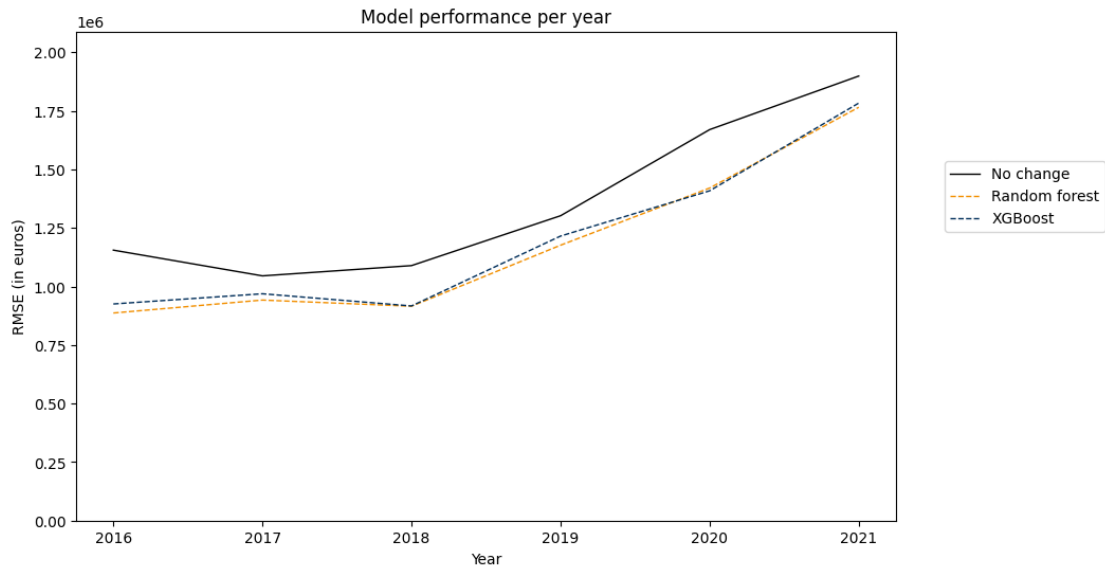


Figure 4.11: The RMSE estimates for each year on the stratified 5% test set in the Estimated Transfer Value case study.

#### 4.2.6 | Feature importances

The linear and tree-based models have methods to describe the importance of features for predictions of the model. These were rescaled and visualized in Figure D.28. In general, different features were considered important by the linear and tree-based models.

The results show that the ETV-related features ('etv', 'etv\_diff\_6m\_ago', 'etv\_diff\_6m\_ago') were most often the important features. This implies that information about historical values contains information to predict future values.

The results indicate that, for instance, the number of minutes played in the last 6 months ('minutes\_played\_last\_6m'), the number of minutes played multiplied by the resistance value of the domestic competition within the last 6 months, ('minutes\_x\_resistance\_domestic\_last\_6m'), and the number of minutes played in an international competition in the last 6 months ('minutes\_played\_international\_competition\_last\_6m'). The relatively high importance, mostly in the tree-based models, of these features indicated the importance of playing time in the development.

More surprisingly, the results show that the feature describing the month only has a large influence on the predictions for the tree-based models and not for the linear models. This value only contained two different values, which makes it impossible that this difference is caused by a nonlinear relationship with the dependent variable. This means that it must be caused by the interaction terms in the underlying relationship be-



tween the dependent variable and the variable month, as tree-based models are able to take this interaction into account whereas these were not included in the linear models.

A similar pattern is shown for the age-related variables, which are only assigned relatively high importance by the tree-based models. As these are continuous variables, this indicates that there are nonlinearities or interactions with other variables that are important.

The results also show that the linear mixed effect model and the random forest models assign relatively high importances to less important variables for the other models such as the number of minutes played by a player in the last six months ('minutes\_played\_last\_6m') and the sum of the variances in the SciSkill historical values ('sciskill\_variance\_6m'). For the random forest, this can be explained by the fact that these features are correlated with other features combined with the fact that a random forest has the ability to handle highly correlated features relatively well. For the linear mixed effect model, this can be explained by the fact that only the 20 most important features in the lasso model were selected, which is a relatively small feature set compared to the other models. The features in this set, therefore, have a relatively larger feature importance as there are fewer features.

## 4.3 | Analysis time series adjustment cross-validation

For the hyperparameter tuning, an adjusted time-series cross-validation estimation method was applied. To investigate the behavior of this method, it was analyzed by applying bootstrap 20 times to the loss estimations. This was also done for comparison with a time-dependent train/test split, a random train/test split, and a 5-fold cross-validation. Because the models have to be trained repeatedly, this was only done for the OLS and decision tree models to reduce computation time.

The results of the bootstrap analysis on the validation methods are shown in Figure 4.12 and Figure 4.13. The estimated standard deviations show that cross-validation and the adjusted time series cross-validation methods both result less variation within the estimates. This is due to the fact that they average over different folds and was the behavior that was aimed for when applying cross-validation.

The results show that the time-dependent validation methods, the time-dependent train/test split and adjusted time series cross-validation, have larger mean loss estimates than the classical validation methods, the random train/test split and 5-fold cross-validation. This difference indicates that data leakage occurred with the classi-

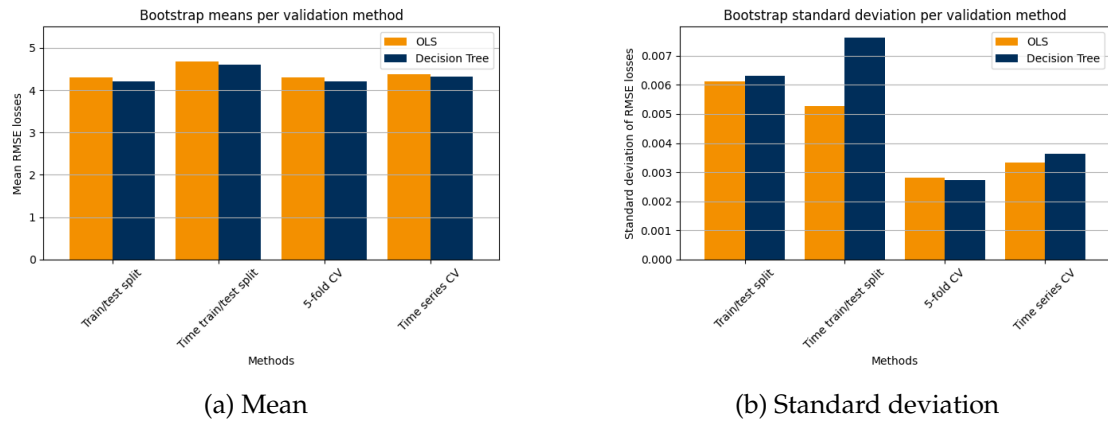


Figure 4.12: The mean and standard deviation estimates using 50 bootstrap samples of the SciSkill case study for the OLS and decision tree models.

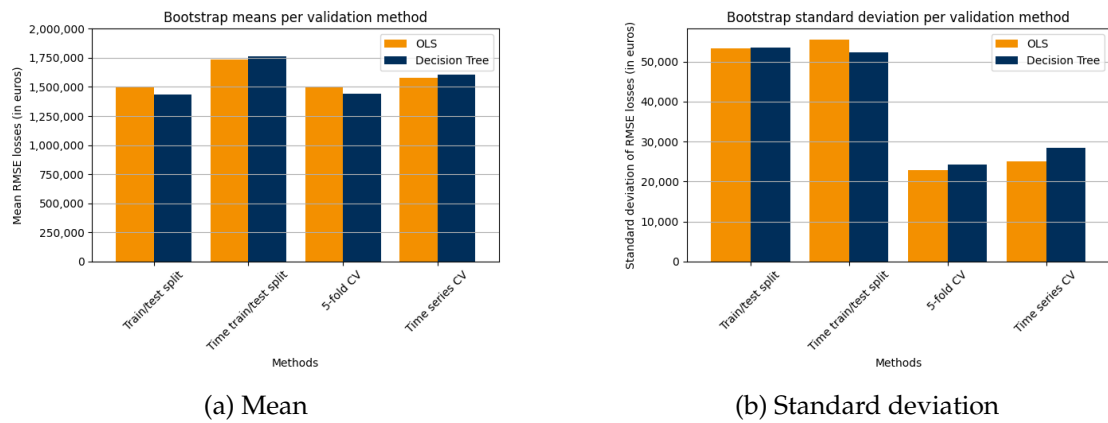


Figure 4.13: The mean and standard deviation estimates using 50 bootstrap samples of the Estimated Transfer Value case study for the OLS and decision tree models.

cal validation methods, which causes an overly positive loss estimate. This means that it was necessary to apply time-dependent validation methods.

The mean of the RMSE estimates of the time-dependent train/test split are larger than the adjusted time series cross-validation method. This indicates that the distribution of the points can differ for each year. It could, for instance, be the case that data points in later years are harder to estimate. Another possible explanation would be that the most relevant information is in the last year in the test set. The adjusted time series cross-validation had folds with only one year training data and the models would consequently not be influenced by the less relevant years. The adjusted time series cross-validation method, thus, had less variance in its estimates at the cost of slightly

too positive estimates of the loss functions.

The train/test split and the 5-fold cross-validation have similar mean RMSE estimates. This was expected as the samples they draw from have the exact same distribution. As a result, the 5-fold cross-validation method only decreases the variance of the loss estimates compared to the random train/test split for the predictive problems considered in these case studies and does not appear to change the mean of the estimates.

The time-dependent train/test split in the SciSkill case study is the only validation method, where there is a large difference between the standard deviations of the OLS and the decision tree models. Surprisingly, this difference is not present for the other validation methods. This might be caused by the appearance of some outliers in the training set that are very influential. As the decision tree model can split based on a few outliers, it can result in worse estimations on data points that are non-typical for the data set. The influence of these outliers is smaller on the OLS model, which is relatively simple and might, therefore, be better at extrapolating patterns. This might explain the large difference in the standard deviation of the OLS and decision tree models for the time-dependent train/test split in the SciSkill case study.

To summarize, the analysis showed that time-dependent validation methods are necessary to prevent data leakage. Although the adjusted time series cross-validation gives more positive values for models that perform well on small data sets compared to the time-dependent train/test split, its estimates have a smaller variance. It is, therefore, the favorable validation method is dependent on the application.



## Discussion

The aim of this research was to determine the best model to predict football player KPIs one year ahead. The quality of the model is considered as a combination of predictive performance, explainability, and methods to implement uncertainty quantification. The predictive performance of the models was studied in the case studies of this research. The explainability and methods for uncertainty quantification were covered in the literature study. Using these parts of the research, it will be concluded which model is the best with respect to these three criteria.

To work towards a conclusion, the predictive quality, explainability, and methods for uncertainty quantification will first be discussed. Afterward, the implications, strengths, and limitations will be covered for the different parts of the performed case studies. These are the data, the features, and the model training.

### 5.1 | Model quality

The main question of the current thesis is to determine the best model to predict future values of player performance KPIs with respect to predictive performance, explainability, and uncertainty quantification. The quality of the models will now be discussed for each of these aspects.

#### 5.1.1 | Predictive performance

##### 5.1.1.1 | SciSkill case study

As found in subsection 4.1.1, the best predictive quality for both the test estimates of the RMSE and the MAE was obtained by the XGBoost model closely followed by the

random forest model. The differences in the scores of the other models were almost all significant. Additionally, the RMSE scores of the XGBoost and random forest models performed best on the subset of young players, which was considered to be an important subgroup for the use case of this case study. Moreover, the XGBoost model had one of the best scores for the subset of outliers, whereas the random forest showed similar predictive performance. This means that the XGBoost model provided the best predictive performance for the SciSkill case study closely followed by the random forest model.

Next to that, the results show that the tree-based models had better loss values than the linear and kNN models, and the linear models achieved better predictive performance than the kNN models. The results show that the tree-based models performed best for players of a young age. Moreover, the losses on the subsets of outliers indicate that the tree-based methods generally provided better estimations for these groups. This means that the tree-based models generally performed well.

#### 5.1.1.2 | ETV case study

The results of the ETV case study provided less significant differences in the predictive quality of the models. The results indicate that the linear models predicted worse than the other models in terms of both the RMSE and the MAE. However, the differences in the RMSE estimates of all of the tree-based models and kNN-based models were not found to be significant. The MAE estimates did indicate the random forest model to be significantly better than almost all other models except for the XGBoost model.

The loss scores per age did not give a distinctly better performance of a model for this case study. The differences in performance on the subsets of important outliers were more obvious, as the random forest estimator clearly outperformed the other models on the subset of the players worth more than 10 million euros. Additionally, the random forest estimator also provided one of the best loss estimates on the other important subset. Therefore, the predictive performance of the random forest model was the best and the XGBoost model had the second-best predictive performance.

#### 5.1.1.3 | General prediction of player performances

Both case studies showed that tree-based models outperformed both linear and kNN-based models. They predicted better than linear models due to their ability to deal with nonlinearities and interaction terms, which were present as discussed in section 5.3. The tree-based models could infer these patterns with less data than kNN-based mod-

els, which explains the corresponding difference in predictive performance. In this way, the tree-based models seem to have better properties to accurately predict the future value of player performance KPIs.

From the tree-based models, the XGBoost had the best predictive performance in the SciSkill case study, closely followed by the random forest model. Conversely, the random forest model predicted most accurately in the Estimated Transfer Value case study, with the XGBoost model having the second-best predictive accuracy. The differences between the two models were larger in the case study of the Estimated Transfer Value. The results of the case studies, therefore, give rise to the conclusion that the random forest model has the best predictive accuracy over the two case studies.

Finally, the loss estimates can be used to describe the predictive problems in this study. First of all, the  $R^2$ -values in this research were below 0.4. This indicates that it was only possible to predict a limited part of the variance in the data. The results also show that the predictive performance decreased on the most interesting subsets of players, such as that of young players, good players, expensive players, and players with large changes in KPI values. Moreover, nonlinearity and interaction effects are important in this predictive problem. This implies that predicting the development of player performance KPIs in football over the next year is a challenging predictive problem.

The loss values of these models were determined in both case studies using a test set that contained the data points after 2020. These loss values estimated the generalization losses and were used to select the best model. These values describe the expected predictive performance of the models on unseen data. It should be noted that for both player performance KPIs, the changes in the player performance KPIs one year ahead appear to have become larger. This was shown by the increased RMSE values of predicting no change of the KPIs in subsection 4.1.6 and subsection 4.2.6. Consequently, it is important to monitor the predictive performance using the most recent data when models are used for the prediction of player performance KPIs.

### 5.1.2 | Explainability

One of the aims of this study is that the models should be explainable models, where the workings and predictions of a model can be explained. The linear models and kNN models are explainable because of the explainable workings of the model resulting in predictions. The tree-based models are also explainable because they are constructed using very explainable decision trees. Additionally, their predictions can be explained

using adjusted Shapley values by Lundberg and Lee (2017). All models in this research are, thus, sufficiently explainable.

Nevertheless, the workings of the kNN models are even more explainable. This is due to the fact that the general actions performed by the model are explainable in non-technical jargon. This makes the kNN models a type of model that can be of added value in providing explainable estimations in predictive problems.

One downside of the use of a general kNN model is the absence of feature importances. To this end, the research in Appendix B was carried out. This resulted in the implementation of the RReliefF algorithm, which estimates the feature importances based on the kNN model. The results show that the different input features describing the time series had different importances, with the most importance assigned to the age and most recent SciSkill value. Because the RReliefF weights did not improve the predictive quality compared to equal weights, the meaning of these weights is questionable. Consequently, the RReliefF weights do not provide the improved explainability that was aimed for.

Although all models are sufficiently explainable, there are some models that are more explainable than others. Linear models are highly explainable as the influence of the features can be described in layman's terms. The workings of a kNN are also explainable to practitioners because the algorithm provides historical data points of which the prediction is the weighted average. The random forest and XGBoost models, on the other hand, are less explainable compared to the other methods as a bit of the explainability of the decision trees is lost due to the bagging and boosting procedures. Consequently, the random forest and XGBoost models are slightly less favorable compared to the other implemented models in terms of explainability, although all models are sufficiently explainable.

### 5.1.3 | Uncertainty quantification

The last criterion on which models are assessed for their quality is their ability to provide uncertainty quantification. Uncertainty quantification can be solved by applying quantile regression like performed for XGBoost by Yin et al. (2023), which was based on changing the loss function so that it learned to estimate prediction intervals. A downside of this method is the fact that this type of uncertainty quantification is not explainable and needs an additional model.

This is different for uncertainty quantification methods that naturally arise from the inner workings of the predictive models. The kNN models, for example, make pre-



dictions based on  $k$  data points. These data points do not only provide the possibility to explain the predictions, but also the possibility to estimate the uncertainty of the prediction. This can be done by studying the variety between the values of the dependent values in these data points. A large variety means that the prediction is unsure, whereas a small variety within these values indicates that the model is certain about the estimation. In this way, the  $k$  neighbors can be used to obtain confidence intervals naturally arising from the model to provide uncertainty quantification.

Moreover, the OLS model has an underlying theory that gives prediction intervals, which arise from the uncertainty of the estimated coefficients and the randomness in new points. Although these prediction intervals are based on assumptions that did not hold, such as the normality of the errors, they do provide a measure of the uncertainty of the predictions.

Random forests are obtained using the bagging procedure, which provides a set of predictions from which the average is taken. This set of predictions can be utilized as described in Wager et al. (2013) to obtain uncertainty quantification. In this way, a natural uncertainty quantification method also emerges from the random forest models.

In these ways, uncertainty quantification methods can arise naturally from these models. For the OLS, random forest, and kNN models this means that they have good methods for uncertainty quantification. The XGBoost and the other models have the possibility for quantile regression, which needs an additional model that makes it possible to do uncertainty quantification. As a result, these models satisfy the requirement for uncertainty quantification, although the OLS, random forest, and kNN models are preferable in terms of uncertainty quantification.

#### 5.1.4 | Determination of best model

For the SciSkill case study, the findings show that the best predictive performance was obtained using the XGBoost model, closely followed by the random forest model. Both models achieved good loss estimates on the test set, on young players, and interesting outliers. Due to their tree-based structure, these models are capable of describing feature importance. Moreover, their predictions can be explained using SHAP values. This makes the random forest and XGBoost model explainable methods. The random forest model's bagging procedure naturally facilitates uncertainty quantification. On the other hand, the XGBoost model requires an additional XGBoost model with quantile regression to do this. This makes the random forest model more advantageous compared to the XGBoost model. Considering that the random forest model had nearly the

best predictive results, is explainable, and provides naturally arising uncertainty quantification, it can be concluded that the random forest model is the optimal model for the SciSkill case study.

In the case study of the Estimated Transfer Value, the predictive quality was the best for the random forest model. This was due to its good performance on the general test set and the important subgroups of players. Most notable was the good performance on the subset of players with an Estimated Transfer Value exceeding 10 million euros. The XGBoost model emerged as the second-best. Similarly as in the SciSkill case study, the random forest model is an explainable model and facilitates an uncertainty quantification method naturally arising from the bagging procedure. As a result, the random forest model is deemed the best model for the Estimated Transfer Value case study.

This research aims to find the best model with respect to the predictive quality, explainability, and methods for uncertainty quantification. In both case studies, the random forest estimator emerges as the best model with respect to these criteria. The results of this research, therefore, show that the random forest is the best model when taking into account predictive quality, explainability, and methods for uncertainty quantification.

## 5.2 | Used data

For the case studies of this research, two data sets were used to study. The data set for the SciSkill case study consisted of more than 3,800,000 monthly data points on more than 80,000 football players in the years 2014 up to 2022. Because this data set consisted of 86 features and the dependent variable, the challenging part of this study was to make sure that the different models could work with the size of the data set.

The data set of the SciSkill case study consisted of at least 20 data points per player and on average approximately 48 data points per player. The different data points corresponding to the same football player are similar, especially when close in time. This means that the samples in the data set are dependent, which creates an additional challenge for the models to infer patterns from the data.

On the other hand, the case study on the Estimated Transfer Value provides different challenges. This data set consisted of approximately 7 data points per player for more than 60,000 football players. The 413,000 data points covered the period from 2016 up to 2021 with 57 features. This time period is shorter because of limitations for the data availability at the time of the research. Nevertheless, the data set consists of 6 years of

data, which should still be enough to train predictive models for the development of the Estimated Transfer Value.

Another challenge is caused by the fact that the distribution of the ETV values is highly skewed and mainly has data points with a small Estimated Transfer Value. Additionally, the few players with large Estimated Transfer Values contribute to the largest errors. This means that the main challenge for the models in this case study is to infer patterns from a limited amount of a highly skewed data set.

The SciSkill case study comprised a data set with around 9 times as many data points as the Estimated Transfer Value case study. At the same time, the best model in the SciSkill case study attained an  $R^2$  of 0.378, despite the best model of the Estimated Transfer Value case study attaining 0.194. This indicates that a smaller proportion of the variance could be predicted for the Estimated Transfer Value than for the SciSkill. This might be due to the smaller data set available for the case study.

## 5.3 | Features

The features in the data sets were created to predict the development of football players. These features were constructed using domain knowledge of the football context and the processes of the models generating the player performance KPIs. Of course, it should be noted that it might be the case that more informative features could have been constructed for this study. However, the data availability provided a challenge for the construction of features such as player popularity or injury proneness.

As discussed in the literature study, the performance of a football player is known to be dependent on the age of the football player with a nonlinear relation. To deal with this, the age, the age squared, and the differences of the age and the average peak age on the given player position were included as features. The feature importances in the SciSkill case study show that the age of a football player is important for the development of the quality of a player as these quantities had a large importance in the models. The influence of the squared age in the linear, random forest, and XGBoost models in the SciSkill case study shows the presence of nonlinearity in the prediction problem. This means that the findings of this research are in accordance with the existing literature.

In the Estimated Transfer Value case study, information about the age of a footballer was also taken into account in other features in the data set, such as the Estimated Transfer Values itself or the difference between the SciSkill and the estimated potential

SciSkill. Nonetheless, the results attributed importance to the age-related features like the age or age squared. This means that the age-related features also influence the predictions of the development of the Estimated Transfer value one year ahead.

Next to that, it was found that features such as the current SciSkill ('sciskill'), and differences with the SciSkill and historical values ('sciskill\_diff\_1m\_ago', 'sciskill\_diff\_6m\_ago', 'sciskill\_diff\_12m\_ago') influenced the predictions of multiple models in the SciSkill case study. This implies that the time series information of the model provides useful information for the prediction of SciSkill values one year later. A similar occurrence was visible in the Estimated Transfer Value case study, where the current ETV ('etv') and differences with the ETV and historical values ('etv\_diff\_6m\_ago', 'etv\_diff\_12m\_ago') have high feature importances. This shows that the historical time series of the KPI values provides information about future values.

An interesting finding is that the development of the ETV in the next year was heavily dependent on the current month of the year. This is shown by the high feature importance of the month and could be expected because transfer decisions are generally different in the winter and in the summer. This was not reflected in the linear models and the feature only attained two values. This implies that this is due to the interaction of this variable with other variables, which was not taken into account by the linear models. Although the month is important for the ETV development, the results show that it is not considered important for the development in the SciSkill. This means that the time of the year could be an important factor in the prediction of future values dependent on the player's KPI, but it not necessarily is.

The study performed by Baouan et al. (2022) showed that the age and the number of playing minutes were important variables in predicting the market value of football players one year ahead. The results in the current thesis attribute large features importance to features describing the number of minutes played. The feature importances of the tree-based methods also indicated that age-related features were important for the prediction of the ETV values a year later. The findings in this research are, therefore, largely in line with the results of Baouan et al. (2022).

## 5.4 | Model training

### 5.4.1 | Hyperparameter optimization

Except for the OLS and LME models, all models in this research have been trained after applying hyperparameter tuning. This was done by applying a Bayesian optimization

algorithm to minimize the estimated loss values. The visualizations of the evaluations in Appendix B show that the hyperparameter tuning algorithm evaluated the boundary points of the hyperparameter space for every model. This can be explained by the fact that the Bayesian optimization algorithm selects points by taking the point that maximizes the chance of improvement and minimizes the lower confidence bound, which is often the most extreme point as the algorithm is least sure about its value. Consequently, the Bayesian algorithm favors the boundary points of the search domain. With enough function evaluations for the hyperparameter tuning, this effect should be reduced. As the Bayesian algorithm only found minimal gains for the last function evaluations, it can be assumed that enough function evaluations have been performed. Therefore, this bias is most likely not problematic.

The cross-validation method adjusted for time series was implemented as the objective function of the optimization problem. The results showed that the adjusted time series cross-validation decreased the variance of the loss estimates at the cost of taking into account loss estimates of models trained on smaller data sets when they are compared to the time-dependent train/test split. Consequently, hyperparameter values that better handle smaller data sets are favored. Therefore, the use of the time series adjusted cross-validation results in the hyperparameter tuning method selecting parameters corresponding to less flexible models.

As the first ten function calls of the Bayesian algorithm are chosen randomly in the data set, the improvements after the first ten function calls are improvements that can be attributed to the use of the Bayesian algorithm. The convergence plots in Appendix D show the improvements in the best-found value for all numbers of function calls. It can be seen that all models improve their function value the most in the first few function calls. The plots of all models with up to two hyperparameters show that almost no improvement was attained after the first ten function calls. It can, therefore, be concluded that the added value of the application of the Bayesian algorithm was minimal for these models.

In contrast, the random forest estimator and the XGBoost estimator have more hyperparameters, and finding the best values, therefore, is a harder problem. As an example, the hyperparameter tuning of the final random forest model in the Estimated Transfer Value case study shows that the best-found value improved after the first ten function calls. Similar behavior was more often displayed by the function values in the convergence plots of the hyperparameter tuning for the random forest and XGBoost models. This means that the Bayesian algorithm did manage to improve the quality of the chosen hyperparameters for these models.

In general, the Bayesian algorithm provides a way to minimize the losses in order to perform hyperparameter tuning that treats all models equally. Although the algorithm favors boundary points slightly, this does not appear to have much impact. Therefore, the application of the Bayesian algorithm to minimize the losses in the hyperparameter tuning was of added value to this study.

### 5.4.2 | Feature selection

The first part of the model training was performing feature selection. For each model, a suitable feature selection was applied that is closest to the model itself. Consequently, the linear models, for instance, cannot describe nonlinearities or interactions between the features. The linear models assign higher importance to features with a linear relation. Because the different models infer patterns differently, taking one universal feature selection method would introduce favoritism in this research toward certain models. The model-specific feature selection methods provide the possibility to compare the models without indirectly favoring a specific model.

The feature selection methods selected on average approximately 70 out of 86 features in the SciSkill case study. This shows that the models generally excluded only a few features. It could be an indicator of the fact that almost all features contain some sort of information about the development of the SciSkill in the next year.

On the other hand, the feature selection methods selected on average approximately 31 out of 56 features in the ETV case study. This means that a smaller proportion of the features was selected in the ETV case study compared to the SciSkill case study. A possible cause for this could be the fact that fewer features contain useful information to predict the ETV development in the next year. This could explain why the models in the ETV case study were able to predict a smaller proportion of the differences compared to the SciSkill case study, as reflected by the generally lower values of the  $R^2$  loss estimates.

To summarize, the feature selection methods were chosen to be model-specific. This avoids possible favoritism of certain models. Almost all features were selected in the SciSkill case study. This was probably due to the fact that most features have some importance. In the Estimated Transfer Value case study, around half of the features were selected, which illustrates that the feature selection methods are able to exclude unimportant features. The feature selection methods, therefore, performed as aimed for.

### 5.4.3 | Chosen models

#### 5.4.3.1 | Linear models

Linear, tree-based, and kNN-based models were studied in this research, of which the most basic type of model was that of linear models. The implemented linear models were OLS, lasso, and linear mixed effect models. The OLS model was implemented as a baseline. The OLS model and the lasso model performed similarly in both case studies. This could be due to the fact that overfitting is not an issue in the OLS model, because of the size of the data sets in this research. Nonetheless, the OLS model had large feature coefficients due to high correlations between the age-related variables. The lasso model shows smaller feature coefficients with similar performance. As a result, the coefficients of the lasso model are most likely closer to the true relationship, and regularization is of added value in the predictive problems of this study.

The third studied linear model is the linear mixed effect model. This model provides the possibility to include a random effect to correct for the random influences of this variable on the predictions. In theory, it is possible to include multiple random effects in the model. In practice, no suitable implementation for this was available in Python. An own implementation of this method would have taken up too much of the available time. The inclusion of only one random effect might be the cause of the limited performance of this model.

The random effect considered in this study is the nationality of a football player. This means that the influences of nationality were compensated by the model. If the mixed linear effect model had performed differently than the other linear models, this would have been an indicator that the other models discriminated indirectly based on nationality. The linear mixed effect models performed similarly to the other linear models, which means that it does not indicate discrimination based on nationality by the models.

#### 5.4.3.2 | Tree-based models

The tree-based models in this research are the decision tree, a random forest, and an XGBoost model. With these models, it is possible to study the added value of the bagging and boosting by the random forest and XGBoost models, respectively. Within the tree-based models, the random forest and XGBoost models had a better predictive performance than the decision tree. This shows that both bagging and boosting provided an increased predictive performance of the models.

Furthermore, it was observed that the tree-based models generally outperform the linear models with respect to predictive quality. Even the decision tree model, which was implemented as a baseline model, outperformed the linear models in terms of predictive accuracy. The feature importances imply the presence of nonlinearities and interaction effects between the features in the true underlying relation as discussed in section 5.3. Tree-based models can take these into account, whereas linear models cannot. This explains the improved qualities of the predictions by tree-based models.

A surprising finding is the fact that training loss of the XGBoost model clearly showed that this model was overfitting in the SciSkill case study as described in subsection C.1.6. Similarly, the random forest model was overfitted in the Estimated Transfer Value case study as discussed in subsection C.2.5. Still, these models had the best test loss values in said case studies. This means that the overfitting of these methods does not seem extremely harmful, although the hyperparameters could be chosen by hand to prevent overfitting. This might result in a more robust model.

The results show that the XGBoost and random forest models had an improved predictive performance compared to the linear models in both case studies. Still, the linear model had an  $R^2$  of at least 0.3 in the SciSkill case study. This means that the linear models were able to infer some of the patterns of the development of the SciSkill in the next year. It might have been interesting to create a stacked model where a tree-based model was applied to the residuals of the OLS model. In this way, the tree-based models could focus on specific patterns that a linear model could not describe, which might provide new insights. Features with a large importance for the tree-based model could, for instance, be considered to have a nonlinear influence or to have important interaction effects. Additionally, it might improve the predictive performance.

#### 5.4.3.3 | Time series kNN models

The kNN models are the last type of models that was studied in this research. These models were trained with time series information as features and, in this way, did not have access to the information contained in various features of the general data sets of the case studies. The results show that the predictive performance of these kNN models was not competitive with the tree-based models that were provided with more information, although they had better predictive performance than the linear models in the Estimated Transfer Value case study.

The results also show that the bad performance of the kNN models was mostly caused by the larger RMSE for the lower ages in the SciSkill case study. There were



fewer data points corresponding to the lower ages. Because the kNN models are local models, the small amount of data points made it more difficult to predict for the models as there were only a few data points to infer the right patterns. The kNN models, therefore, performed worse because of the low number of data points available for the younger ages.

At the same time, the loss functions of some kNN models show multiple local optima in the marginal objective function with respect to the number of neighbors  $k$ . This could, for example, be found for hyperparameter tuning of the final kNN model in the SciSkill case study. The appearance of multiple optima can be explained by the fact that several subsets of the data have a different optimal number of neighbors. These different optimal values were dependent on age as not many similar data points were available for younger players as discussed. Consequently, it might be better to introduce a dynamic number of neighbors or implement a hybrid model based on multiple kNN models with different numbers of neighbors.

For the kNN models, time series information was used to predict the dependent variable, which created extra dependencies within the features. The Mahalanobis distance transforms the features into a space where all features are independent. This explains why the use of the Mahalanobis distance for the kNN model resulted in improved predictive quality compared to the standard Euclidean distance of the normal kNN model in the SciSkill case study.

In order to measure and use feature importance, kNN models with RReliefF weights were implemented. The RReliefF weights, however, did not improve the loss values for the kNN model with RReliefF weights compared to the normal kNN model. This means that the inclusion of the feature importances did not improve predictive quality and that the aim of the RReliefF implementation is not achieved. However, the feature importances were multiplied in this model with the features to influence the scale, but it might be the case that another method provides better results.

#### 5.4.3.4 | Reflection on the methods

To summarize, the results indicate no large differences in performance between the linear models. The results show that the lasso model regularization was of added value. The LME models did not improve the predictive quality and did not indicate discrimination based on nationality. The tree-based methods showcased the possible improvements obtained by bagging and boosting. The kNN models showed that the time series features contained interesting information, although they did not perform best. In this

study, the tree-based models outperformed the linear and kNN-based models.

At the same time, the tree-based models are the most complex models. As they also had the best predictive performance, this might indicate that more complex models could obtain better predictions. Although more complex models often have a worse explainability, they might provide better predictions. Therefore, the insights obtained by this study in the trade-off between the different criteria are limited by the considered models.

#### 5.4.4 | Validation methods

Different validation methods were used to do the hyperparameter tuning and the assessment of the models. The bootstrap analysis of different validation methods in section 4.3 indicated that with normal cross-validation or a random train-test split, data leakage would have occurred resulting in overly positive results. This means that the implementation of time-dependent validation methods was necessary.

To estimate the generalization loss, which was used to assess model quality, the time-dependent train/test split was considered. The results of the analysis of the validation methods showed that this method had a larger variance of the predicted loss value. Nonetheless, conclusions about differences in performance could oftentimes be determined because bootstrap analysis was applied to study this variance. In this way, the larger variance of the predicted loss values did not prevent the assessment of the predictive quality of the models.

For the objective function of the hyperparameter tuning, it was not possible to do a bootstrap analysis. Therefore, the adjusted time series cross-validation method was applied. Although the adjusted time series cross-validation method resulted in more positive loss values than the time-dependent train/test split, this method reduced the variance, sometimes called noise, of the objective function of the hyperparameter tuning. The reduction of noise in the objective function was of added value, as it speeds up convergence and results in more accurate minima. Therefore, the adjusted time series cross-validation methods were better than a time-dependent train/test split for the purpose of hyperparameter tuning.

The root mean square error (RMSE) is the main loss function in this research. The RMSE assigned larger penalties to large errors. This might be harmful if there are many outliers of the data set that produce large errors and are not of specific interest. This was not the case because the case studies contained some interesting subgroups that produced higher loss functions such as the players with large SciSkill values, Estimated

Transfer Values, decreases, and increases. The large penalties of the RMSE for these outliers forced the models to predict with smaller errors on these subsets. The conscious choice for the use of the RMSE as the mainly considered loss function, therefore, resulted in models with a better description of the quality for the use case.

The  $R^2$  and mean absolute error (MAE) were also estimated for the models. The  $R^2$  is equivalent with the RMSE in study as proven in section A.1. Nonetheless, the  $R^2$  was of added value because it provided a different interpretation as it describes the fraction of variances in the data explained by the model.

Some studies in the literature study, such as Al-Asadi and Tasdemir (2022), attained  $R^2$  values above 0.9, although the  $R^2$  values in the current thesis do not exceed 0.2. This difference can be explained by the fact that the other studies predicted the KPI values themselves. The current thesis, on the other hand, predicted the difference in the current values and the values one year ahead. This is more informative as it is actually the quantity of interest, but results in smaller  $R^2$ . The way the  $R^2$  is used in this study, therefore, provides the most useful insights but should be carefully considered when comparing the results with possible new methods.

The results in the ETV case study show that the decision tree model performed better than the linear models with respect to the MAE, but similarly with respect to the RMSE. As explained in subsection 4.2.1, this is most likely due to the decision tree being able to predict well on the average data points in the data set. In this way, the MAE provides insights into the predictive quality with the small errors having a stronger weight. This means that the calculation of the MAE was of added value.

In short, the results show that the time-dependent validation methods are necessary to assess the predictive quality of models. The choice for the adjusted time series cross-validation for the hyperparameter tuning and the time-dependent train/test split for the generalization loss obtained better validation possibilities for their corresponding applications. The use of the RMSE as the main loss function resulted in better performance of the most interesting football players. Moreover, the  $R^2$  and MAE provided additional insights into the predictive quality of the models. This means that validation methods are a strong point of this study.



## Conclusion

This research aimed to fill the gap in the current knowledge about predicting the values a year later of model-based player performance KPIs. Two case studies were carried out in order to give answer to the research question: what is the best prediction model to predict KPI values of football players one year ahead considering the accuracy, possibilities for uncertainty quantification of predictions, and the explainability of the models?

For the SciSkill case study, the results indicated that the XGBoost model had the best predictive performance, closely followed by the random forest model. These models attained good loss estimates on the test set, the young players, and the interesting outliers. These models have a tree-based nature and can be used to describe feature importance. Moreover, predictions can be explained using SHAP-values. Therefore, these models were considered to be explainable methods. The bagging procedure of the random forest model gave a naturally arising method to uncertainty quantification, whereas the XGBoost model needed an additional model with quantile regression to do this. This made the random forest model more favorable than the XGBoost model. Given that the random forest model was explainable, provided a natural uncertainty quantification, and had nearly the best predictive results, it was concluded that the random forest model was the optimal model for the SciSkill case study.

In the Estimated Transfer Value case study, the random forest model demonstrated the lowest loss values on the general test set. Furthermore, it outperformed the other methods on the subset of interesting outliers, particularly the subset of players with an ETV of more than 10 million euros. Due to the tree-based nature of these models, the feature importances, and the prediction explanations via SHAP values, the random forest model was an explainable model. The uncertainty quantification method that arose from the bagging procedure that characterizes the random forest model was promising.

Consequently, the random forest was the best model in the ETV case study.

In both case studies, the tree-based models came forward as models with the best predictive accuracy due to their ability to infer nonlinear patterns and feature interactions. More specifically, the random forest model came forward as the best model in both case studies with respect to the study goals. This gave rise to the conclusion that the random forest model was the best model to predict the player performance KPIs a year ahead due to its good predictive accuracy, methods for uncertainty quantification based on bagging, and the explainability of the model.

## Applications and recommendations for future research

### 7.1 | Applications

The work in this thesis has resulted in predictive models for the SciSkill and ETV case study that describe the development of the KPIs of a player in the next year. As the current situation only provided the current values of the player's performance KPIs, describing the expected future development adds new information to the existing models. In this way, the models introduce methods to extract more information from the existing player performance, which provides essential information for transfer decisions.

More specifically, the SciSkill and ETV are models that describe the general player quality and the monetary value of a football player. With the current values of these models, a football club can determine whether the transfer of a player is good value for money at that moment. The predicted future values of the SciSkill can be used, for example, to determine whether a player is expected to improve in quality. This could be useful to assess whether a young player who is not currently at the desired level of the team, is expected to reach that level. Using this information, a club could decide to keep players because the club expects them to improve in quality in the future. Using the estimated future values of the ETV, it is possible to find players who are of good value for money at the moment but whose transfer values are expected to increase. In this case, the club should buy the player sooner rather than later. These examples show two of many possible applications of the models predicting future values of the SciSkill and ETV KPIs in the case of data-informed decision-making.

The information about the future values of both models can also be combined. For

example, the models could predict that both the SciSkill and the Estimated Transfer Value will increase in the current situation of a player on their scouting list. The SciSkill could increase rapidly, whereas the Estimated Transfer Value increases slightly in the player's current situation. If the club does not have the urgent need for a player in that position, they could choose to wait a year. In that case, they could buy the player that increased notably in quality at a slightly increased price. This gives an example of the added value of combining the information of both models describing the development in the SciSkill and Estimated Transfer Value.

The model in the SciSkill described the quality of a player one year in the future. As discussed, one of the features of the Estimated Transfer Value case study was the potential SciSkill of a football player. This was the expected highest SciSkill value in the future career of the player and was based on an iterative model. The models in the current thesis could be adapted such that they could be called iteratively for multiple years. The application of these models could possibly lead to an improved potential estimation.

Moreover, the models predicting future SciSkill and Estimated Transfer Value can be used to facilitate data-driven decision-making. Pantuso and Hvattum (2020) presented a method to optimize the squad of a football club. Their optimization problem was defined using the current quality and market value of a player together with the future quality and market value of a football player. The predictive models from this thesis that predict the KPI values one year ahead can replace their method, which was limited to describing historical data. By combining the newly obtained models and the optimization methods of Pantuso and Hvattum (2020), a new optimization method can be derived that can be applied to perform data-based decision-making.

As discussed in the literature study, the football transfer market can be seen as a labor market with a wide availability of data on the performance of employees. A similar approach as in this study can be used to predict future values of KPIs of other labor markets. Moreover, the fact that interactions and nonlinearities are present in the football transfer market means that it can be expected that these might also be present in general labor markets. In this way, the results of this study can be generalized and used as a foundation to study general labor markets.



## 7.2 | Recommendations for future research

This study was performed to obtain insight into predicting the player performance a year in the future. This was done based on two case studies on the values resulting from SciSkill and the Estimated Transfer Value models. As discussed in chapter 2, there exist various other model-based KPIs and ideally models should be trained to predict future values of all player KPIs. However, the VAEP models, for instance, are based on event data instead of match sheet data and can be expected to behave differently. Additionally, the study only considers values one year in the future. It might, for instance, be the case that seasonality has a stronger influence on predictions six months in the future. Therefore, further research on the prediction of the development of metrics describing football players is recommended.

Another possible direction of research would be to study the influence of contextual information. The models employed in the conducted research predicted the development of the player performance KPIs in the next year without taking into account all contextual information. We expect that the player's performance is dependent on, for instance, the team's playing style, the number of minutes given to the player, and the teammates. It would be beneficial for a manager to ascertain the anticipated impact of certain decisions, such as the allocation of playing time within a team, on the players' development. Consequently, we recommend a further study of the influence of contextual information on the players' development.

Moreover, the results showed the presence of nonlinearities and interactions in the development of player careers of footballers. As described by Franceschi et al. (2023), most previous studies on player evaluation methods have often used linear models. In recent years, more nonlinear models have been used to obtain predictive models on player performance and player valuation as shown in the literature study. The presence of nonlinearities and interactions that were found in this thesis suggests that this transition towards nonlinear models is beneficial for accurately describing a football player's career. Therefore, it would be advised to use nonlinear models in new studies on further research of the prediction of football players' development.

A possible extension of linear models are Generalized Additive Models (GAM) and Quantile Additive Models (QAM). The studies of Decroos and Davis (2020) and Yang et al. (2022) showed the potential of these models in the domain of football analytics. GAMs and QAMs might have provided interesting insights as they are able to describe nonlinear relations and interactions while still being based on the additive structure of the linear models. They might be of added value to the proposed studies.

Other methods to improve the predictive accuracy and obtain new insights might be obtained by considering the trained models in this study in a different way. The kNN results, for instance, showed multiple local optima for the number of neighbors. This could be dealt with by implementing a hybrid model based on multiple kNNs with different numbers of neighbors. Another possibility would be to train a random forest estimator on the residuals of a lasso model. This hybrid method would combine two explainable methods and could use the uncertainty quantification of the random forest. If a feature has a low feature importance in the lasso model and a large importance in the random forest model, this would mean that the feature has a nonlinear influence or interactions are present. In this way, hybrid models could provide new insights and possibly enhance predictive performance at no cost of explainability and uncertainty quantification. It would, therefore, be recommended to include hybrid methods in future research about the development of model-based player performance metrics.

The findings of this research indicated that the kNN models provide improvements in predictive quality compared to taking the mean or a constant increase. Additionally, the feature importances showed that time series related features influence the predicted values of the models. This means that there is useful information in the time series. It would be beneficial to investigate whether combining certain features with a time series model would enhance performance. Consequently, further research on the time series perspective of player performance KPIs is recommended, as this may yield new insights into patterns and potentially improve predictive performance.

A possible model to leverage the time series data would be recurrent neural networks (RNNs), which are less explainable than the GAMs, QAMs, and models considered in the current thesis. Nevertheless, RNNs such as long short-term memory (LSTM) or gated recurrent unit (GRU) might provide improvements in predictive quality because of their ability to learn from both features and time series data. The implementation and study of these models would provide further insights into the trade-off between predictive quality and explainability. Therefore, a further study is recommended on the trade-off between predictive accuracy and explainability of the models by investigating the predictive performance of recurrent neural networks.

## Mathematical details

### A.1 | Equivalence of RMSE and $R^2$

In this research, both the root mean squared error (RMSE) and the  $R^2$  are used to describe the predictive quality of the trained models. The RMSE can be considered as the average predictive error with large errors being punished harder. The  $R^2$  has the popular interpretation that it describes the fraction of the variance explained by the predictions of the model. It typically attains values between 0 and 1, with a value of 1 meaning that it perfectly predicts everything and 0 meaning that it predicts as well as taking the mean. If the model predicts worse than the mean of the sample, even negative values can be attained.

If the RMSE and  $R^2$  are applied to the same sample, they can be considered equivalent. A.1.1 states that the estimated loss functions are equivalent when applied to a sample of data. It should be noted that by replacing the averages with expectations, the same proposition and proof hold for the relation between the RMSE and  $R^2$ .

**Proposition A.1.1.** *Let  $(x_1, y_1), \dots, (x_N, y_N)$  denote a data set with  $x_i$  being the features and  $Y_i$  being the dependent variables of a predictive model with  $N$  data points. Assume that there exists  $y_i \neq y_j$  for some  $i, j$ . Let  $f$  be the predictive model for which the RMSE and  $R^2$  are determined. Let*

$$\widehat{RMSE} = \sqrt{\sum_{i=1}^N (y_i - f(x_i))^2},$$

and

$$\widehat{R^2} = 1 - \frac{\sum_{i=1}^N (y_i - f(x_i))^2}{\sum_{i=1}^N (y_i - \bar{y})^2},$$

denote the estimates of the RMSE and  $R^2$  where  $\bar{y}$  is the sample mean. There exists a strictly decreasing function describing  $\widehat{R^2}$  in terms of  $\widehat{RMSE}$ .

*Proof.* Define  $g : [0, \infty) \rightarrow \mathbb{R}$  as  $g(x) = 1 - x^2/C$  with  $C > 0$ . Then  $f'(x) = -2x/C$  which is strictly positive for  $x > 0$  and equals zero at the only boundary point  $x = 0$ . Consequently, we have that  $f$  is a strictly decreasing function.

Chose  $C := \sum_{i=1}^N (y_i - \bar{y})^2 \geq 0$ . The existence of  $y_i, y_j$  such that  $y_i \neq y_j$  implies that there exists a  $y_k$  such that  $y_k \neq \bar{y}$ . This means that the chosen  $C > 0$ . Note that  $\widehat{R^2} = 1 - \frac{\widehat{RMSE}^2}{\sum_{i=1}^N (y_i - \bar{y})^2} = 1 - \frac{\widehat{RMSE}^2}{C}$ . We find that it is a function in the form of  $g$ . Thus, there exists a strictly decreasing function of the required form.  $\square$

We can consider two loss functions equivalent if the order of quality corresponding to the loss functions is the same. Now that we have that there exists a strictly decreasing function between the RMSE and  $R^2$ , this means that this function keeps the order of the RMSE and the other way around. This means that the RMSE and  $R^2$  are equivalent loss functions.

## kNN feature weight optimization

The implementation of the kNN model provided promising results for the SciSkill use case despite the fact that kNN is a fairly basic algorithm. An important downside of the interpretability of a kNN is the absence of feature importances. In order to be able to fine-tune the kNN algorithm to the use case and to provide feature importances, adaptations were applied. These will be discussed in this appendix.

### B.1 | Methods

#### B.1.1 | Requirements weighting methods

The goal of this study was to fine-tune the weights for a regression prediction problem. Three requirements were defined to measure the quality of a method. The seven different implementations were tested against these requirements to find out which are suitable for application in the main research of this thesis.

The first goal is that the tuning of the feature weights also improves the predictive quality of the methods. Therefore, a lower value of the RMSE is preferred and an increase in performance compared to the standard kNN implementation. Without this requirement, the feature weights would have no meaning as weights that give a worse result than applying equal weights do not satisfactory reflect the importances of the features in the data.

The second requirement is the maximal training time of a model. The maximal training time is set at 20 minutes for the training set. This ensures the applicability of the model as it makes the model fast enough to later apply hyperparameter tuning.

The last goal is that the interpretation of feature importance can be derived from the weighting methods. The normal kNN does not have feature importances, but by rescaling the features with weights, the weights can be seen as feature importances. The methods that weigh the features in a way that can be interpreted as feature importances are preferred.

### B.1.2 | Data

To compare the quality of the weighting methods, they were applied to the data for the SciSkill case study. To prevent data leakage, the data set of the original test set containing the years 2014 up to 2020 was used.

This data set was divided into a training set containing the years 2014-2019 and a test set containing the data of the year 2020. This resulted in a training set of 2,348,565 data points and a test set of 442,639 data points. The distribution of data points throughout the years can be seen in Figure B.1.

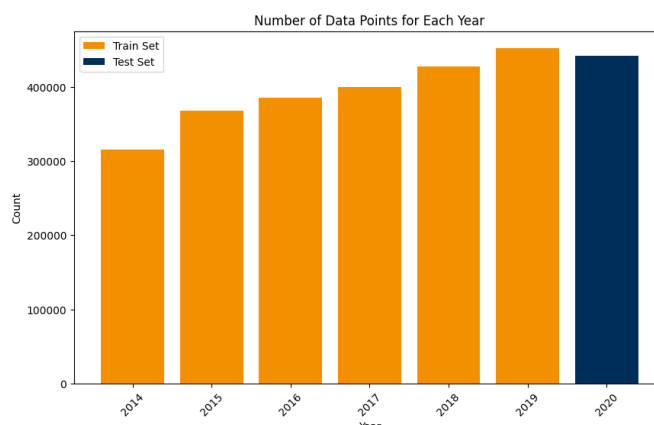


Figure B.1: The number of data points per year for the data set used to test the different kNN implementations.

For the purpose of studying the kNN with time-series data, a specific set of features was used. These features mainly consisted of time-series data containing the SciSkill values of the last 12 months of a football player. These features were complemented with the features 'age\_years', 'previous\_zero\_months', and 'month' as these features are cheap to compute and provide context to the time-series data. This resulted in a data set containing 15 different features and the dependent variable containing the difference between the future value of the SciSkill in 12 months and the current SciSkill. When the SciSkill is seen as a function with month and player as variables, this differ-

ence can be denoted as

$$\Delta\text{SciSkill}(\text{month}, \text{player}) = \text{SciSkill}(\text{month} + 12, \text{player}) - \text{SciSkill}(\text{month}, \text{player}).$$

The distribution of the dependent variable and two features are shown in Figure B.2. It can be seen that the SciSkill has a distribution that seems to be close to a slightly skewed normal distribution. The number of months since the last game of a player on the other hand is highly skewed as almost all data points have the value 0, which can be expected due to most players playing regularly. The distribution of the development Sciskill over a year seems to be strongly concentrated around the value 0 with a symmetric distribution and a points mass at 0.

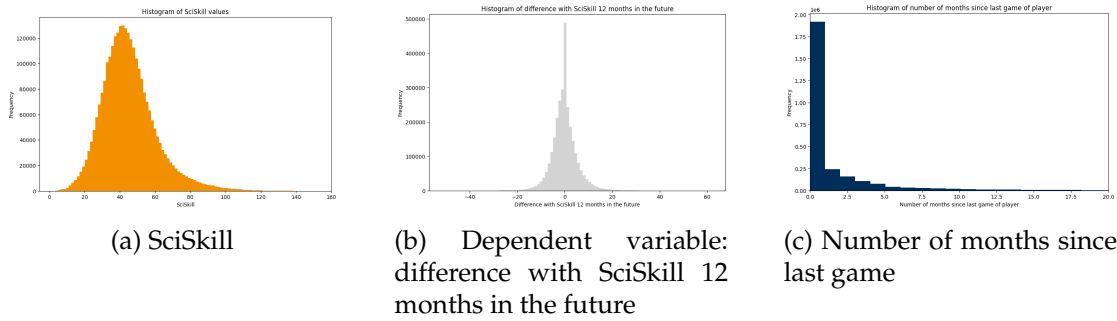


Figure B.2: The distributions of two features and the dependent variable used in the kNN study.

To summarize, the data set used consists of the data from the years 2014 up to 2020. The last 12 data points of the time series were used as features complemented with 3 contextual features. The resulting data set consisted of 2,791,204 data points in total.

### B.1.3 | Considered weighting methods

Several methods were considered in this study of feature importance weighting. All kNN models were implemented using the faiss package Douze et al. (2024) with hierarchical navigable small world indexing which builds a graph based on the similarity between data points in the data set. This provided a significant speed-up compared to more well-known scikit-learn implementation which follows Goldberger et al. (2004).

In order to save computation time, no hyperparameter tuning of the number of neighbors was performed for each method. A small line search of the number of neighbors for the standard kNN implementation resulted in an optimal value of around 40 neighbors. Short testing with the number of neighbors for the different algorithms did

not result in large changes in the predictive quality. All different feature weight optimization methods were therefore implemented with a 40 nearest neighbors algorithm.

In the first three optimization methods, the feature weights are determined by considering the selection of feature weights as an optimization problem. Define the predicted value of a kNN regressor with  $k$  neighbors based on the training data  $(Y_{\mathcal{D}}, X_{\mathcal{D}})$  as  $f_k(X|Y_{\mathcal{D}}, X_{\mathcal{D}})$ . When the weights of the features are denoted as  $\mathbf{w}$ , the optimization problem can be described as

$$\underset{\mathbf{w} \in \mathbb{R}^p}{\text{minimize}} \quad \mathbb{E}[(Y - f_{\text{kNN}}(X\mathbf{w}|Y_{\mathcal{D}}, X_{\mathcal{D}}\mathbf{w}))^2 | Y_{\mathcal{D}}, X_{\mathcal{D}}\mathbf{w}]. \quad (\text{B.1})$$

In this case,  $(Y, X)$  is an unseen pair of data points and the loss function was chosen to be the mean absolute error. The multiplication of  $\mathbf{w}$  with  $X$  and  $X_{\mathcal{D}}$  can be seen as multiplying the features with corresponding feature importance. The expectation in Equation B.1, the expectation was approximated by an average.

### B.1.3.1 | Bayesian optimization

The first implementation of feature weight optimization was obtained by applying Bayesian optimization to the problem described in Equation B.1. The Bayesian optimization was implemented using the implementation in Scikit-Optimize as described by Louppe and Kumar (2016).

The influences of the weights in the optimization problem Equation B.1 are relative to the other weights. For instance, two weight vectors result in the same results if they are linearly dependent. This means that the relative differences in order of magnitude are more important. To make the algorithm optimize these instead of the actual values, a softmax transformation was applied to the weights. The softmax function of a vector is defined elementwise as  $\sigma(\mathbf{w})_i := \frac{e^{w_i}}{\sum_{l=1}^p e^{w_l}}$ . In this way, the optimization algorithm was better able to determine the desired differences in the order of magnitude between the different feature weights.

The Bayesian optimization algorithm was implemented with the optimization space defined as the  $[-20, 20]^p$  hypercube of the input variables of the softmax transformation. The implementation was done using the scikit-optimize package. The acquisition function was chosen to be a function that iteratively alternates between minimizing the lower confidence bound, maximizing the expected improvement, and maximizing the probability of improvement.

The evaluation of the objective function consists of first training a kNN model and subsequently evaluating it on a test set. This resulted in a model that took considerably



longer than 20 minutes to train due to the costly objective function. To decrease the computation time, a subset of the training data was taken to compute the objective function on. This resulted in a trade-off between the quality of the estimate of the objective function and the computation time. A trial and error-investigation of this trade-off resulted in the best results. It was determined that taking a new 1% training set and a new 1% test set for each objective function call gave the best results in this trade-off between computation time and accuracy of the estimate of the objective function.

### B.1.3.2 | Genetic Algorithms

The second optimization method was optimizing the weights using a Genetic Algorithm. Similarly to the implementation of the Bayesian optimization, the weights were calculated after a softmax transformation. Next to that, the objective function was also obtained by taking a 1% subset for both the training and test set like was done for the Bayesian optimization. As Genetic Algorithms are robust against noisy objective functions, it was expected that the Genetic Algorithms could improve the quality of the performance.

The Genetic Algorithm was implemented using the toolbox provided in the package ‘deap’ by Fortin et al. (2012). The algorithm was applied with uniform initiation, tournament selection, a population of 20, and 50 generations. The best three individuals were kept in the new generations and the final best individual was taken as the solution.

### B.1.3.3 | Zeroth order optimization

The predictive function of the kNN algorithm is not a smooth function because it consists of the sum of multiple indicator functions, the optimization problem described in Equation B.1 does not have a gradient or Hessian available to use for optimization the problem. For this type of problem, zeroth order optimization methods were introduced which try to make local approximations of the objective functions without the gradient to obtain the next step to optimize the objective function (Liu et al., 2020).

To study the performance, an implementation using the method provided by Liu et al. (2022) was applied in the package ‘ZOOpt’. Again, a softmax transformation was applied to the feature weights and the objective function was calculated with 1% subsets to obtain the train and test data. In order to deal with the noisy objective function, re-sampling was applied which re-evaluated the best solution 10 extra times.

#### B.1.3.4 | RReliefF algorithm

Relief algorithms are methods to determine feature importance on a dataset based on the workings of a kNN algorithm (Robnik-Šikonja and Kononenko, 2003). The original Relief algorithm estimates the feature quality for classification problems by calculating the difference in feature value for the nearest neighbor having the same class and the nearest neighbor having another class label. In this way, the Relief algorithm measures how many differences exist for a feature between different class labels.

Kononenko (1994) introduced several extensions of the Relief method, including the ReliefF algorithm. Kononenko extended the Relief method by making it applicable to multiclass problems. He did this by considering the  $k$  nearest neighbors having the same class, and the  $k$  nearest neighbors for each different class. This method makes the ReliefF algorithm also more robust to noise (Robnik-Šikonja and Kononenko, 2003).

In order to make the algorithm applicable for regression problems, Robnik-Šikonja and Kononenko (1997) adapted the original algorithm to introduce the Regressional ReliefF (RReliefF) algorithm. This method determines the feature importance  $W[A]$  of feature  $A$  in kNN by defining

$$W[A] := P(\text{different value of } A | \text{nearest instances}) \\ - P(\text{different prediction} | \text{nearest instances}).$$

These probabilities in this expression were approximated using the pseudo-code in Figure B.3. In this pseudo code, the term *diff* is defined as

$$\text{diff}(A, I_1, I_2) := \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)}$$

where  $value(A, I)$  is the value of feature  $A$  for data point  $I$ . Next to this,  $d(i, j)$  should be a measure that indicates the similarity between data point  $i$  and data point  $j$ . This can be chosen as the number of data points that are closer to data point  $i$  than  $j$ . It can also be chosen as the reciprocal of the distance between the data points. This similarity measure should then be normalized such that it sums to 1 for each data point.

In this study, a kNN algorithm was first trained in order to calculate the neighbors for the RReliefF algorithm. The feature importances were then used to rescale the standardized features of the original kNN algorithm. By doing this, differences within a feature with high feature importance weigh more heavily in computing the final distances.

The implementation of the RReliefF method was done using the algorithm described in Figure B.3. In order to speed up the calculation time, a vectorized version of the al-

*Algorithm RReliefF*

*Input:* for each training instance a vector of attribute values  $\mathbf{x}$  and predicted value  $\tau(\mathbf{x})$

*Output:* vector  $W$  of estimations of the qualities of attributes

```

1. set all  $N_{dC}, N_{dA}[A], N_{dC\&dA}[A], W[A]$  to 0;
2. for  $i := 1$  to  $m$  do begin
3.   randomly select instance  $R_i$ ;
4.   select  $k$  instances  $I_j$  nearest to  $R_i$ ;
5.   for  $j := 1$  to  $k$  do begin
6.      $N_{dC} := N_{dC} + \text{diff}(\tau(\cdot), R_i, I_j) \cdot d(i, j)$ ;
7.     for  $A := 1$  to  $a$  do begin
8.        $N_{dA}[A] := N_{dA}[A] + \text{diff}(A, R_i, I_j) \cdot d(i, j)$ ;
9.        $N_{dC\&dA}[A] := N_{dC\&dA}[A] + \text{diff}(\tau(\cdot), R_i, I_j) \cdot$ 
10.         $\text{diff}(A, R_i, I_j) \cdot d(i, j)$ ;
11.     end;
12.   end;
13. end;
14. for  $A := 1$  to  $a$  do
15.    $W[A] := N_{dC\&dA}[A] / N_{dC} - (N_{dA}[A] - N_{dC\&dA}[A]) / (m - N_{dC})$ ;

```

Figure B.3: Pseudo code of the RReliefF algorithm (Adapted from (Robnik-Šikonja and Kononenko, 2003))

gorithm was implemented from scratch. Next to that, using trial and error  $d(i, j) := \frac{\|I_i - I_j\|_2}{\sum_l \|I_l - I_i\|_2}$  was chosen as  $d(i, j)$  where  $\|I_i - I_l\|_2$  is the Euclidean distance between two data points  $I_1$  and  $I_2$ .

### B.1.3.5 | Predictive performance of individual feature

Feature importances can also be quantified by determining the quality of the predictive performance using each feature individually. The estimated decrease in the loss function could then be seen as the feature importance.

This study introduced the method where an individual kNN algorithm was trained individually for each feature in order to estimate the predictive quality of each feature. This was done by splitting the training data into a training (80%) and test (20%) set. The model with one feature was fitted on the training set and the quality was determined using a loss estimate on the test set. The feature importance could then be defined as

$$W[A] := \max \left\{ 0, \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - f_{kNN,A}(X_i))^2} \right\}, \quad (\text{B.2})$$

where  $f_{kNN,A}(X_i)^2$  is the kNN estimation based of the model with the feature  $A$ . The idea behind this feature importance metric is that features that have a large predictive value are seen as more important. A downside of this method is that possible interactions between the features are not taken into account.

### B.1.3.6 | Mahalanobis distance

A different method of rescaling the features is by applying an affine transformation of the features. This can be done by multiplying the features with a matrix instead of a vector as done for the other methods and additionally adding a matrix. The new matrix of features  $X'$  can then be obtained by  $X' := TX + U$ , where  $X$  is the original feature matrix  $X$  and both  $T$  and  $U$  are chosen matrices. When  $X$  has  $p$  features and  $X'$  has  $p'$  features, this results in an optimization problem that can be described as

$$\underset{T \in \mathbb{R}^{p' \times p}, U \in \mathbb{R}^{p'}}{\text{minimize}} \quad \mathbb{E}[(Y - f_{\text{kNN}}(TX + U|Y_{\mathcal{D}}, TX_{\mathcal{D}} + U))^2]. \quad (\text{B.3})$$

The resulting optimization problem described in Equation B.3 is complex due to the high number of decision variables in  $T$  and  $U$  compared to the optimization problem described in Equation B.1. In order to make an educated guess, the Mahalanobis distance was used as described by McLachlan (1999). The Mahalanobis distance is a special choice of  $T$  and  $U$  in Equation B.3 such that a generalization of standardization is applied.

Let  $\Sigma$  denote the covariance matrix of the random variables which are the features and let  $\mu$  denote the average value of these random variables as described by McLachlan (1999). The Mahalanobis distance is then obtained by applying the affine transformation  $x' = (x - \mu)\Sigma^{-1/2}$ , where  $x$  is a random vector having the distribution of the features. Note that  $\Sigma$  is positive semi-definite by definition and can be assumed to be positive definite by dropping random variables with a single value. The expression  $\Sigma^{-1/2}$  is therefore valid.

The idea behind this method is that it not only standardizes the variances of the features but also takes the covariances into account when doing so. This is reflected by the fact that the covariance matrix of the transformed features is an identity matrix. Therefore, a more general standardization is applied. The final distances within the features are then calculated using the expression

$$||x_i - x_j|| = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}.$$

In this study, the covariance matrix  $\Sigma$  was estimated using the element-wise estimation of the covariance matrix,  $\text{cov}(X_i, X_j) = \frac{1}{n-1} \sum_{l=1}^n (X_{l,i} - \bar{X}_i)(X_{l,j} - \bar{X}_j)$ . The estimation of  $\Sigma^{-1/2}$  was then performed by calculating the Choleski decomposition of the  $\Sigma$  and subsequently calculating the inverse of this decomposition matrix. The values of  $\mu$  were estimated by calculating the sample mean value of each feature. Using both the estimated  $\hat{\Sigma}^{-1/2}$  and  $\hat{\mu}$ , the data was transformed using  $x' = (x - \hat{\mu})\hat{\Sigma}^{-1/2}$ . The transformed data was then used to train the kNN algorithm.

### B.1.3.7 | PCA distance

Another possible way to preprocess the input features is to apply principal component analysis (PCA). The PCA algorithm identifies new, simplified features by finding the best linear combinations of the original features that capture the greatest variance. This process helps to reduce the number of dimensions in the dataset while retaining most of the essential information that distinguishes the data points.

The PCA algorithm was implemented using the PCA implementation in the scikit-learn package which follows the implementations of (Halko et al., 2011; Martinsson et al., 2011). This resulted in principle components with the explained variance ratio as shown in Figure B.4. It shows that most of the information on the variances is contained within the first 7 principle components. It was therefore chosen to do the PCA implementation with the first 7 principle components.

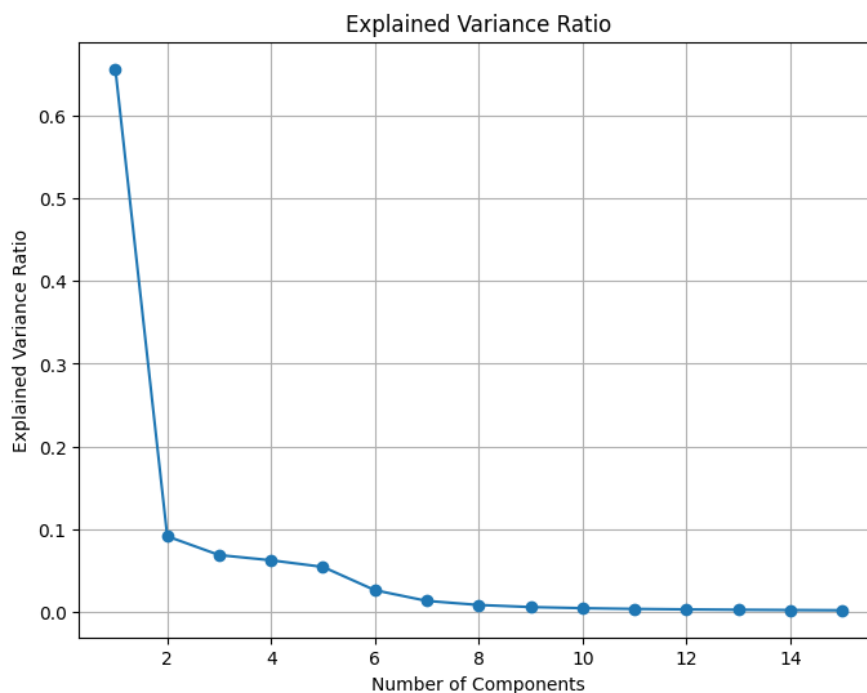


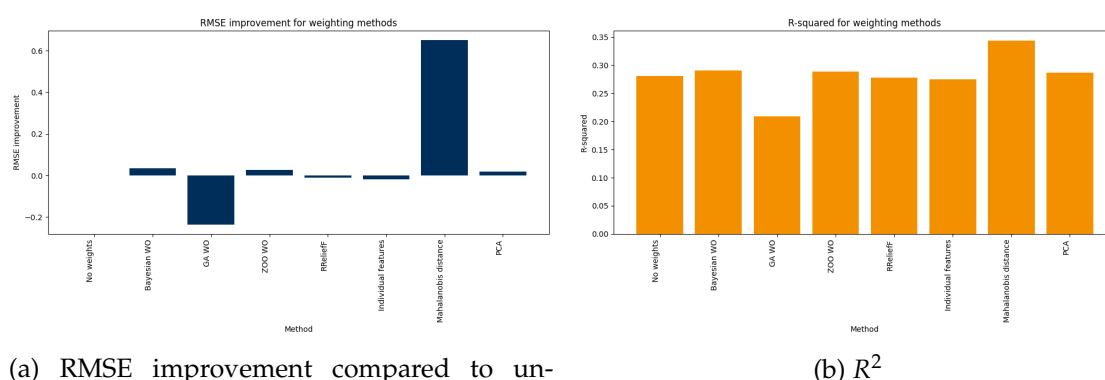
Figure B.4: A plot of the explained variance ratio for PCA.

## B.2 | Results

### B.2.1 | Predictive quality

The decrease in RMSE and the  $R^2$  of the different weighting methods are shown Figure B.5. It can be seen that the GA weight optimization method perform clearly worse than an unweighted kNN implementation. This means that these weighting methods did not improve the predictive performance as desired.

The Bayesian weight optimization, zeroth order weight optimization, RReliefF weights, the weights based on individual predictive performance, and PCA transformation all had similar performance to the kNN model with equal weights. These models, therefore, do not provide a clear improvement in predictive quality.



(a) RMSE improvement compared to un-weighted kNN

(b)  $R^2$

Figure B.5: Two performance metrics of the predictive quality of the models compared to an unweighted implementation. The unweighted implementation has  $RMSE = 4.76$  and  $R^2 = 0.31$ .

On the other hand, Mahalanobis distance method improved the quality of the predictions compared to the normal kNN implementation. This is probably because of the large correlations between the different time series features, which can be handled well by the Mahalanobis distance as transforms the features to an uncorrelated state. This means that it is a very suitable candidate considering that the requirement that the weighting methods should improve the predictive quality.

### B.2.2 | Running time

The running times of the applied methods are shown in Figure B.6. It shows that all methods have a running time within the desired 20 minutes. More specifically, the methods that determine the weights based on an optimization algorithm have the longest computing time. This is because of the high computational costs for the esti-

mated objective function that is used. Next to that, it can be seen that the Mahalanobis distance and PCA methods increase the computation time the least.

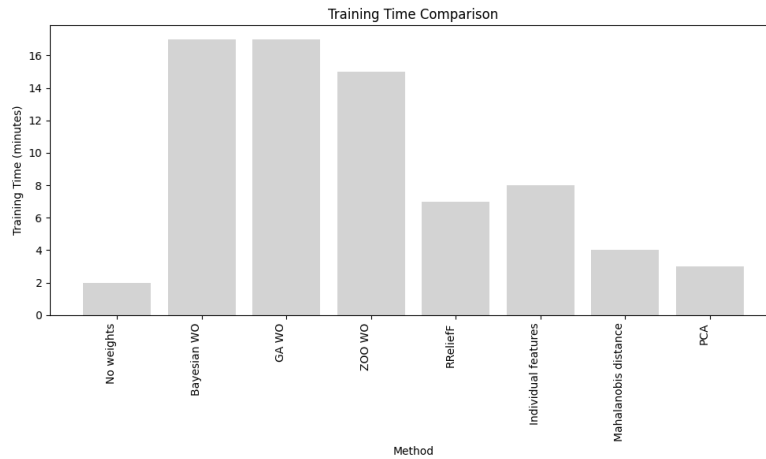


Figure B.6: The running times in minutes of the feature weighting methods.

### B.2.3 | Interpretation

Lastly, it was desired that the weighting methods should have some interpretation of feature importance. All methods had some sort of feature importance measure as will be described below. The resulting feature importances are shown in Table B.1.

The Bayesian weight optimization, Genetic Algorithm weight optimization, and zeroth order weight optimization methods are based on the optimization problem as described in Equation B.1. The optimally found values for  $\mathbf{w}$  in Equation B.1 can then be considered as the feature importances because a large weight value for a certain feature means that it has a large importance in predicting the dependent variable. Because the Bayesian optimization method and the zeroth order optimization method both improve the predictive quality, their weights can therefore in some way be interpreted as feature importance.

The individual feature predictive performance method gives a larger weight to a feature that has a better predictive quality for the dependent variable. It therefore has a clear interpretation of their feature importance. However, the predictive quality is not better than assigning equal feature weights in this prediction problem. This can be caused by the fact that this method ignores the possible dependence and interactions between the features. The possible interpretation does therefore not have much interpretive value similar to the genetic algorithm.

The RReliefF method is a method that was originally meant for feature importance based on a kNN algorithm. The resulting weights  $W[A]$  from the algorithm can therefore be interpreted as the feature importances as it will assign large values to more important values that have the largest influence on making good predictions for a kNN algorithm.

The Mahalanobis distance implementation is based on the optimization problem described in (B.3). As the influences of the different features are not weighted per feature, the interpretation is not as clear as for the methods based on (B.1). Still, the transformed features are linear combinations of the original features. When the coefficients that correspond to the original feature are added, this represents a single weight for the original feature. Let  $T = (\hat{\Sigma}^{-1/2})_{i,j}$ . The weight of feature  $j$  can then be formulated as

$$\sum_{i=1}^{p'} T_{i,j}.$$

This provides a method to calculate an interpretation of the feature importance based on the Mahalanobis distance.

This interpretation however does not take into account correlations between the variables. In this specific case, it is known that the SciSkill-related features are strongly correlated. As some values in the transformation matrix  $T$  are negative, this means that the combination of linear variables might cancel each other partly. This above-described method is, therefore, not useful for this application.

The PCA method also performs a linear transformation from the original space to a new feature space. The feature importance of this method has the same problem as the kNN model with Mahalanobis distance. On top of that, the PCA method does not provide a better predictive performance than an unweighted kNN. The PCA method is therefore considered a method without feature importance interpretation.

## B.3 | Conclusion and discussion

It can be concluded that all methods satisfy the computing time requirement and almost all satisfy the requirement for a meaningful interpretation of the feature importance. The Mahalanobis distance is the only technique that clearly improves the predictive quality of the kNN model as it can deal with highly correlated features. With this, the kNN implementation with Mahalanobis distance provides the largest improvement in the predictive quality and has a short computation time. This method, therefore, is the most promising weighting method for this use-case.



Feature	Bayesian WO	GA WO	ZO WO	RReliefF	Indiv. perf.
Age (years)	$1.67 \times 10^{-1}$	$7.32 \times 10^{-1}$	$1.34 \times 10^{-1}$	$4.15 \times 10^{-4}$	$3.99 \times 10^{-1}$
#Months inactive	$1.67 \times 10^{-1}$	$3.76 \times 10^{-5}$	$1.34 \times 10^{-1}$	$1.46 \times 10^{-4}$	$1.19 \times 10^{-1}$
Month	$7.07 \times 10^{-19}$	$1.23 \times 10^{-8}$	$9.22 \times 10^{-4}$	$7.90 \times 10^{-4}$	0.00
SciSkill (current)	$1.67 \times 10^{-1}$	$1.08 \times 10^{-1}$	$1.34 \times 10^{-1}$	$6.99 \times 10^{-4}$	$3.52 \times 10^{-2}$
SciSkill ( $-\Delta_t$ )	$3.47 \times 10^{-9}$	$1.01 \times 10^{-3}$	$2.33 \times 10^{-6}$	$4.57 \times 10^{-4}$	$5.70 \times 10^{-2}$
SciSkill ( $-2\Delta_t$ )	$7.07 \times 10^{-19}$	$2.18 \times 10^{-3}$	$1.34 \times 10^{-1}$	$4.20 \times 10^{-4}$	$7.40 \times 10^{-2}$
SciSkill ( $-3\Delta_t$ )	$1.67 \times 10^{-1}$	$1.77 \times 10^{-4}$	$2.30 \times 10^{-7}$	$4.13 \times 10^{-4}$	$8.67 \times 10^{-2}$
SciSkill ( $-4\Delta_t$ )	$7.07 \times 10^{-19}$	$4.67 \times 10^{-2}$	$1.34 \times 10^{-1}$	$4.13 \times 10^{-4}$	$9.56 \times 10^{-2}$
SciSkill ( $-5\Delta_t$ )	$1.79 \times 10^{-11}$	$8.48 \times 10^{-7}$	$5.95 \times 10^{-2}$	$3.99 \times 10^{-4}$	$1.04 \times 10^{-1}$
SciSkill ( $-6\Delta_t$ )	$1.67 \times 10^{-1}$	$5.58 \times 10^{-6}$	$1.63 \times 10^{-9}$	$3.76 \times 10^{-4}$	$1.12 \times 10^{-1}$
SciSkill ( $-7\Delta_t$ )	$2.14 \times 10^{-15}$	$1.13 \times 10^{-2}$	$1.34 \times 10^{-1}$	$3.54 \times 10^{-4}$	$1.12 \times 10^{-1}$
SciSkill ( $-8\Delta_t$ )	$8.21 \times 10^{-4}$	$9.74 \times 10^{-4}$	$1.88 \times 10^{-4}$	$3.22 \times 10^{-4}$	$1.23 \times 10^{-1}$
SciSkill ( $-9\Delta_t$ )	$1.22 \times 10^{-13}$	$8.55 \times 10^{-8}$	$1.60 \times 10^{-4}$	$3.02 \times 10^{-4}$	$1.22 \times 10^{-1}$
SciSkill ( $-10\Delta_t$ )	$1.67 \times 10^{-1}$	$5.75 \times 10^{-3}$	$2.34 \times 10^{-5}$	$3.16 \times 10^{-4}$	$1.38 \times 10^{-1}$
SciSkill ( $-11\Delta_t$ )	$7.07 \times 10^{-19}$	$9.22 \times 10^{-2}$	$1.34 \times 10^{-1}$	$3.63 \times 10^{-4}$	$1.49 \times 10^{-1}$

Table B.1: The feature weights of the different weighting methods. ‘WO’ corresponds to ‘weight optimization’, ‘GA’ to ‘Genetic Algorithms’, ‘ZO’ to ‘zeroth order’, and ‘Indiv. perf.’ to ‘Individual feature predictive performance’.

The other methods did not provide an improvement compared to the normal kNN model. This is probably due to the difficult nature of the predictive problem, possibly caused by the highly correlated features and the fact the largest errors are caused by the outliers of the data set. It could also be the case that weighting every feature equally gives good weights for this kNN weighting problem.

The most promising model of the features with similar performance is the RReliefF method. This is due to the fact that it is well-known for its feature importance and that it has a short computation time. Although it most probably did not show improvements on the data of the SciSkill case study, it might perform relatively well for the Estimated Transfer Value case study in this thesis.

Lastly, it can be seen that the feature weights in Table B.1 differ strongly in their values. Surprising is the fact that the Bayesian algorithm weight optimization only gives a very small or very large weight. This is probably caused by the fact that these values are the given boundary values which are more thoroughly discovered by the Bayesian weight optimization algorithm compared to the genetic algorithms and the zeroth order optimization. By doing this, the Bayesian algorithm practically applies feature selection.



## Model specific results

This appendix presents the results of the training of the individual models. Visualizations that take in much space are placed in Appendix D to improve readability.

### C.1 | SciSkill case study

For all implemented models, the loss values were estimated on the test and train sets in the SciSkill case study. These results are given in Table C.1. The results generally show that the XGBoost shows the most overfitting but also attains the best loss scores as discussed in subsection 4.1.1. In the following section, the results of the individual models will be discussed.

Model	Train set			Test set		
	RMSE	MAE	R <sup>2</sup>	RMSE (Std. dev.)	MAE (Std. dev.)	R <sup>2</sup> (Std. dev.)
OLS	4.3096	3.2040	0.3364	4.5996 (0.0053)	3.4086 (0.0034)	0.3223 (0.0056)
Lasso	4.3096	3.2038	0.3364	4.5998 (0.0054)	3.4088 (0.0033)	0.3222 (0.0011)
LME	4.3586	3.2273	0.3212	4.6595 (0.0057)	3.4362 (0.0035)	0.3045 (0.0011)
Decision tree	4.1797	3.0958	0.3758	4.5414 (0.0054)	3.3577 (0.0033)	0.3393 (0.0012)
Random forest	3.7500	2.8264	0.4975	4.4599 (0.0053)	3.3001 (0.0033)	0.3628 (0.0011)
XGBoost	1.0817	0.7606	0.9582	4.4078 (0.0051)	3.2707 (0.0033)	0.3776 (0.0012)
kNN	4.2844	3.1500	0.3441	4.7307 (0.0059)	3.4655 (0.0036)	0.2831 (0.0011)
kNN Mahalanobis	4.2930	3.1579	0.3415	4.7006 (0.0058)	3.4443 (0.0034)	0.2922 (0.0011)
kNN RReliefF	4.2688	3.1337	0.3489	4.7684 (0.0061)	3.4907 (0.0036)	0.2716 (0.0011)

Table C.1: The estimates on the train and test set for the different loss functions in the SciSkill case study. The standard deviations of the estimates on the test set were calculated using 1000 bootstrap samples.

### C.1.1 | Ordinary least squares

Backwards selection algorithm resulted in the 70 selected features out of 86. This feature subset was then used to train a final model.

Figure C.1 shows the feature importances of the features in the final OLS model. It can be seen that the age-related features have the strongest feature importance. These are the age of the player ('age\_years'), the age of the player squared ('age\_years\_squared'), and the difference between the age of the player and the peak age of his position in years ('years\_diff\_peak\_age'). Other variables that seem to be of medium importance are for instance the current SciSkill, the number of minutes played in a domestic league multiplied by the resistance factor of the league both in the last month and the last 6 months, the position line of the player, and the difference between the player's SciSkill and the average SciSkill of the team.

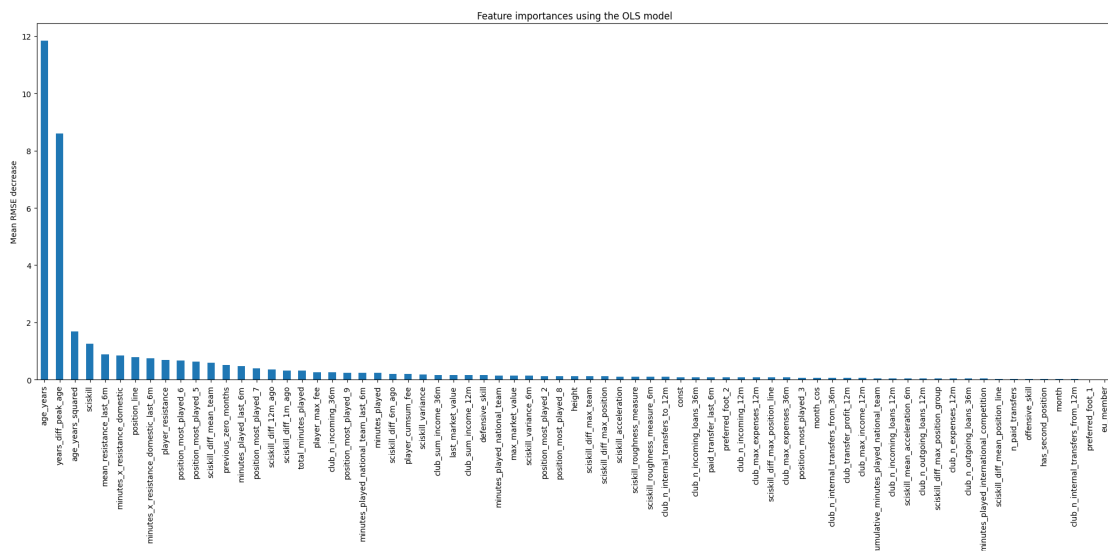


Figure C.1: The feature importance in the RMSE decrease for each feature for the final OLS model in the SciSkill case study.

Figure C.2 provides a visualization of the coefficients in the final OLS model. These coefficients were standardized before the model training. It shows that a higher SciSkill ('sciskill'), age ('age\_years'), position line ('position\_line'), or difference with the mean SciSkill of the team ('sciskill\_diff\_mean\_team') indicates that the SciSkill one year later is lower. On the other hand, a higher value for the age in years squared ('age\_squared'), the number of minutes times the resistance factor of the domestic competition ('minutes\_x\_resistance\_domestic'), or the difference in SciSkill with 12 months ago

(‘sciskill\_diff\_12m\_ago’) results in a higher predicted SciSkill value for one year later.

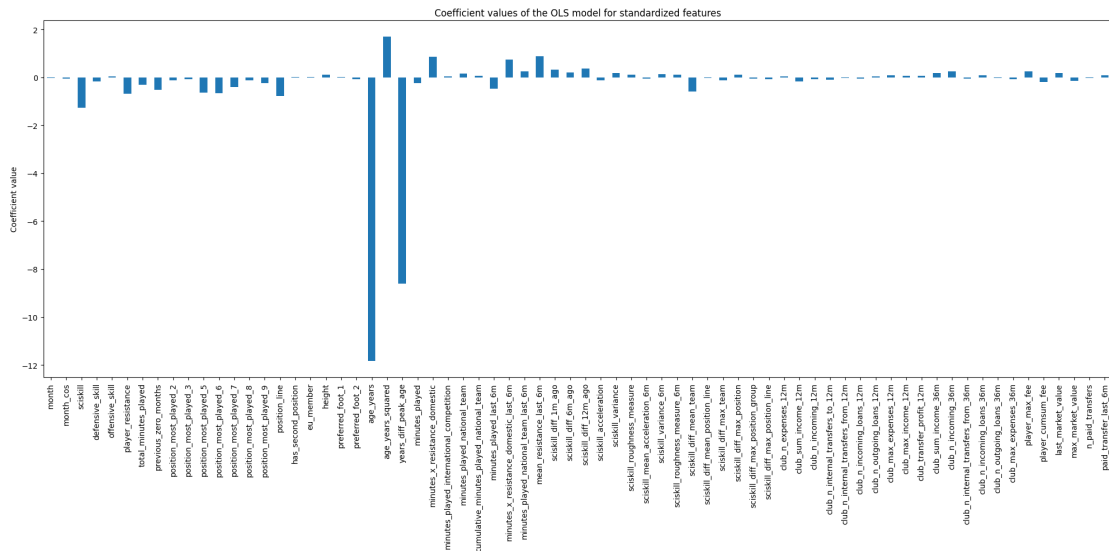


Figure C.2: The coefficients of the final OLS model of the standardized features in the SciSkill case study.

Because of the standardization of the features, these coefficient values cannot directly be used to describe the influence of the features on the prediction in the original scale. In order to do this, the coefficients can be rescaled to correspond to the original features before standardizing. These values are shown in Figure C.3.

For example, it shows a coefficient of around -2.9 for the age of a player (‘age\_years’). This means that for each year increase in the player’s age, the model will predict a value of 2.9 lower. It can also be seen that there seem to be mostly large negative influences of the features. The different scales of the features should, however, be taken into account. The indicator of the player being an attacking midfielder (‘position\_most\_played\_6’) has one of the largest coefficients and the SciSkill of a player (‘sciskill’) has a smaller coefficient. Because the first only takes values in  $\{0, 1\}$  and the latter takes values in  $(0, \infty)$ , the scales of these variables should also be taken into account. If this is done, the influence of the SciSkill is larger than the influence of the player being an attacking midfielder as indicated by Figure C.1 and Figure C.2. The results in Figure C.2, therefore, only provide insight into the influences of the features on the predictions in their original scale, but cannot be used to infer the feature importances.

This final model had the estimated loss values as shown in Table C.1. It shows that the loss values are slightly higher for the test set than for the training set. However, overfitting is probably not a problem due to the small differences.

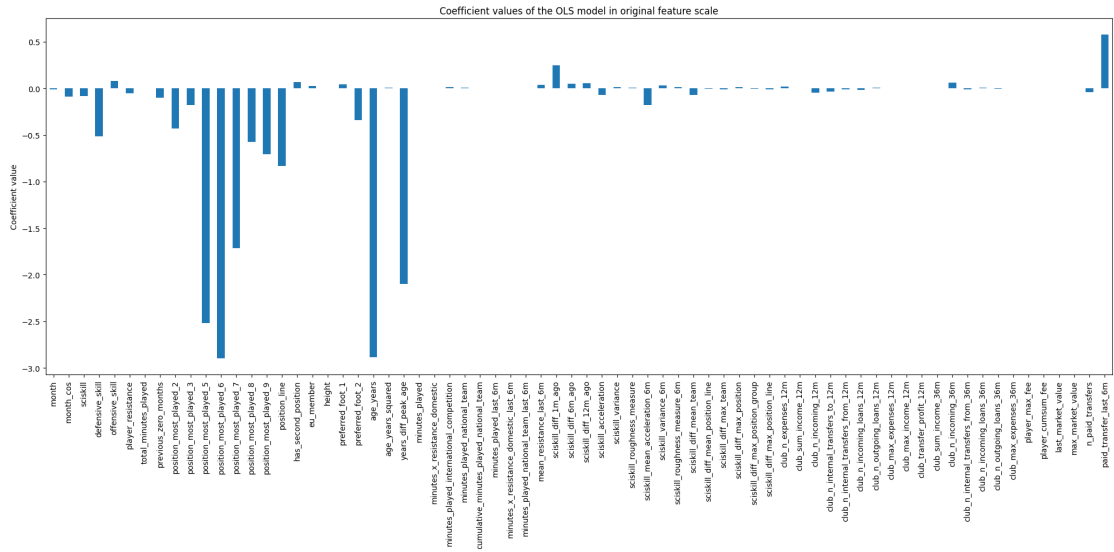


Figure C.3: The coefficients of the final OLS model in the original scale of the features in the SciSkill case study.

Table C.1 includes the bootstrap estimates of the standard deviations of the loss values on the test set. These were obtained using bootstrap resampling of the test set with 1000 samples. These models show that the distribution of the points appears to be symmetric. Next to that, they appear to have a single mode around the mean value. The distribution of bootstrap samples of the other models showed similar behavior. Therefore, the someone heuristic assumption is made in the current thesis that a difference in the performance of two models of more than two estimated standard deviations can be considered a significant difference.

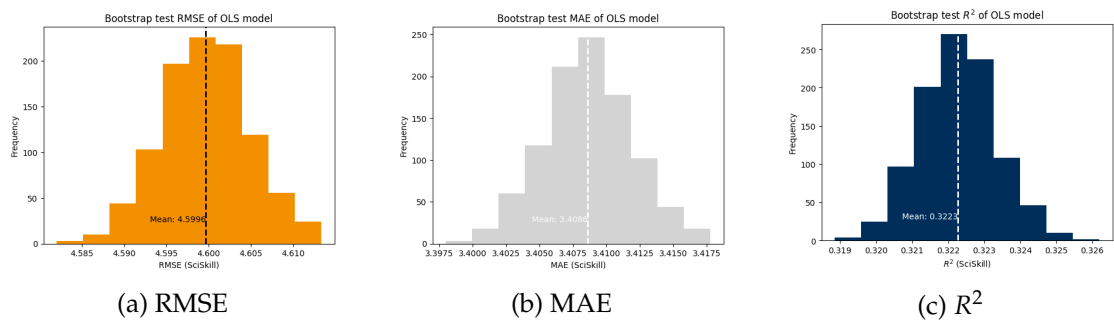


Figure C.4: Histograms of the distributions of the 1000 bootstrap samples for the different estimated loss functions of the OLS model in the SciSkill case study. The bootstrap mean is indicated by a vertical line.



be more smooth than Figure D.1b. Similar to the hyperparameter tuning for the feature selection, almost no improvement in the best-found objective function was found after only two function calls. This hyperparameter tuning could, therefore, be performed in fewer calls too.

Figure C.6 shows the feature importance of the final lasso model. It can be seen that the features related to age are the most influential features. Next to that, there are many features with only a limited feature importance. This means that the feature selection probably has been too conservative.

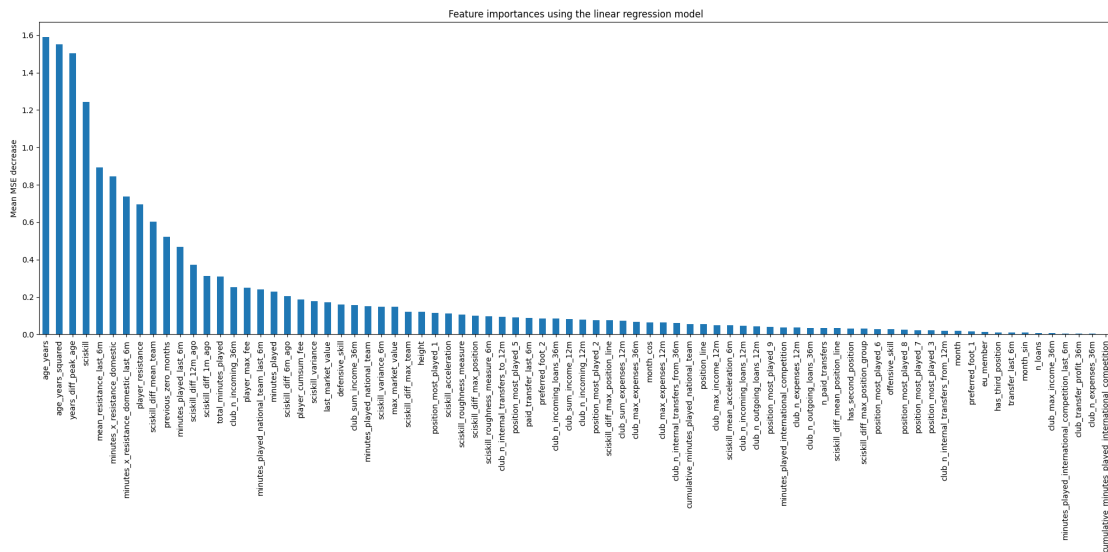


Figure C.6: The feature importance of the final lasso model in the SciSkill case study.

The coefficients of features in the final lasso model are visualized in Figure C.7. They show that the age in years ('age\_years') has a negative influence on the predicted SciSkill values one year later of the lasso model. This is opposite to the influence of this feature in the OLS model, which can be explained by the collinearity of the age-related features such as the age ('age\_years'), the age squared ('age\_years'), and the difference in age with the peak age of the players' position ('years\_diff\_peak\_age'). This also explains the lower coefficient values for these variables as they are not blown up by collinearity. The negative coefficient values of the SciSkill ('sciskill'), position line ('position\_line'), and difference with the mean SciSkill of the team ('sciskill\_diff\_mean\_team') are similar as in the OLS model.

Also for the lasso model, the rescaled coefficients were calculated as visualized in Figure C.8. Here, a difference of 1.0 in the feature will give the corresponding difference in the estimated value that is shown for that feature. For instance, if the difference with



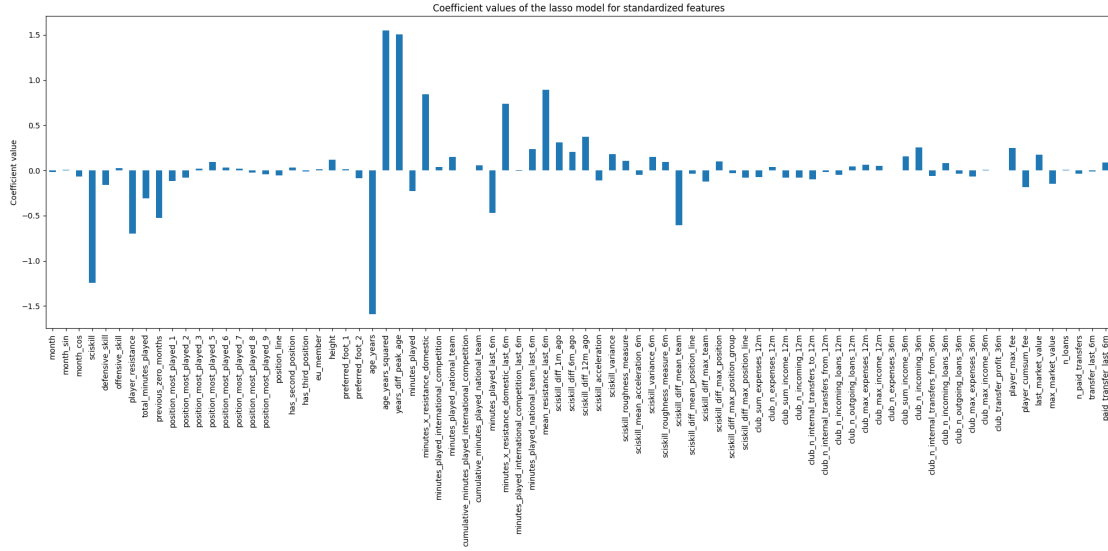


Figure C.7: The coefficients of the final lasso model for the standardized features in the SciSkill case study.

the peak age is 1 year, this means that the model will add  $1.0 \times 0.4 = 0.4$  to the expected rise in SciSkill in a year. It can also be seen that a player with '*position\_most\_played\_1*', corresponding to a left back, is predicted to have a 0.35 lower SciSkill the next year.

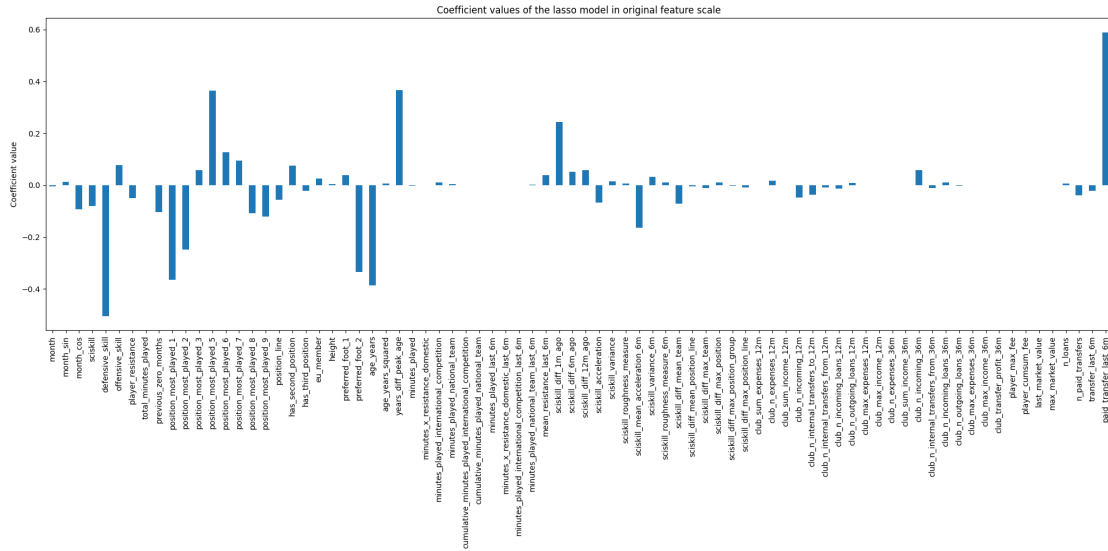


Figure C.8: The coefficients of the final lasso model in the original scale of the features in the SciSkill case study.

The losses on the training and test set were estimated and are given in Table C.1.

It can be seen that the training scores are slightly more positive, but the difference is limited. It can, therefore, be concluded that there have most likely not been problems with overfitting for the training of this model.

The estimated losses of the lasso model seem to have similar values as those of the OLS model. This could be expected as the used  $\alpha$  hyperparameter gives the importance of the regularization term. The Bayesian algorithm selected the value of  $\alpha = 10^{-3.23056} \approx 0.000588$  which is a relatively small value, most likely chosen because of the large number of data points. The regularization penalty does, therefore, not influence the actual predictions much, which makes it behave similarly to the OLS model.

It should, however, be noted that the feature importances in the OLS model are more distinct than for the lasso model. This is most likely due to the age-related features being highly correlated, which might cause a small blow-up in these coefficients. Because of the regularization term, this is prevented for the lasso model.

### C.1.3 | Linear mixed effect models

The linear mixed effect model was trained on the subset of the 20 most important features of the lasso model. The model was trained on these features with the nationality of the football player as the random effect. Figure C.9 shows the resulting feature importance. It can be seen that the player's resistance and the total minutes played per player are by far the most important features, as opposed to the OLS and lasso models.

The feature coefficients of the final linear mixed effect model are shown in Figure C.10. The coefficients of the age ('age\_years') and resistance level of the player ('player\_resistance') indicate that larger values for these features result in a lower prediction for the SciSkill one year later. Conversely, larger values for the age squared ('age\_years\_squared') and minutes played multiplied with the resistance level of the domestic competition in the last 6 months ('minutes\_x\_resistance\_domestic\_last\_6m') are associated with larger predicted values for the SciSkill a year later. These results are similar to the coefficients of the lasso model.

The feature coefficients rescaled to the original feature scales are shown in Figure C.11. They show that the increase in age of one year results in a predicted SciSkill of around 0.48 lower. At the same time, the increase of the age of a player will also give a smaller value of the difference between the age and the peak age. This will result also result in a smaller predicted SciSkill development value of 0.3. This means that

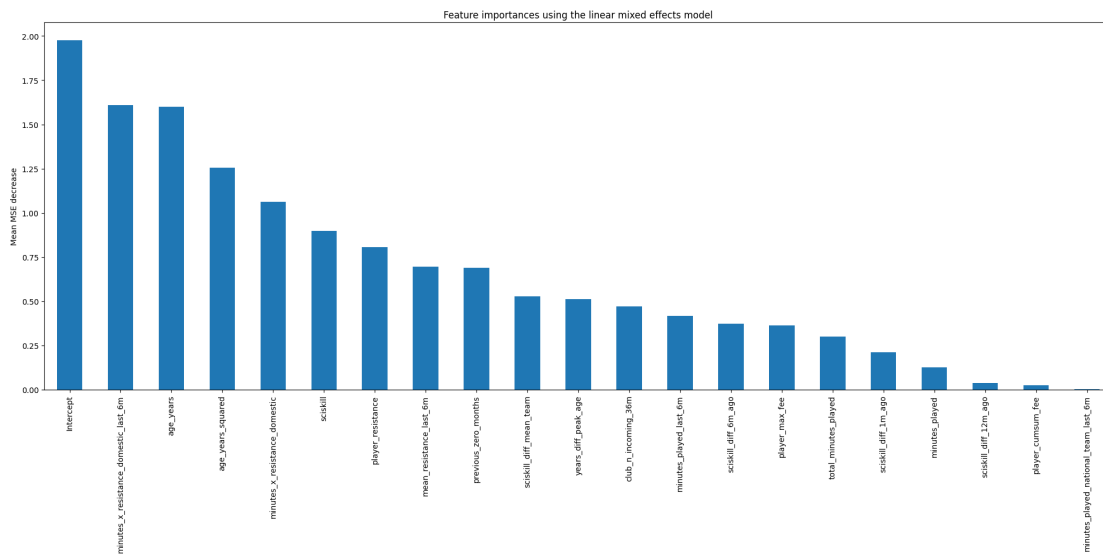


Figure C.9: The feature importance of the final linear mixed effects model in the SciSkill case study.

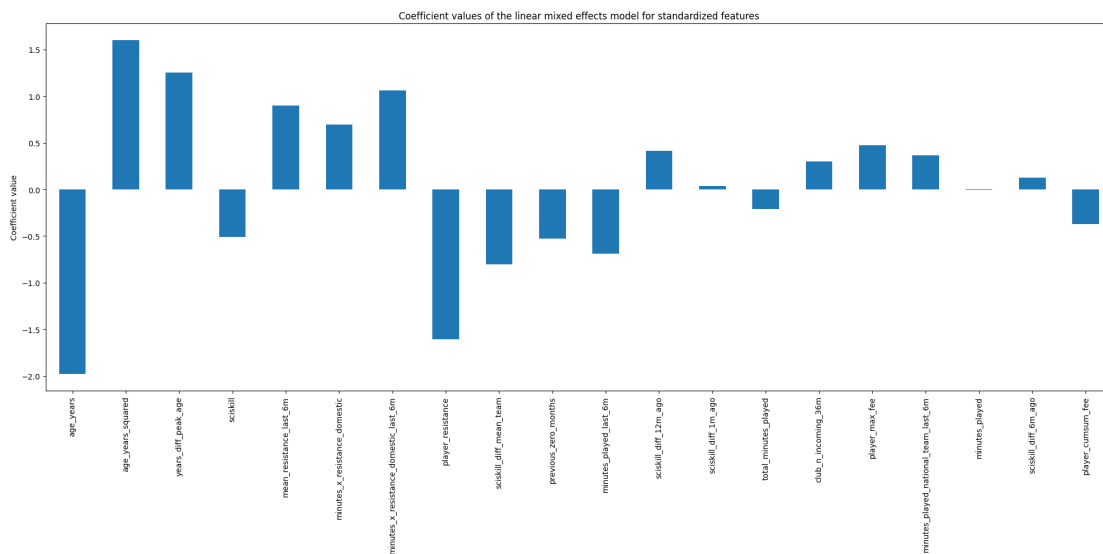


Figure C.10: The coefficients of the final linear mixed effects model for the standardized features in the SciSkill case study.

an increased age of one year makes the model predict SciSkill development values of around 0.8 lower. This means that older players are predicted to have less increase in SciSkill as could be expected.

Table C.1 shows the estimated training and test losses by the linear mixed effects

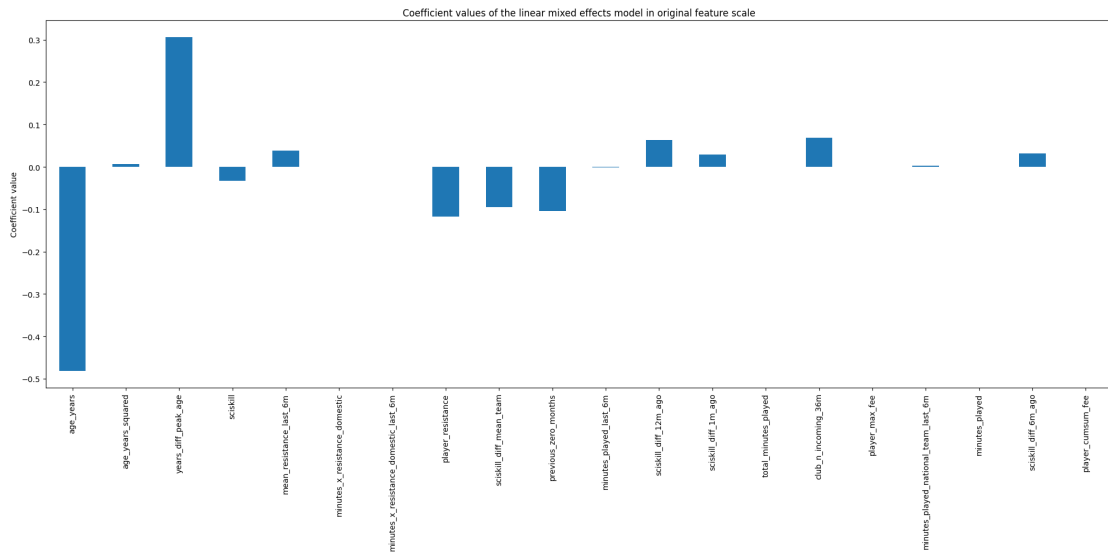


Figure C.11: The coefficients of the final linear mixed effects model in the original scale of the features in the SciSkill case study.

model. It is significantly less than the predictive performances of the OLS and lasso model for all loss functions. It can, therefore, be concluded that the performance was decreased by either the small subset of the features or the inclusion of the nationality of a player as a random effect. Further research could be done to draw this conclusion.

#### C.1.4 | Decision tree

To train the decision tree model, feature selection was first applied. To this end, the decision tree model was trained after a short hyperparameter tuning with 25 objective function calls. The process is visualized in Figure D.3. This process selected a tree depth of 24 with a minimum of  $2^{10} = 1024$  data points per leaf.

Figure D.3b shows that the differences in the objective value differ a lot for the parameter giving the minimal number of samples in a leaf, whereas the objective function for tree depth only slightly changes. As this means that the algorithm focussed on exploring the minimal leaf size, this might explain why Figure D.3a shows that the majority of the evaluated values of the tree depth are at the boundaries of the domain.

Figure D.3b also shows the interaction between the two hyperparameters. For the low and high values of the minimal leaf size, the objective values for the maximal depth seem to be similar, which indicates that there is only a limited interaction in these parts of the search domain. However, the objective function for minimal leaf sizes around

10 does differ horizontally for the different values of the maximal tree depth. This means that the tree depth influences the objective function for these parts of the domain, indicating some interaction between these hyperparameters.

Figure D.3c shows that after the first 10 evaluations, almost no improvement was found. As can be seen in Figure D.3a, only a specific part of the domain was discovered extensively by the algorithm. Although it is unlikely that this will provide large improvements, running the Bayesian algorithm for some more iterations might give improved hyperparameters.

The feature importances of the features with the added noise features are visualized in Figure C.12. It shows that there is only a small part of the features with a clear nonzero feature importance. It does, however, show that the noise variables were almost never selected by the decision tree algorithm. This could be expected as they could not be selected by coincidence as there is no random part in this decision tree model. This means that the features with a nonzero feature importance were selected by the algorithm. Because many features had a really small nonzero feature importance, this resulted in the selection of 70 out of the 86 features. As many of these features only have a limited feature importance, the algorithm probably was too conservative with the feature selection.

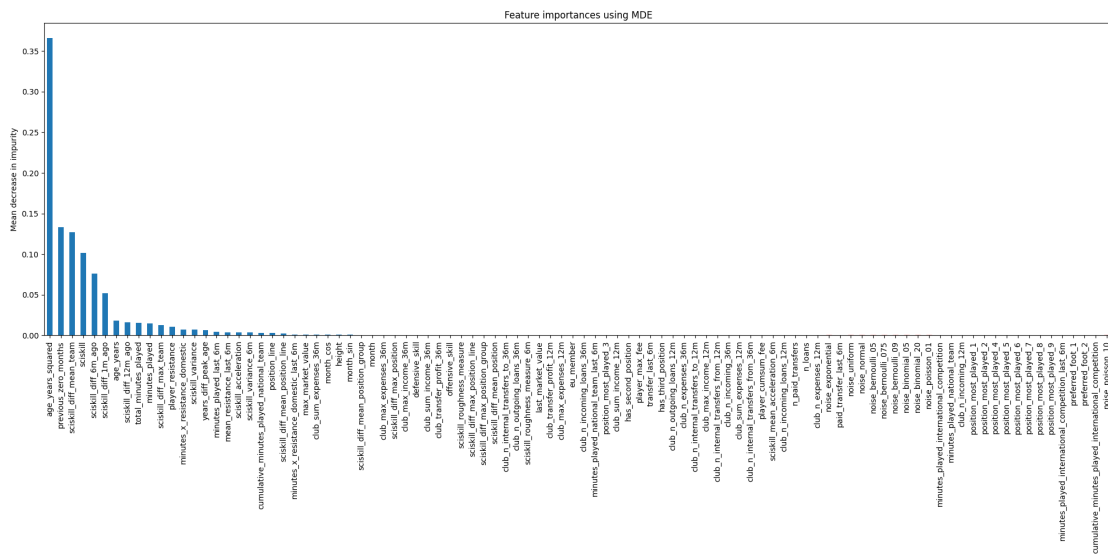


Figure C.12: The feature importance of the features in the decision tree model for feature selection in the SciSkill case study. The first 70 features were selected and the feature importances of added noise variables are shown in red.

After the feature selection, the hyperparameter tuning was applied to the decision

tree model with 50 function calls for the selected features. The process is visualized in Figure D.4. It shows some clearly distinct patterns compared to those for the hyperparameter tuning for feature selection. Figure D.4a shows that more non-boundary points were evaluated, which might be caused by the increased number of function calls. The two-dimensional estimation of the objective function in Figure D.4b shows that the differences in the objective function are less vertically oriented. This means that there is more interaction now that the unimportant and noise variables are removed.

The convergence plot shown in Figure D.4c also shows similar behavior as in Figure D.3c. This is reflected by the fact that no significant improvement was obtained after the 14th function call. This means that the increased number of function calls did not necessarily improve the best-found hyperparameter values.

However, the marginal influence plot of the minimal number of samples in the leaves still shows larger differences than the marginal influence plot of the tree depth. It can also be seen that the selected combination of hyperparameters is at the lowest point of the marginal influence of the minimal number of samples in the leaves, while it does not have a value that seems to be optimal in the marginal influence plot for the tree depth. This means that the minimal number of samples in the leaves is the most important parameter.

The feature importances of the final decision tree model are visualized in Figure C.13. The feature importances show nearly identical behavior compared to the feature importances for feature selection in Figure C.12. As many of the variables have almost no importance, a more rigorous feature selection method might have been better for the decision tree method.

The training and test losses of the final decision tree are given in Table C.1. It shows that the training losses are more positive compared to the test losses, which might indicate that a small amount of overfitting has occurred. But as the differences are limited, this probably does not cause problems.

The results of the decision tree also show an improved performance compared to all linear models. As the decision tree is still a fairly simple model, this is a surprising result. A possible explanation for this is the fact that the decision tree is able to take interactions between the different into account, whereas the linear models could not due to computational infeasibility. This indicates that there might be interactions within the features in the data.

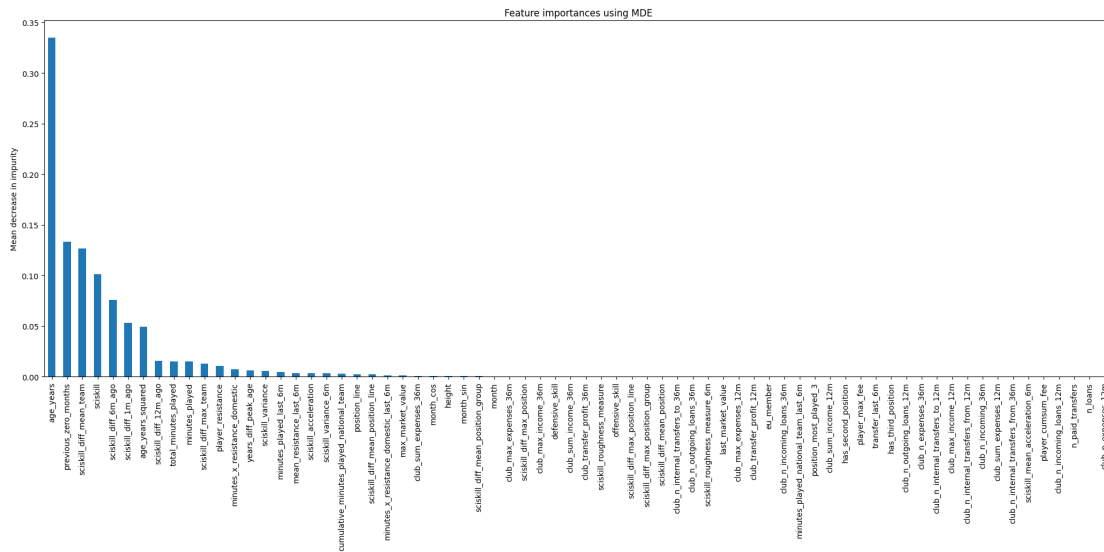


Figure C.13: The feature importance of the final decision tree model in the SciSkill case study.

### C.1.5 | Random forest

To perform the feature selection for the random forest model, a model was trained on the features after hyperparameter tuning with 19 function calls and added noise variables. The tuning process is visualized in Figure D.5. Figure D.5a shows that the tried parameter values are quite uniformly distributed for all hyperparameters. It should be noted that this model has four hyperparameters which is more than the models before. This means that the Bayesian algorithm needs more function evaluations to get near the optimal value. This is also shown by Figure D.5c as it still decreases in the last function calls. This means that the algorithm has not converged yet. Due to the high computational costs for the random forest algorithm, it was decided not to increase the number of function calls.

Because of the relatively low amount of function calls, the visualizations of the objective function in Figure D.5b should be interpreted with care. On top of that, the model has more than two hyperparameters. This means that the visualization of the objective function consists of multiple two-dimensional marginalizations of the hyperparameter pairs. The actual objective function can, therefore, not be inferred using this method.

However, it is possible to draw certain conclusions. It can be seen that the one-dimensional marginal plots in Figure D.5b show that more difference is found by the

algorithm in the hyperparameters ‘max\_depth’ and ‘min\_samples\_leaf’. It can also be seen that the best-found values are near the optimal values for these parameters. It can, therefore, be concluded that the maximal tree depth and the minimal samples of the leaf are the most hyperparameters in this problem for the random forest model.

The feature importances with added noise variables resulting from this model are shown in Figure C.14. It can be seen that many features have a larger feature importance than with Figure C.12. This also holds for the noise variables, which are more important than more than half of the features. Therefore, 41 features with a larger feature importance than the noise variables were selected by this feature selection method based on a random forest model. As discussed in the literature study, the random forest

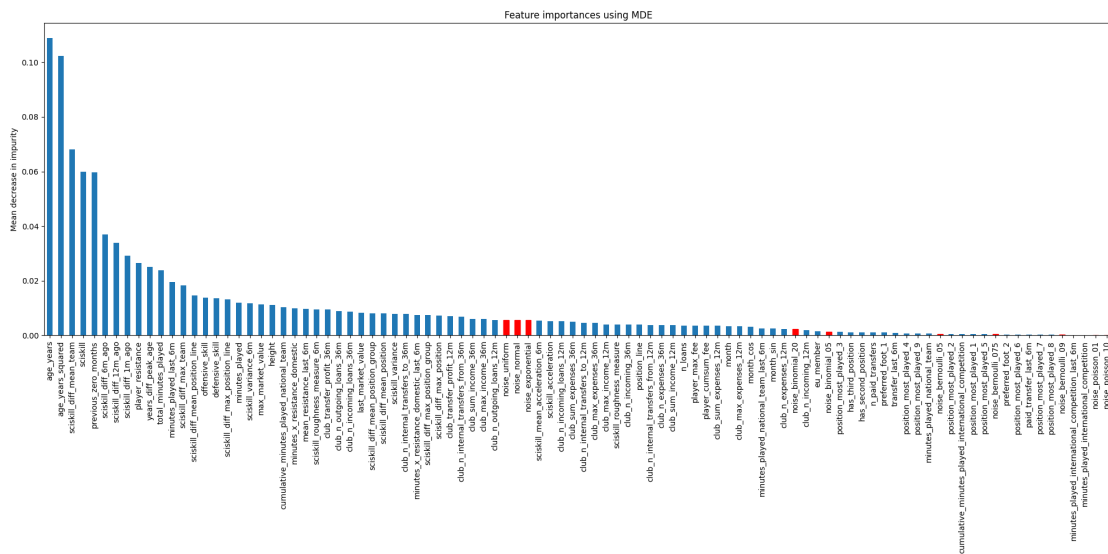


Figure C.14: The feature importance of the features in the random forest model for feature selection in the SciSkill case study. The first 41 features were selected and the feature importances of added noise variables are shown in red.

models are known to have favor variables with a variety of possible values, such as continuous variables. This is also reflected in the results in Figure C.14, as most binary and categorical variables such as the position variables have a small feature importance. It is also shown by the fact that the categorical noise variables were given a higher importance by the random forest. It can, therefore, be concluded that in this application, the random forest model favors the use of non-discrete variables.

Figure D.6 shows the process of the hyperparameter tuning of the final random forest model trained with 25 function calls and the selected features. It can be seen in Figure D.6a that the distribution of the assessed hyperparameters is not uniformly



distributed, in contrast with Figure D.5a. Additionally, Figure D.6 shows that the best-found value only minimally decreased after 9 function calls. This shows that the removal of unimportant variables speeds up the optimization problem.

The marginal influences of the hyperparameters are shown in Figure D.6b. As opposed to Figure D.5b, the influences of the number of trees ('n\_estimators'), the sampling fraction of the features ('max\_features'), and the sampling fraction of the samples are visible in the marginal influence plots ('max\_samples'). A larger number of random forests marginally gives better estimations as could be expected via theorems on random forests. Using a larger fraction of the features and the samples increases the predictive performance in general.

It can also be seen that there seem to be visible interactions within the hyperparameters. This can be concluded from the fact that the plots in Figure D.6b do not consist of vertical or horizontal patterns. The marginal plot of the maximal tree depth and the minimal samples in a leaf shows that the best models are obtained for deep trees with a relatively low minimal number of samples in the leaf.

Next to that, the patterns in the plots for the sampling fractions of the features and the samples are really similar. This holds for both the one-dimensional and two-dimensional marginal plots. It, therefore, seems that both have a similar influence on the performance of the model.

Figure C.15 shows the feature importance of this final random forest model. It can be seen that age-related features are the most important. Additionally, the information about how long a player has not played a game, and about the SciSkill values compared to the team seem to be of relatively important. It also shows that the SciSkill and features describing the difference with historical SciSkill values are used by the model to make predictions. This means that there appears to be information in the time series of the SciSkill values.

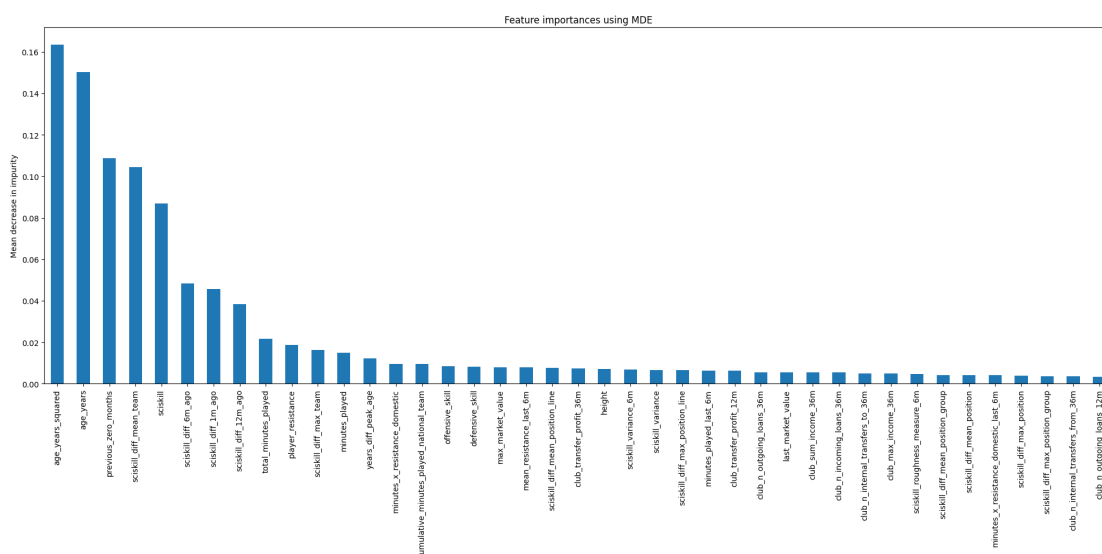


Figure C.15: The feature importance of the final random forest model in the SciSkill case study.

The feature importance in Figure C.15 indicates that the random forest does not have many discrete variables to make predictions. This is not surprising because of the known bias of random forest models against discrete variables with a low number of possible values. This bias was also apparent in the feature importances of the feature selection model.

The estimated loss values on the training and test set of the random forest model are shown in Table C.1. The values are better than those of the decision tree, which means that the bagging procedure of the random forest increases the predictive quality.

On top of that, the random forest shows that the loss values of the training set are clearly lower than the test losses. This means that overfitting is occurring for the random forest model. If more conservative hyperparameters had been chosen than those found by the Bayesian optimization algorithm, the performance of the model could have been increased.

## C.1.6 | XGBoost

The feature selection for the XGBoost was performed by selecting the features that had a larger feature importance than added noise features. This was done by fitting a model after a hyperparameter tuning of the XGBoost model with 50 function calls. Figure D.7 and Figure D.8 show the visualizations of this hyperparameter tuning process.

The XGBoost model has eight hyperparameters to be tuned, which is the most of all models in the current thesis. The first hyperparameter tuning is visualized in Figure D.7 in which seven parameters were tuned. It can be seen that the  $\eta$  parameter had a significant influence, which is the learning rate of the boosting algorithm. Next to that, the subsample rate of the data points ('subsample') and the maximal tree depth ('max\_depth') appear to be influential hyperparameters. This can be seen that the values in their marginal plots seem to influence the predictive values quite much.

Figure D.7b also shows that the sampling rate of the features per tree level ('colsample\_bylevel') and the regularization term  $\alpha$  ('alpha') have almost no interaction with the other hyperparameters. This is shown by the fact that these hyperparameters show horizontal and vertical patterns in the two-dimensional marginal plots.

The convergence plot shown in Figure D.7c shows that most improvements in performance were found in the first 20 objective calls. After this, only a small improvement was obtained with the last 30 function calls.

The second part of the hyperparameter optimization was to optimize the number of trees ('n\_estimators'). Figure D.8 contains visualizations of the process of the Bayesian algorithm. Figure D.8b shows that tuning the number of decision trees can provide a small improvement in performance. It also shows that the optimal value is attained with the upper bound given to the optimization algorithm. This means that adding more decision trees would increase the performance of the models. However, this would also result in longer computation times and the gains seemed to be limited as the curve has almost flattened out at that point.

Figure D.8c shows that all improvement was attained in one step of the optimization. This was the step where it tested the value of the maximal bound, which is expected behavior given the seemingly monotone relation of the number of decision trees.

The tuned hyperparameters were used to train a model to perform feature selection. Figure C.16 shows the feature importance of the features and the noise variables in the XGBoost model. It can be seen that the XGBoost model is good at filtering out the noise variables as all noise variables have a very low feature importance. This could also mean that all other features appear to contain some information on the SciSkill development of players over the next year. It can also be seen that the month-related features have a small feature importance. Still, these features were selected as they were considered to be discrete variables and they have a larger feature importance than all discrete noise variables.

The same hyperparameter tuning process was applied to the features without the noise variables as visualized in Figure D.9 and Figure D.10. It shows that the influences

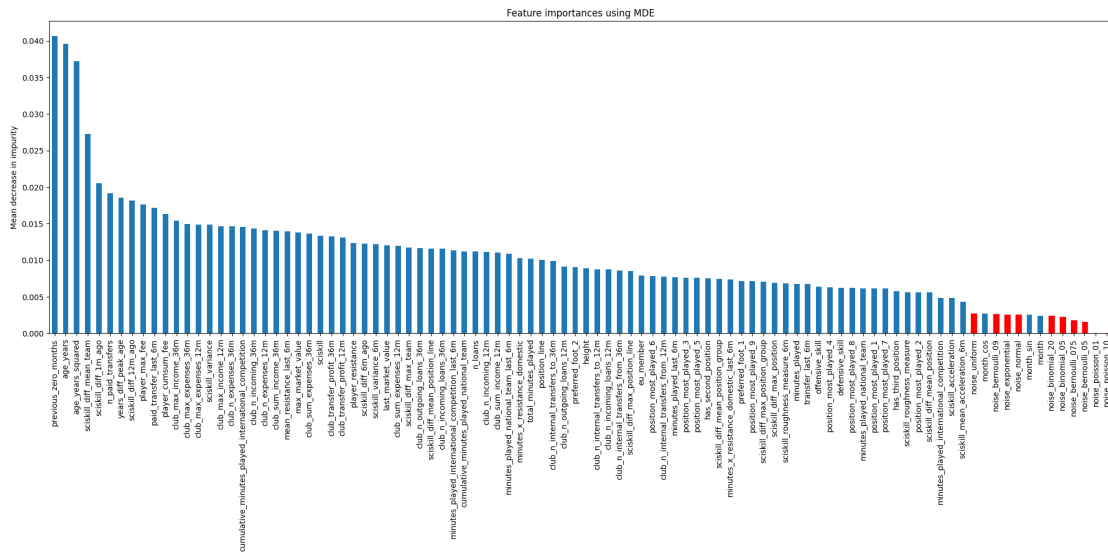


Figure C.16: The feature importance of the features in the random forest model for feature selection in the SciSkill case study. All features were selected and the feature importances of added noise variables are shown in red.

of the hyperparameters significantly differ without the presence of the noise variables. Although  $\eta$  ('eta') appears to have a similar behavior, the influence of the data subsample rate ('subsample') is significantly different. Instead of a monotone influence as in Figure D.7b, it shows multiple local minima and maxima in Figure D.9b. Whereas the sampling rate for each tree ('colsample\_bytree') had a visual influence on the performance in the marginal plot in Figure D.7b, it does not seem to be influential in Figure D.9b. Conversely, the influence of the sampling rate of columns for each tree level was clearly visible with the presence of noise variables although it does not seem to have a clear influence without noise variables. This might be occurring because of the fact that the column sampling rate for the tree and per level have similar functions.

Figure D.9c shows that the algorithm still had improved just 10 objective function calls before the end of the tuning process. It could be possible that further hyperparameter tuning would increase the predictive quality of the models. This was not explored because of the large computational costs of training an XGBoost model.

The tuning process of the number of decision trees ('n\_estimators') is shown in Figure D.10. It can be seen that the estimated loss appears to have a decreasing relationship with the number of decision trees similar to in Figure D.8b. The absence of noise variables does not seem to influence the behavior of the number of decision trees very much.

The feature importances of the final XGBoost model are shown in Figure C.17. It can be seen that the differences between the most important features are more apparent than in Figure C.16 with the absence of the noise variables. It can be seen that the number of months without a game of a player ('previous\_zero\_months') is the most important feature. This is not surprising as the SciSkill gives a penalty for inactivity if a player did not play for a long time. As the SciSkill is not updated in the meantime, this means that this penalty is given in the first game after the absence. It is, therefore, not surprising that it has a large influence on the prediction.

Table C.1 shows the estimated train and test losses for the XGBoost model. This shows that the other model seems to show overfitting behavior as the training loss is much too positive compared to the test loss. However, despite the overfitting behavior of the model, it attains the lowest test loss so far. This means that it is able to generalize the patterns in the training data the best, despite the fact that it overfits. This surprising behavior indicates that the prediction in this case study appears to be hard to model.

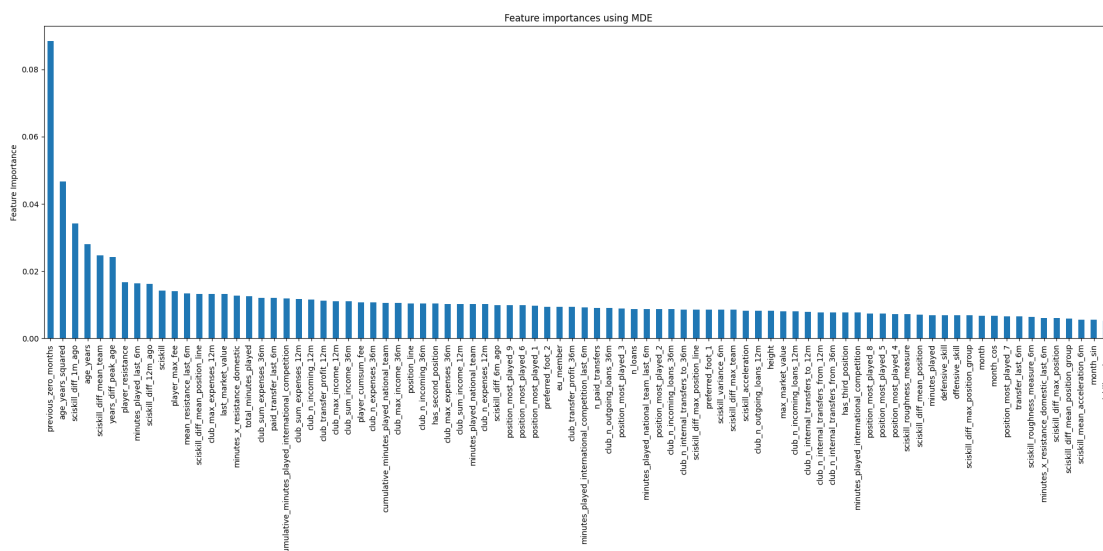


Figure C.17: The feature importance of the final XGBoost model in the SciSkill case study.

A surprising occurrence is that the best-attained value of the adjusted cross-validation was lower for the hyperparameter tuning with noise variables instead of the one without the noise variables. This is visible in the values Figure D.8c and Figure D.10c. This might be explained by the fact that the presence of noise forces the hyperparameters to be chosen more conservatively. As the model appears to be overfitting, this would decrease the overfitting behavior and improve the predictive performance.

### C.1.7 | kNN

The features chosen for the kNN model were the age in years of the player, the number of months since the last game, and the SciSkill values of the last 12 months as times series. The hyperparameter tuning process is visualized in Figure D.11.

Figure D.11b shows that the estimated loss is decreasing for values larger than 20 for  $k$ , the number of neighbors. An increased number of neighbors could, therefore, have been interesting to investigate. The results also show a local minimum around  $k = 10$ , which might be caused by the fact that certain subsets of the test set can be predicted better with a fewer number of neighbors. The results also show that the neighbor weighting method based on the distance with the data point ('uniform') marginally gives a higher loss function. There appears to be a large interaction of the performance between the number of neighbors and the neighbor weighting function. In the end,  $k = 50$ , and the uniform weighting method were selected.

Table C.1 gives the estimated loss functions on the training and test set. The results do not imply that overfitting is a large problem, although it might be apparent. At the same time, it has a worse test loss as the random forest and linear models. The normal kNN, therefore, does not have a competitive predictive accuracy compared to the other implemented models.

### C.1.8 | kNN with Mahalanobis distance

The kNN with Mahalanobis distance was trained on the same features as the normal kNN implementation. The hyperparameter tuning process was visualized in Figure D.12.

It can be seen that the marginal estimation of the objective function for this model is nearly flat after  $k = 20$ . This means that the model is able to obtain the same amount of information using less data. Nevertheless, the algorithm still found some small improvement in predictive accuracy for the larger values of the hyperparameter  $k$  and selected  $k = 47$  as the best number of neighbors.

The 2-dimensional estimation of the objective function in Figure D.12b shows different interaction effects as Figure D.11b. The performance of the 'inverse distance' weighting method is more competitive with the other loss functions. The marginal plot for the weighting method even shows that the 'inverse distance' method attains the lowest marginal loss. However, due to interactions with the number of neighbors, the uniform weighting method found to be the best weighting method.

Table C.1 shows the training and test score estimates of the kNN model with Mahalanobis distance. The results do not indicate a large difference in the training and test losses. The test losses are slightly better than those of the normal kNN model. This means that the introduction of the Mahalanobis distance increases the accuracy. Nonetheless, this did not result in loss values that were competitive with the linear and tree-based models.

### C.1.9 | kNN with RReliefF weights

The kNN model with RReliefF weights was trained on the same features as the normal kNN implementation. The hyperparameter tuning process was visualized in Figure D.13.

In this figure, it can be seen that the kNN model with RReliefF weights behaves similarly to the normal kNN model with Mahalanobis distance. The largest difference in the behavior is the fact that the ‘inverse distance’ weighting method performs relatively worse. The Bayesian algorithm selected the uniform neighbor weighting method as the best.

The results also showed that the predictive performance appeared to increase with the number of neighbors  $k$ , albeit in a less smooth relation. It could still have been beneficial to have increased the number of neighbors. With 50 as a maximum, the best-found number of neighbors was found to be  $k = 47$ .

The training and test scores of the kNN with RReliefF weights are given in Table C.1. The results show that the kNN method with RReliefF weights performs worse than the kNN model with Mahalanobis distance and the normal kNN model. This is in line with the results in Appendix B, where it was found that the RReliefF weights did not significantly the predictive performance of the methods.

The RReliefF model also resulted in the feature importance as shown in Figure C.18. For instance, the number of months since the most recent game of a player (‘previous zero months’) appears to have almost no feature importance according to the RReliefF algorithm. This is surprising because this appeared to be an important variable in other models. It might be possible that this is due to the fact that most samples have a value of 0 for this variable. As the RReliefF algorithm determines the differences within each feature for the  $k$  closest points, it might be the case that this value is often the same for points close to each other. This would make the RReliefF method incorrectly determine that the feature has a low importance. This could possibly be solved by adjusting the RReliefF algorithm to also take into account the  $k$  furthest points.

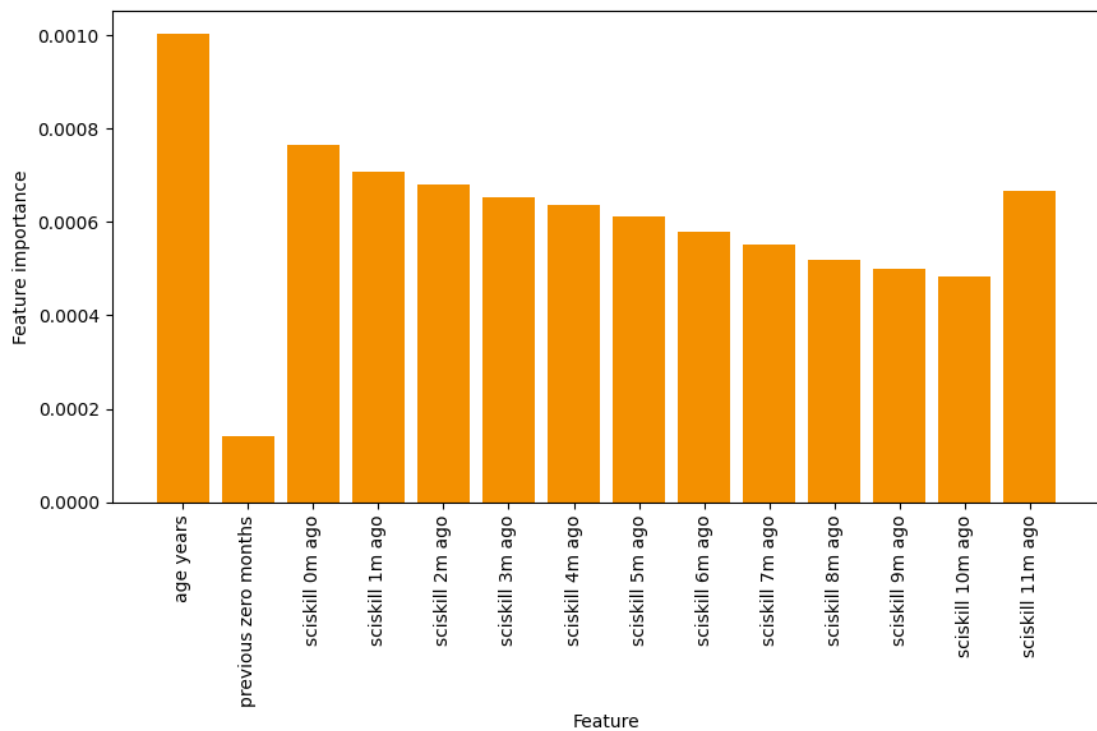


Figure C.18: The feature importance of the final kNN model with RReliefF weights in the SciSkill case study.

Still, the results show that the most important feature is the age of the player in years. The most recent values in the time series features appear to be the most influential. Surprisingly, the oldest value of the time series features also appears to have an important influence. This data point was most close to the same data point a year ago, and might, therefore, contain information about seasonal changes. In this way, the RReliefF algorithm shows that the age, the most recent time series value, and the oldest time series value were the most important features in the RReliefF kNN model.

## C.2 | Estimated Transfer Value case study

### C.2.1 | Ordinary least squares

The backwards feature selection algorithm was applied using the OLS regression model using a significance threshold of 0.001. This resulted in 31 features being selected out of 57 features.



Model	Train set			Test set		
	RMSE	MAE	R <sup>2</sup>	RMSE (Std. dev.)	MAE (Std. dev.)	R <sup>2</sup> (Std. dev.)
OLS	1,506,568	410,451	0.1465	1,795,871 (43,924)	533,513 (6,492)	0.1059 (0.0148)
Lasso	1,506,621	408,783	0.1464	1,795,462 (43,978)	531,579 (6,494)	0.1063 (0.0140)
LME	1,507,395	409,854	0.1455	1,797,849 (44,063)	531,049 (6,509)	0.1039 (0.0141)
Decision tree	1,410,098	319,703	0.2523	1,783,137 (43,923)	462,347 (6,492)	0.1185 (0.0141)
Random forest	991,242	266,296	0.6305	1,704,637 (41,258)	441,403 (6,215)	0.1944 (0.0220)
XGBoost	1,243,872	297,811	0.4182	1,737,930 (41,412)	453,343 (6,381)	0.1626 (0.0181)
kNN	1,307,758	299,970	0.3569	1,766,151 (44,580)	457,924 (6,420)	0.1352 (0.0171)
kNN Mahalanobis	1,348,455	309,709	0.3162	1,743,481 (43,385)	450,904 (6,309)	0.1573 (0.0165)
kNN RReliefF	1,342,309	318,041	0.3224	1,760,266 (42,905)	460,119 (6,461)	0.1410 (0.0176)

Table C.2: The estimates on the train and test set for the different loss functions in the Estimated Transfer Value case study. The standard deviations of the estimates on the test set were calculated using 1000 bootstrap samples. The values for the RMSE and MAE are given in euros.

Figure C.19 shows the feature importance of the selected features for the OLS model. It can be seen that the ETV-related features ('etv'; 'etv\_diff\_6m\_ago'; 'etv\_diff\_12m\_ago') are considered to be the most important features. Next to that, historical transfer information of both the club and the player is considered important ('club\_average\_paid\_fee\_36m'; 'player\_max\_fee'; 'player\_cumsum\_fee') by the OLS model.

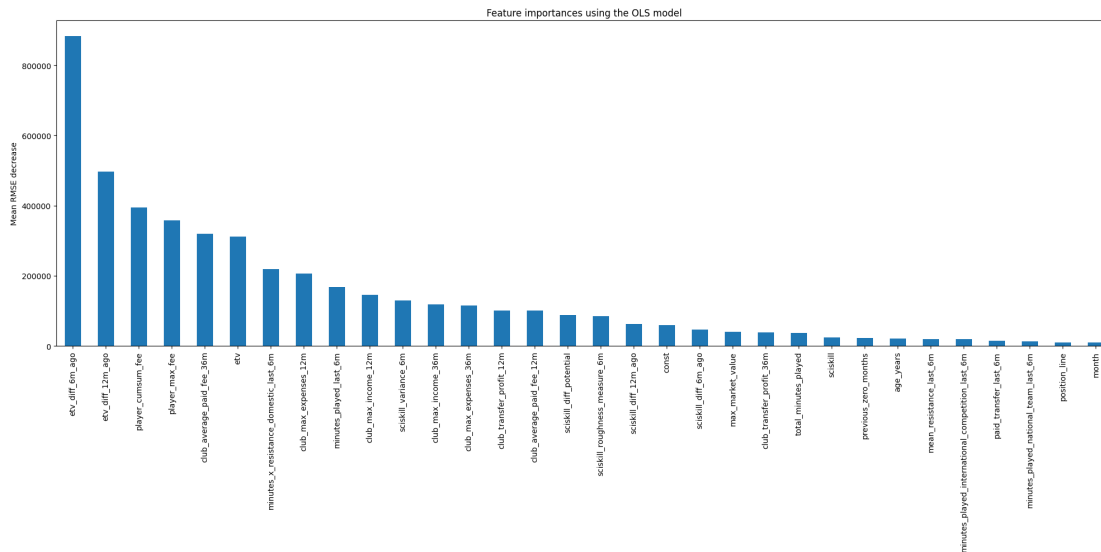


Figure C.19: The feature importance in the RMSE decrease in euros for each feature for the OLS model in the Estimated Transfer Value case study.

The coefficients of the OLS model are visualized in Figure C.20. The results show that a positive development of the ETV in the last 6 months results in higher prediction for the ETV one year later, whereas a positive development of the ETV in the last 12

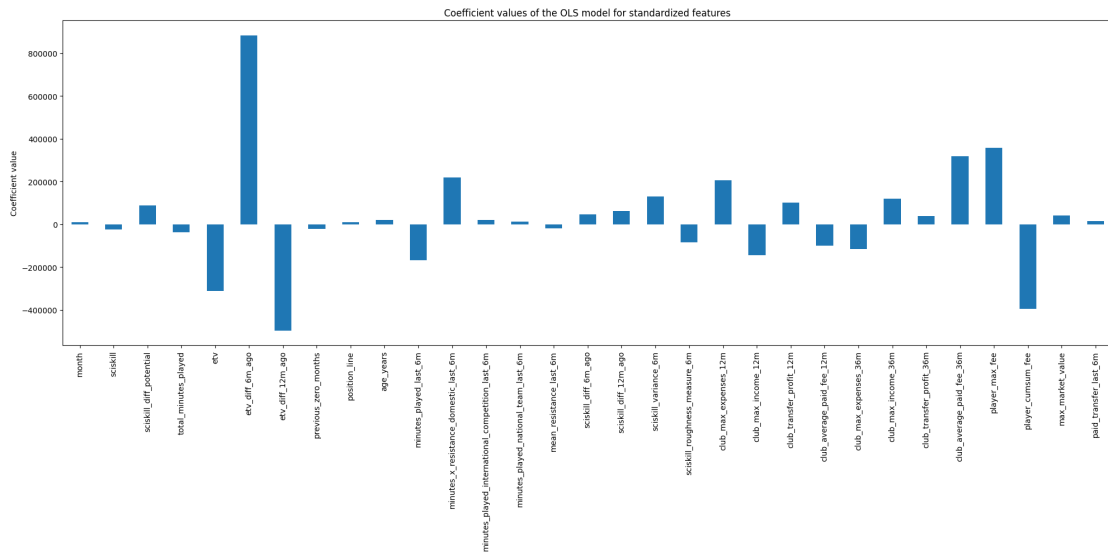


Figure C.20: The coefficients of the final OLS model of the standardized features in the Estimated Transfer Value case study.

months is associated with a lower predicted value of the ETV one year later. High values for the current ETV result in lower predictions of the ETV one year later. This might be due to the fact that players with a large ETV value will decrease in value somewhere in their career combined with the fact that it might be hard for them to increase their value as it is already one of the highest.

Figure C.21 describes the coefficients of the features in the OLS model in their original scale. It can be found that having been inactive for some months ('previous\_zero\_months') has a negative influence on the development of the ETV. It can also be seen that the influence of having had a paid transfer in the last six months ('paid\_transfer\_last\_6m') has a positive influence on the ETV. This can be because of the fact that the ETV-values are influenced by recent transfers. The features with large-scale differences in their original features, such as the ETV, and club expenses seem to have small coefficient values. These are due to the fact that their differences are of a larger scale and these might, therefore, have a large influence. This is actually the case as the ETV itself has quite a large feature importance as shown in Figure C.19.

The values of the estimated loss values are given in Table C.2. It can be seen that the test losses are slightly larger than the training losses and that the value of the test RMSE is around 1.8 million euros. This is relatively large and at the same time, the  $R^2$  is relatively small. This means that the OLS model gives less information compared to taking the mean in this case study than in the SciSkill case study.

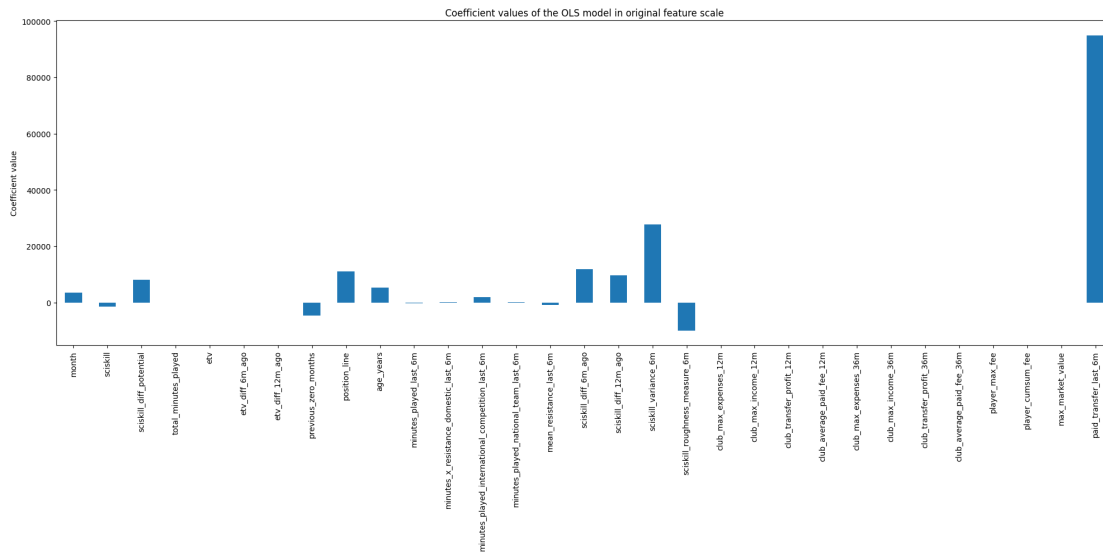


Figure C.21: The coefficients of the final OLS model in the original scale of the features in the Estimated Transfer Value case study.

## C.2.2 | Lasso

The feature selection process for the lasso model was performed by selecting the features with nonzero coefficient values in a lasso model. To this end, a short hyperparameter tuning was performed of which the process is visualized in Figure D.15. It can be seen that the function is relatively flat for small values of the hyperparameter  $\alpha$  and rises steeply for values with  $\alpha > 10^4$ . There appears to be a minimum around  $\alpha = 10^{-3.5}$  which is reflected by the fact that almost all function evaluations were around this point as shown in Figure D.15a.

The lasso model trained with these hyperparameters was trained on the data. The feature importances are displayed in Figure C.22 and show that most features have nonzero coefficients as 51 out of 57 features have been selected. It can be seen that the last 20 features have limited feature importance and it could, therefore, be the case that the feature selection method was not rigorous enough and a higher value of  $\alpha$  should have been chosen.

The hyperparameter process was again applied to the subset of the selected features as visualized in Figure D.16. The estimated objective function in Figure D.16b and the function evaluations in Figure D.16a are similar to those of the hyperparameter tuning for the feature selection. A similar value of  $\alpha$  was selected and the algorithm reached this value much faster as reflected by Figure D.16c.

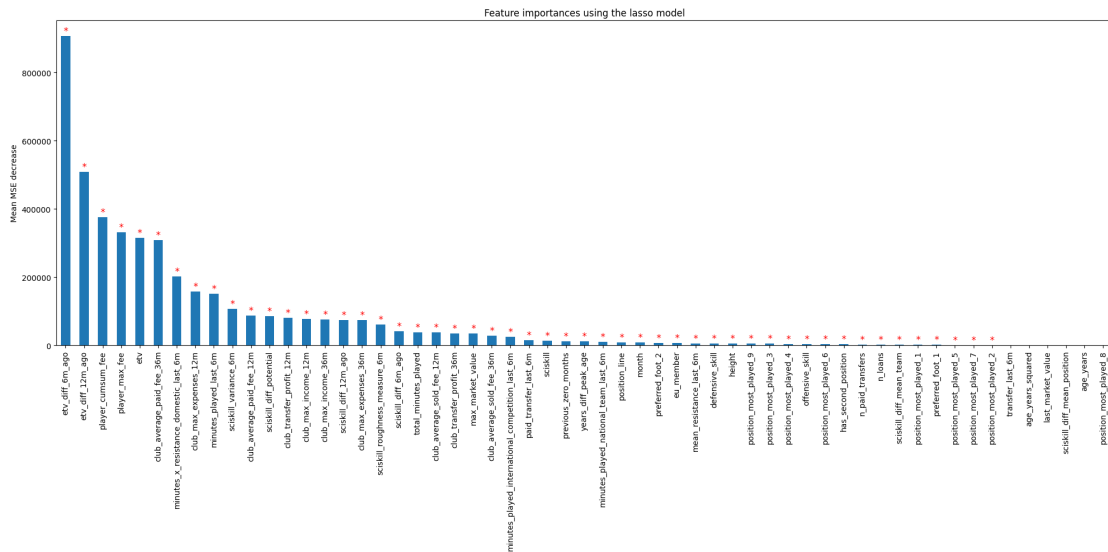


Figure C.22: The feature importance of the features in the lasso model for feature selection in the Estimated Transfer Value case study. Selected features are indicated with a red (\*).

This resulted in a lasso model with the feature importances as depicted in Figure C.23. It can be seen that the feature importances of the final lasso model are similar to those of the lasso model for feature selection. Moreover, both the lasso and the OLS model give a large importance to similar features, which is, for instance, reflected in ETV-related variables being important.

The estimated values of the coefficients in the final lasso model are visualized in Figure C.24. The results show that the coefficients for the ETV-related features ('etv'; 'etv\_diff\_6m\_ago'; 'etv\_diff\_6m\_ago') had nearly identical coefficients as in the OLS model. It also indicates that the average fee paid by the club in the last three years ('club\_average\_paid\_fee\_36m') results in a larger predicted value for the ETV one year later. The influence of this feature is surprisingly much larger than that of the average sold player in the last three years ('club\_average\_sold\_fee\_36m'). This might happen because the average fee paid by a club might represent the prestige of the club or the club's competition, which is could be expected to result in larger transfer values.

The feature coefficients in their original scale are visualized in Figure C.25. These coefficients are very similar to those of the OLS model, although more variables have been selected by this method.

The estimated losses of the lasso model for the ETV case study are given in Table C.2. It can be seen that the differences in the test loss functions with those of the OLS model



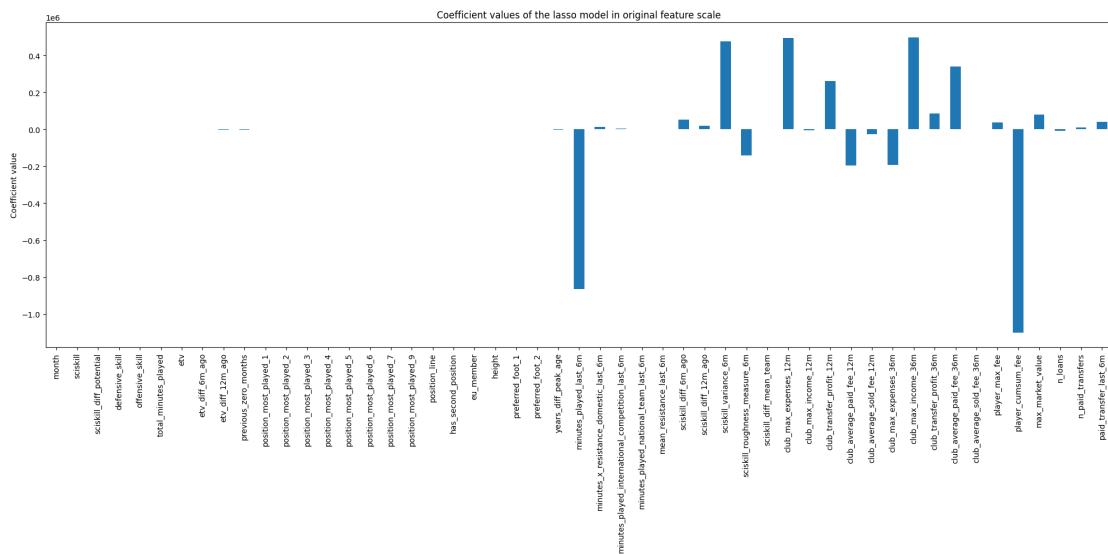


Figure C.25: The coefficients of the final lasso model in the original scale of the features in the Estimated Transfer Value case study.

### C.2.3 | Linear mixed effect model

The third linear model is the linear mixed effect model. The 20 features with the largest feature importance of the lasso model were used for this model and the first nationality of the player was taken as the random effect.

This resulted in the feature importances as shown in Figure C.26. It can be seen that again the same features were considered significant, but that the most important ‘feature’ was the intercept. The other features are similarly behaving as in the OLS and lasso models, except for the fact that the importance of feature describing the difference between the current ETV and the ETV of six months ago (‘etv\_diff\_6m\_ago’) is a little bit less important. This might have been caused by the introductions of the random intercepts in this linear mixed effect model.

Figure C.27 shows the estimated coefficients of the linear mixed effects model. The coefficients of the ETV-related features (‘etv’; ‘etv\_diff\_6m\_ago’; ‘etv\_diff\_6m\_ago’) are similar to the coefficients of the OLS and lasso models. This also holds for the coefficients corresponding to the other selected features, which means that the inclusion of the random effects did not have a large influence on the estimated coefficient values.

The rescaled model coefficients are visualized in Figure C.28. These values have completely different scales compared to those of the OLS and lasso models as no binary features have been included. As binary values have a relatively small difference in their

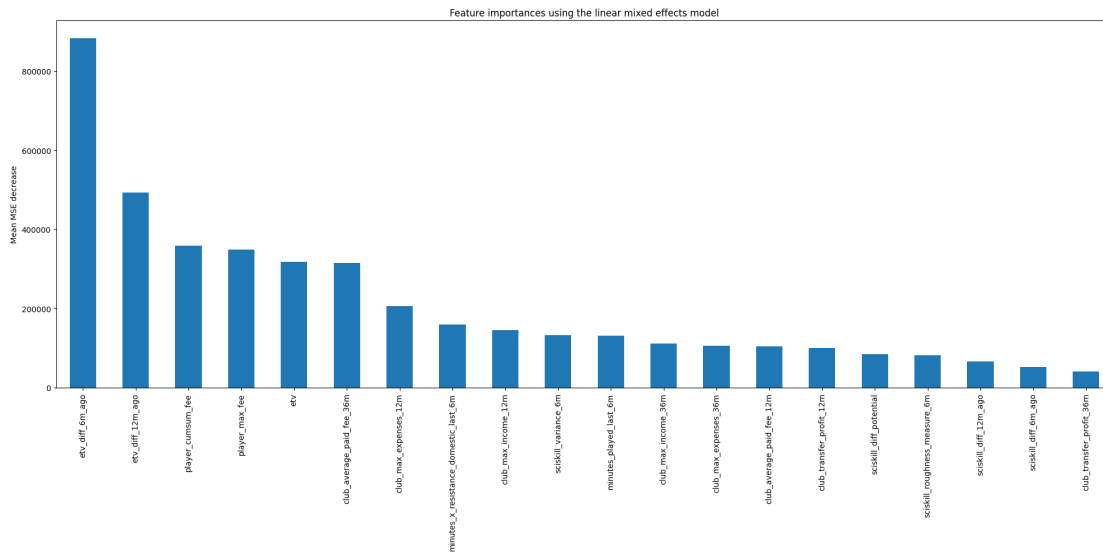


Figure C.26: The feature importance in the RMSE decrease in euros for each feature for the final linear mixed effect model.

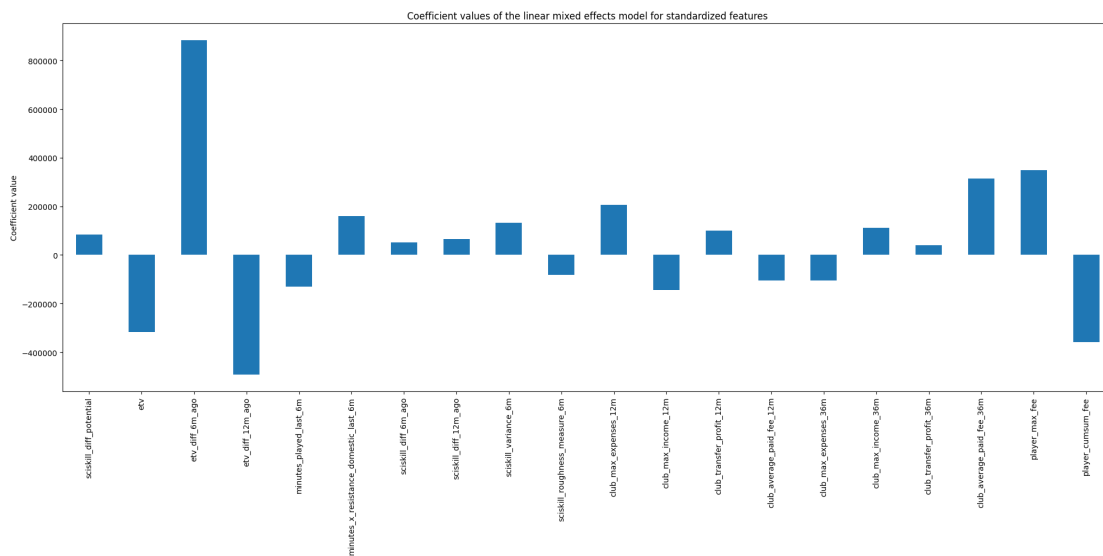


Figure C.27: The coefficients of the final linear mixed effect model of the standardized features in the Estimated Transfer Value case study.

original scale (0 and 1), they need to have large rescaled coefficients to be of any influence. The absence of these binary features, therefore, gives another scale. It can still be seen however, that many changes in the historical values both have both a negative influence ('sciskill\_variance\_6m') and a positive influence ('sciskill\_roughness\_measure\_6m')

on the ETV development. The feature 'sciskill\_variance\_6m' is proportional to the squares of the changes in the velocity of the SciSkill, while 'sciskill\_roughness\_measure\_6m' is proportional to the squares of the in the numerical approximation of the acceleration of the historical values. It can, therefore, be said that many changes in acceleration have a negative influence on the ETV development whereas changes in the velocity of of the SciSkill have a positive influence on the prediction.

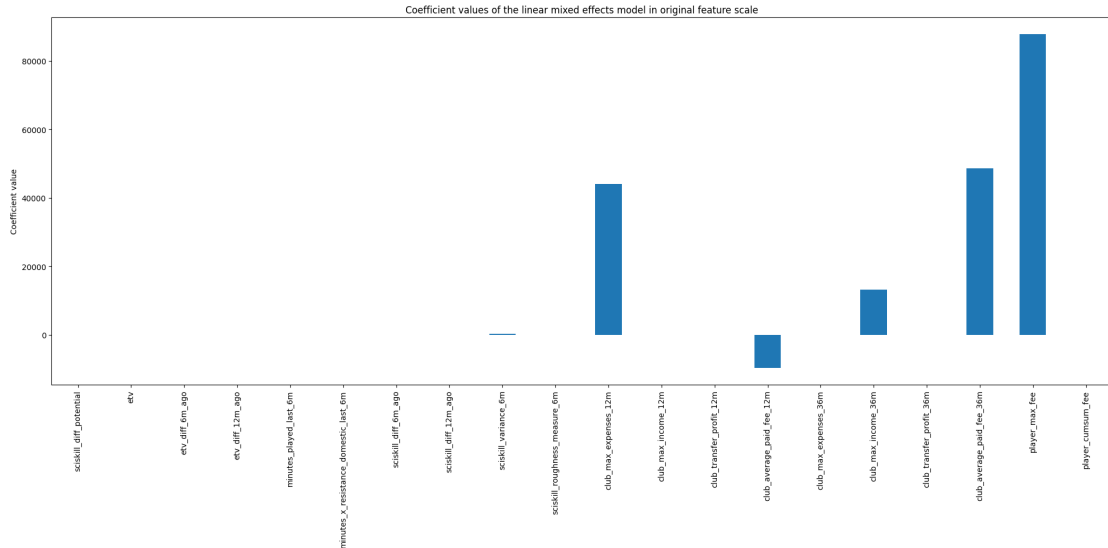


Figure C.28: The coefficients of the final linear mixed effect model in the original scale of the features in the Estimated Transfer Value case study.

The estimated loss values for the linear mixed effects model are given in Table C.2. They show similar values as the OLS and lasso models. Additionally, the differences with both the OLS and the lasso model are less than two times the standard deviation of the training set, which means that the loss values are not considered significantly different.

## C.2.4 | Decision tree

The feature selection for the decision tree was performed by training a decision tree on the predictive task with added noise variables and by selecting the features with a higher feature importance than the noise variables. To this end, a hyperparameter tuning was performed using Bayesian optimization as visualized in Figure D.4. Figure D.3c shows that most gain was obtained in the first five function calls and that later function calls gained no significant improvement. Additionally, the fact that the one-



dimensional marginal plot of the maximal tree depth ('max\_depth') does not select the lowest value in Figure D.17b, indicates that there is some interaction between the maximal tree depth and the minimal number of samples per leaf. This is also reflected in the two-dimensional marginal plot as not all patterns are either horizontal or vertical lines. However, the best-found value is close to the optimal minimal number of samples per leaf in the one-dimensional marginal plot. As this is not the case for the maximal tree depth, it can be concluded that the most influential hyperparameter appears to be the maximal number of samples per leaf.

These hyperparameters were used to train a decision tree model with added noise features. The feature importances of the resulting model are shown in Figure C.29. It can be seen that only a few features were of large importance, which was reflected by the fact that only 28 features were selected out of 57.

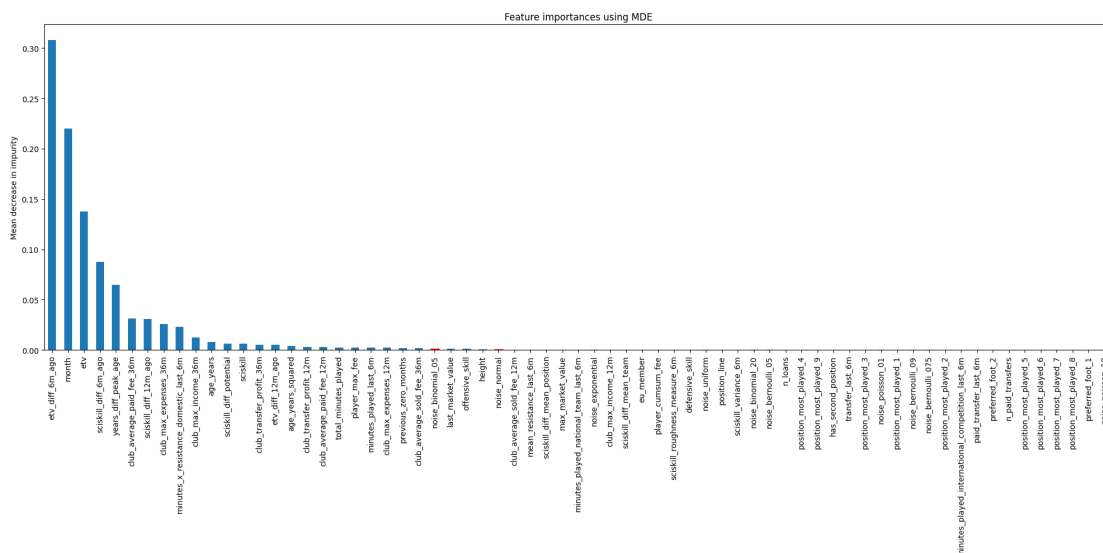


Figure C.29: The node impurity feature importances of the decision tree model for feature selection in the Estimated Transfer Value case study.

A hyperparameter tuning was again applied to the decision tree model on the data set with the selected features as visualized in Figure D.18. It shows similar behavior as the hyperparameter process for the feature selection, especially in the best value for the number of function calls in Figure D.18c. The evaluated points and the estimated objective function in Figure D.18a and Figure D.18b show small differences, but generally the same behavior. It can, therefore, in a similar fashion be concluded that the minimal number per leaf is the most influential hyperparameter.

The final decision tree model was trained using these optimized hyperparameters

and the selected features. This resulted in the feature importances in Figure C.30. The feature importances are similar to those of the decision model for hyperparameter tuning. Opposed to the linear models, the month variable is the second most important feature in the model. The month variable only takes two values as all data points are in January or July, which means that it cannot indicate a nonlinear relation of this feature. The inclusion of the month variable in the decision tree model could, therefore, indicate that the month variable has an interaction term with another variable which is important for the prediction of the development in the Estimated Transfer Value.

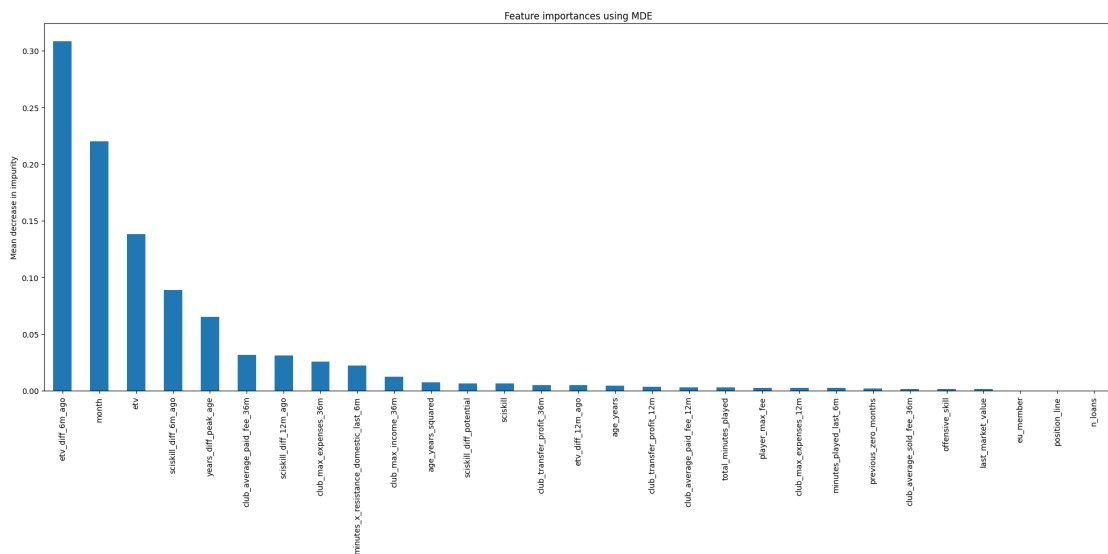


Figure C.30: The node impurity feature importance in euros for each feature for the final decision tree model in the Estimated Transfer Value case study.

The estimated loss values of the decision tree model are given in Table C.2. The loss values of the decision tree appear to be better than those of the linear models. However, the difference with the best RMSE value of the linear models is 12,325, which is less than the standard deviation of the estimates. The decision tree model can, therefore, not be considered to be significantly better performing.

### C.2.5 | Random forest

The feature selection for the random forest was performed by training a random forest regressor to the full data set with added noise features and selecting the best features. To do this, a hyperparameter tuning was first performed. The process of the hyperparameter tuning is visualized in Figure D.19.

Figure D.19c shows that the convergence of the optimization algorithm was significantly slower than that of the decision tree algorithm. This could be expected as there are five hyperparameters instead of two, which increases the dimensionality of the hyperparameter space and slows down the convergence of the optimization algorithm.

The best-found hyperparameters are placed on the or close to boundaries of the hyperparameter space of the dimensions corresponding to the number of decision trees ('n\_estimators'), the maximal tree depth ('max\_depth'), and the minimal number of data points per leaf ('min\_samples\_leaf') as shown in Figure D.19b. The values seem to indicate that more complex models perform better and that the hyperparameter space should be extended at these boundaries. It was chosen not to do this because the random forest model was found to be a computationally expensive algorithm, and making the model more flexible using these parameters would significantly increase the computability. Extending the hyperparameter space was, therefore, not possible for this research.

It can be seen that the estimations of the visualizations of the objective function in Figure D.19b are clearly different from those in Figure D.5b. This can be explained by the increased number of objective function calls, which makes it possible to explore more of the hyperparameter space and improves the quality of the estimates. It can also be due to different behavior in this case study compared to the SciSkill case study.

The feature importances from the random forest model for the feature selection are shown in Figure C.31. The random forest model gives more importance to noise variables compared to the decision tree model. This could be expected as the random forest model randomly includes features in the different decision trees, which makes it possible that noise features are selected randomly. Similarly as with the decision tree, only a few features have a large feature importance. Based on these feature importances, 24 features were selected for the random forest model.

The selected features were used to apply hyperparameter tuning on to find the hyperparameters for the final model. This process is visualized in Figure D.20. The speed of convergence of the model as depicted in Figure D.20c appears to be slightly slower than the hyperparameter tuning for the feature selection random forest model.

Figure D.20a and Figure D.20b show similar behavior of the hyperparameters as in the case of the hyperparameter tuning for the feature selection random forest model. They show that generally, most hyperparameters appear to interact with each other as the patterns in the two-dimensional marginal plots appear to be nonhorizontal or nonvertical. The function evaluations for the number of decision trees ('n\_estimators'), the maximal tree depth ('max\_depth'), and the minimal number of samples per leaf

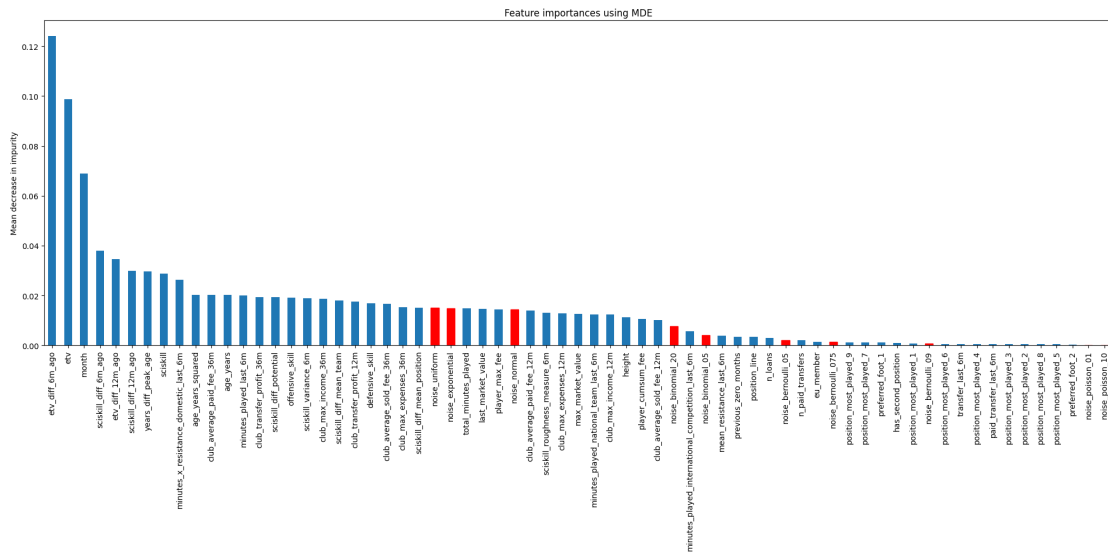


Figure C.31: The node impurity feature importances of the random forest model for feature selection in the Estimated Transfer Value case study. Noise variables are high-lighted in red.

(‘min\_samples\_leaf’) seem to be concentrated around the found optimal values, while this is less clearly visible for the other hyperparameters. This difference indicates that the number of decision trees, the maximal tree depth, and the minimal number of samples per leaf are considered the most important by the Bayesian optimization algorithm.

Similarly as with the random forest model for feature selection, the values of these three hyperparameters indicate that more flexible random forest estimators are preferred by the optimization algorithm. Although it would be interesting to expand the hyperparameter space, this would increase the computational costs, which would make it computationally infeasible. Because of this, we chose not to expand the hyperparameter space in this study.

The best-found value of the fraction of the sampled data points for the training of each individual decision tree (‘max\_samples’) was not on the boundary but around 0.85. This is opposed to the earlier model for feature importance. This makes correlation within the data points a smaller problem for training the model. As the data points of one football player are similar, they can be expected to have collinearity. This problem might be solved by selecting this specific value of this parameter.

The random forest model resulting from the hyperparameter tuning resulted in feature importances as in Figure C.32. It shows that the variables describing the last values of the ETV (‘etv\_diff\_6m\_ago’, ‘etv’, ‘etv\_diff\_12m\_ago’) are important features. The

month of the year also seems to be an important feature. This is similar to the feature importance of the decision tree model, although the features with small importance are relatively more important for the random forest model. This is in line with the theory as the random forest model randomly selects features for each decision tree, which makes less important features having a larger importance.

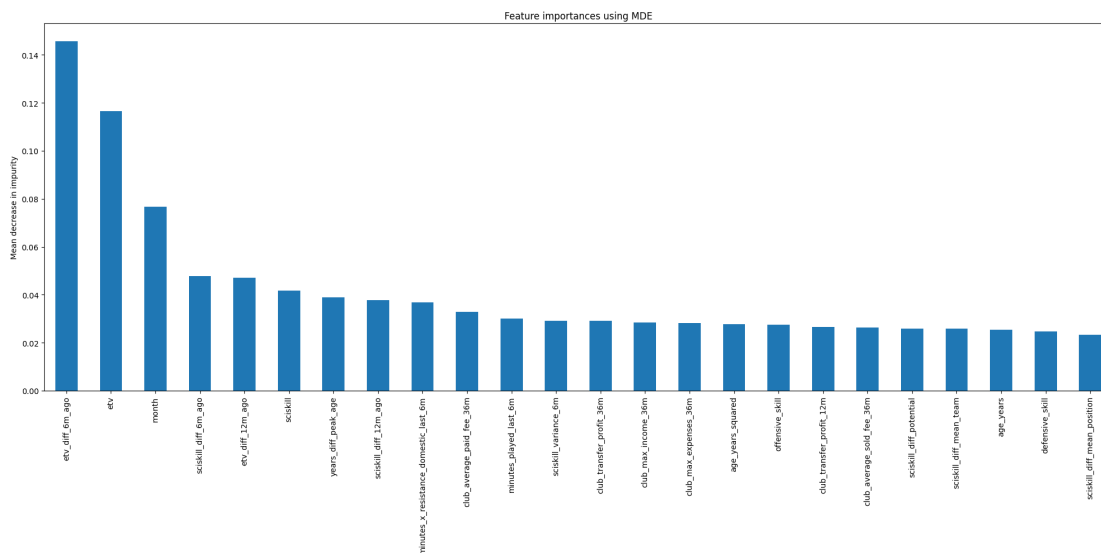


Figure C.32: The node impurity feature importance in euros for each feature for the final random forest model in the Estimated Transfer Value case study.

The estimated loss values of the random forest model are given in Table C.2. The training losses are clearly lower than the test losses, which indicates that the random forest model is overfitting. At the same time, the test estimates of the RMSE, the MAE, and the  $R^2$  of the random forest are better than those of the other models up to now. This means that despite the overfitting, it still performs relatively well for the prediction of unseen data. When the estimated standard deviations for these losses are taken into account, the differences in the MAE and the  $R^2$  are significant. The differences in the RMSE are slightly less than two times the standard deviation, which makes the difference in RMSE nonsignificant.

## C.2.6 | XGBoost

The last tree-based model in this study is the XGBoost model. To apply feature selection, the model was first trained on the full data set with added noise variables. To this end, hyperparameter tuning was first applied in two steps. The first step optimized

seven hyperparameters as visualized in Figure D.7. It can be seen that the improvement in the best value was mostly obtained in the first 10 function calls. At later points, some small improvements were found. As the exact hyperparameters are less important for the feature selection model, the number of function evaluations are considered to be enough.

Figure D.22b shows that the data points are highly concentrated around the value of  $10^{-4}$  for the learning rate  $\eta$ . As this is not the case for the other hyperparameters, it indicates that this hyperparameter is considered to be the most important hyperparameter. For the other hyperparameters except for 'subsample' and 'max\_depth', the influence on the two-dimensional marginal plots with  $\eta$  is minimal, which indicates only a small interaction with this hyperparameter. The learning rate  $\eta$  is thus an important hyperparameter with only limited interaction with the other hyperparameters.

The fraction of the columns sampled per tree level ('colsample\_bylevel') appears to have almost no influence on the predictive quality of the XGBoost model. This is shown by the flat one-dimensional marginal plot and the vertical nature of the two-dimensional marginal plots with the other parameters in Figure D.22b. On the contrary, the columns sampled per tree ('colsample\_bytree') do have an influence on the predictive quality as shown by the marginal plots of this feature. As these two hyperparameters have similar functions, it is not surprising that only one of them has influence on the values chosen by the Bayesian optimization algorithm.

The best maximal depth ('max\_depth') of the XGBoost model was found at a value that was not near the boundaries. This indicates that a more flexible model is most probably not necessary. Another hyperparameter influencing this is the number of decision trees which was tuned in the second hyperparameter tuning as visualized in Figure D.22. The optimal found value in this hyperparameter tuning also had a relatively low value of the number of estimators. This gives that the space of these hyperparameter is most probably sufficient to offer a flexible enough model for the problem.

The resulting feature importances are visualized in Figure C.33. They show that the binomial noise feature ('noise\_binomial\_20') and the uniform noise feature ('noise\_uniform') have relatively high feature importances. Because of this, only 19 features were selected for the XGBoost model.

Based on these selected features, the final XGBoost model was trained. Before the final model was trained, the hyperparameters were optimized in two steps. The first step optimized seven hyperparameters and is visualized in Figure D.23. For all hyperparameters, the one-dimensional marginal plots are more flat as shown in Figure D.24b.

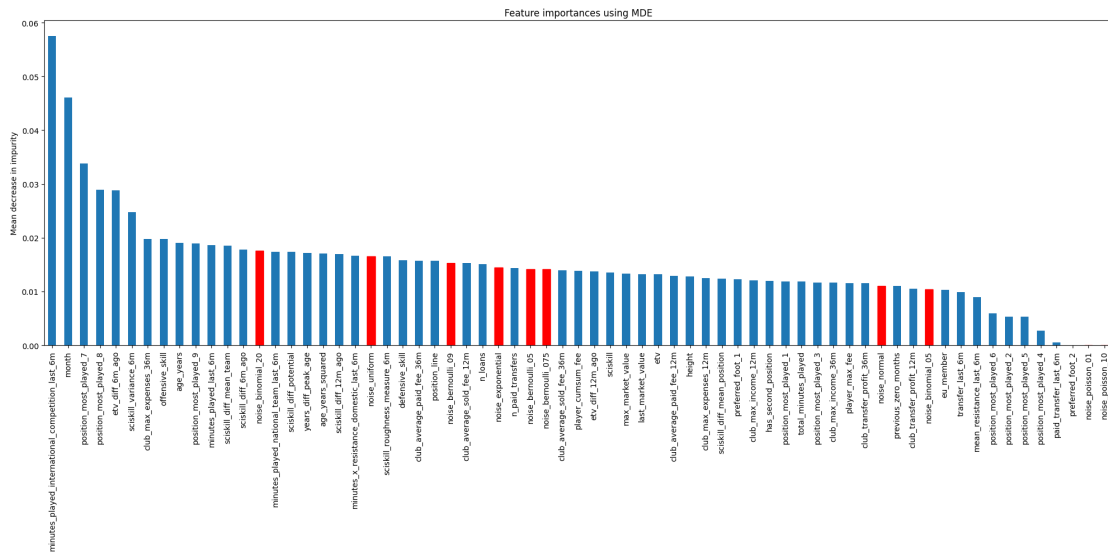


Figure C.33: The node impurity feature importances of the XGBoost model for feature selection in the Estimated Transfer Value case study. Noise variables are highlighted in red.

The predictive performance on this subset of features thus behaves differently compared to the first step in the hyperparameter tuning for the feature selection.

This is most strongly visible in the two-dimensional marginal plots for the learning rate hyperparameter  $\eta$ . Whereas Figure D.7b showed bad performance for small values of  $\eta$ , taking a small value for  $\eta$  does not seem to influence the predictive quality in Figure D.9b. The removal of irrelevant features, therefore, allows for smaller values of the learning rate.

Another difference with the hyperparameter tuning for the feature selection model is that there are more two-dimensional marginal plots of the objective function with elliptic shapes instead of linear patterns. This difference is most obvious for the sample rate of the columns per tree level ('colsample\_bylevel'). This implies the occurrence of different interactions between the parameters.

The results also show that the found value of the column subsampling rate per level of the tree ('colsample\_bylevel') is 1.0, which means that no subsampling occurs. This is opposed to the hyperparameter tuning for the feature selection model, which attained a value of around 0.6. On the other hand, the column subsampling rate per decision tree ('colsample\_bytree') changed from 1.0 for feature selection to around 0.85 for this final model. As both parameters have similar purposes, it can be concluded that the column sampling per tree has taken over the role of the column sampling per tree level.

Figure D.24 visualizes the process of the second hyperparameter tuning which tuned the number of decision trees in the XGBoost model. The behavior of this hyperparameter is similar to the behavior in the hyperparameter tuning for feature selection. In similar fashion, it can be concluded that the the space of this hyperparameter is most probably sufficient to offer a flexible enough model for the problem.

Using these optimized hyperparameters, the final model was trained. The feature importances of this model are given in Figure C.34. It can be seen that the influence of the difference between the current ETV value and that of six months ago ('etv\_diff\_6m\_ago') is not the most important feature and is less important than in the other models so far. Instead, the variable indicating the month is the most important variable, which was an important variable in the decision tree and random forest models too. More surprising is the fact that the number of minutes played by the player in an international competition in the last six months ('minutes\_played\_international\_competition\_last\_6m') is an important feature in this XGBoost model. This was not the case for the other models. It can be concluded that the XGBoost model assigns different importances to the features compared to the other models. Based on this, it could be expected that the behavior of the model is differently.

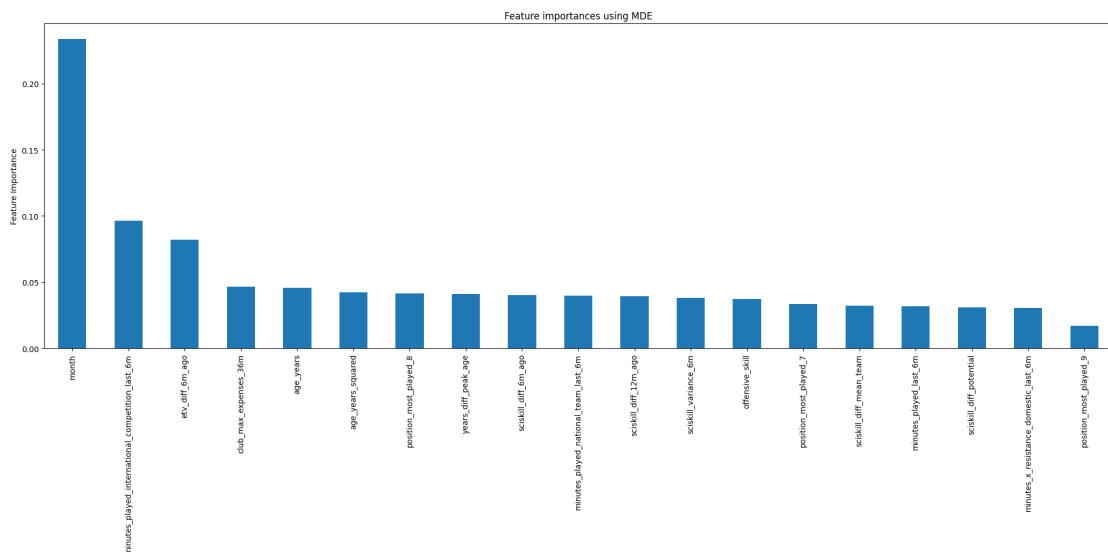


Figure C.34: The node impurity feature importance in euros for each feature for the final XGBoost model in the Estimated Transfer Value case study.

This difference is obvious in the estimated loss functions that are given in Table C.2. The values of the test losses are not very different from the other models. The loss estimates of the XGBoost estimator are worse than that of the random forest model,



but better than the decision tree model and the linear models. It can be seen using two times the standard deviation that the differences in performance in the RMSE are not significant.

### C.2.7 | kNN

Similarly as with the SciSkill case study, three kNN models were trained on the predictive problem. First, a normal kNN model was trained, and to do this, hyperparameter tuning was performed on the number of neighbors  $k$  and the sample weighting method. The resulting process is visualized in Figure D.25. Figure D.25b shows that the performance is bad for low values of  $k$ . For the larger values, the predictive performance appears to become slightly worse with an increase in  $k$ . The distribution of the function evaluations for the neighbor weighting method in Figure D.25a shows that the algorithm clearly favors the distance-based weighting method for the neighbors ('distance'). Because of the low dimension of the hyperparameter space, the algorithm seems to converge fast as visible in Figure D.25c. In this way, the hyperparameter tuning was performed for the kNN model which resulted in selecting  $k = 33$  and the neighbor weight method based on the distance between the neighbors.

Table C.2 gives the estimated loss values for the kNN model. It can be seen that the training losses are smaller than the test losses, which indicates some type of overfitting. Finally, the RMSE and MAE of the method have better values than those of the linear models. Additionally, the value of the RMSE is better than that of the decision tree. However, when taking into account the standard deviation of the RMSE, these differences are not significant.

### C.2.8 | kNN Mahalanobis

The kNN model with Mahalanobis distance was obtained by first applying hyperparameter tuning. This was visualized in Figure D.26. It is surprising that this method does not show a clear preference for the distance-based neighbor weighting method, but also has many function calls for the uniform weights of the neighbors as shown in Figure D.26a. Moreover, the influence of the predictive performance on the  $k$  parameter is less simple as it has a second peak around  $k = 30$  both in the one-dimensional marginal plot and the two-dimensional plot of the objective function in Figure D.26b. It can be concluded that the hyperparameters for the kNN behave differently when the Mahalanobis distance is used.

The one-dimensional marginal plot in Figure D.26b shows a higher loss value for the distance-related neighbor weighting method. The Bayesian optimization algorithm, however, selected this method. This is probably due to the fact that the parameter  $k$  is the most important hyperparameter and that for the best values of  $k$ , the distance-based weighting method performs best. In this way, the interaction between the two hyperparameters influences the selection of the final parameter values.

Table C.2 gives the estimated loss values for the kNN model with Mahalanobis distance. It can be seen that the Mahalanobis distance improves the predictive quality of the kNN model. The RMSE loss value of this model is similar to that of the XGBoost model. Similarly, as with the other loss values, the differences of the RMSE and MAE are not large enough to be significant based on the bootstrap test.

### C.2.9 | kNN RReliefF

The last model was the kNN model with RReliefF feature weighting. Hyperparameter tuning was applied using the Bayesian optimization algorithm and the process is visualized in Figure D.27. The hyperparameters behave similarly to those of the normal kNN model.

One small difference is the fact that the marginal plot of the  $k$  hyperparameter in Figure D.27b has a less steep slope for low values of  $k$ . This means that the kNN model with RReliefF feature weighting needs more neighbors to infer the optimal amount of data. This is also reflected in the fact that the selected value of  $k$  is 50, the maximal value. The predictive performance of the model might have been improved by increasing the maximal number of neighbors.

The RReliefF algorithm was applied to obtain feature weights. These feature weights are shown in Figure C.35. The most important feature according to the RReliefF algorithm is the month variable. This is in accordance with the feature importances of the tree-based methods. Additionally, the current SciSkill ('sciskill') and the SciSkill potential ('sciskill\_potential') are important features. Surprisingly, the features describing the current and historical ETV values ('etv', 'etv\_diff\_6m\_ago', 'etv\_diff\_12m\_ago') only have small importance. This means that the time series of the other variables contains more information about the future ETV development than the ETV itself. This could be expected as the SciSkill and the SciSkill potential are important features in the underlying model of the ETV values.

The estimated loss values of the kNN model with RReliefF feature weights are given in Table C.2. It can be seen that the RReliefF feature weights give better test loss val-

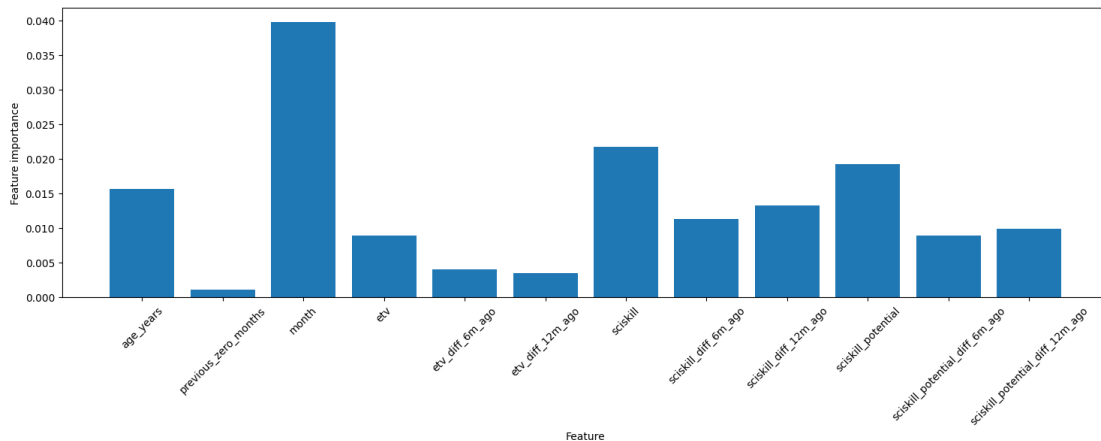


Figure C.35: The feature importance of the final kNN model with RReliefF weights for the ETV case study.

ues compared to the normal kNN. Additionally, the training errors resulting from the RReliefF weighting are worse than for the normal kNN. Thus, overfitting appears to be a smaller problem for the kNN model with RReliefF weighting than with the normal kNN model.

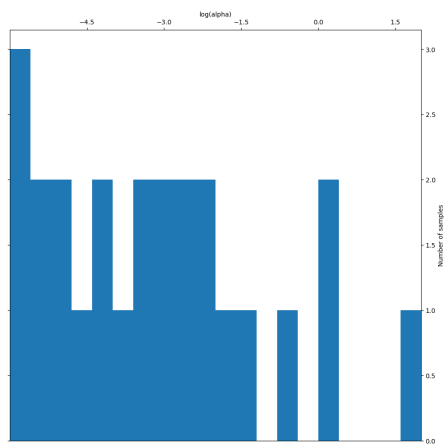
It can be seen that the addition of the Mahalanobis distance provides better predictive performance than the RReliefF feature weighting. This might be due to the fact that there are correlated features that are handled by the Mahalanobis distance.

In general, the kNN model with RReliefF feature weights does not have the best predictive performance compared to the other models, although it has better loss values than the linear models. However, it should be noted that the differences with the other loss values are not significant.

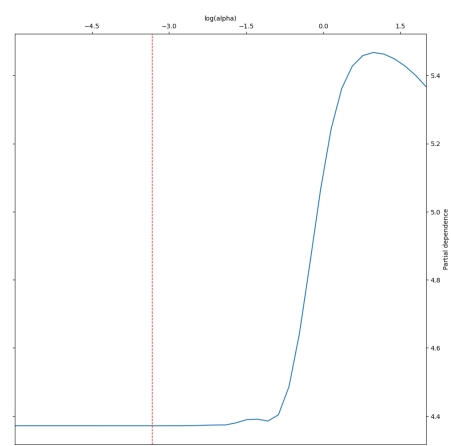


## Visualizations of hyperparameter tuning process

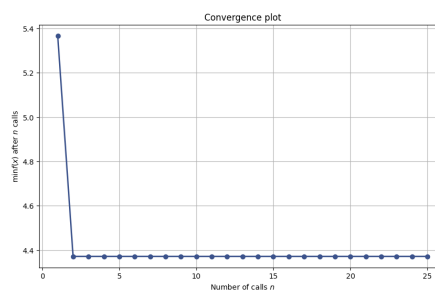
## D.1 | SciSkill case study



(a) The distribution of the function evaluations.

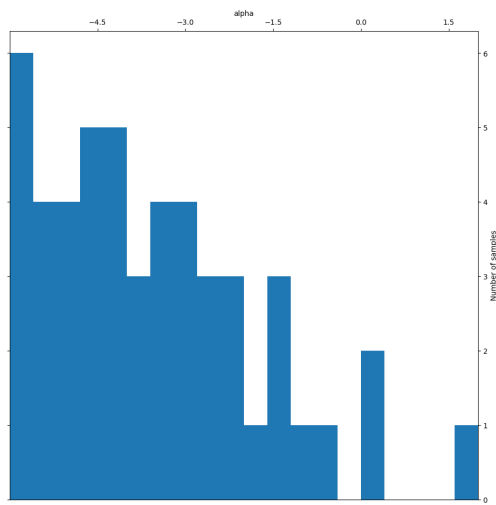


(b) The estimated objective function. The red line indicates the best solution.

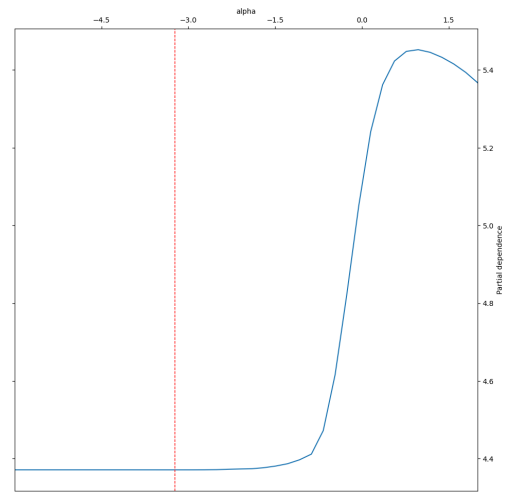


(c) The best value after the number of function calls.

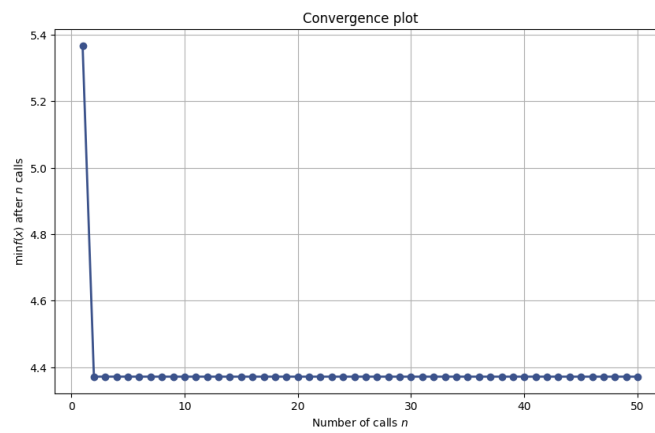
Figure D.1: The visualizations obtained by the hyperparameter tuning for the lasso model for feature selection in the SciSkill case study.



(a) The distribution of the function evaluations.

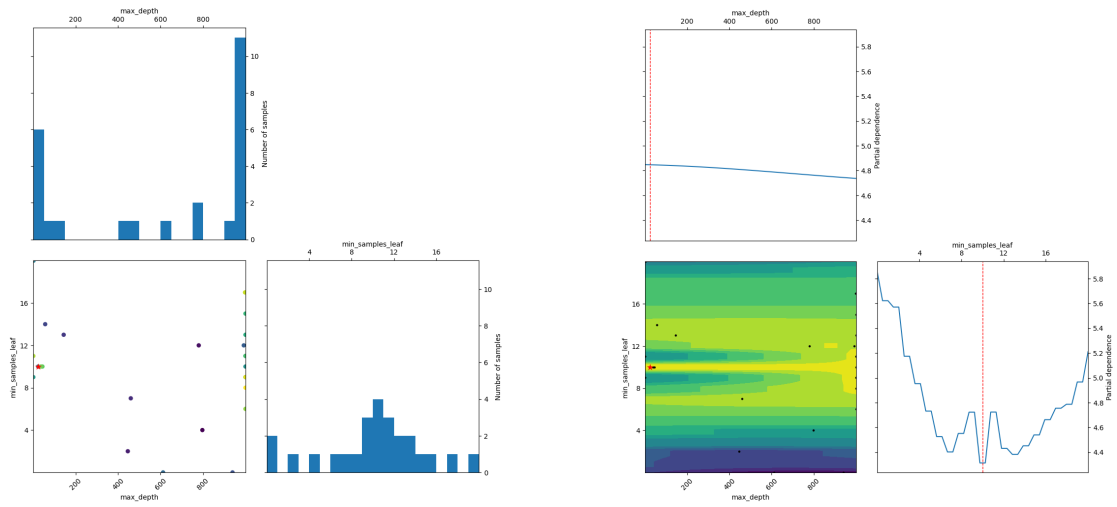


(b) The estimated objective function. The red line indicates the best solution.



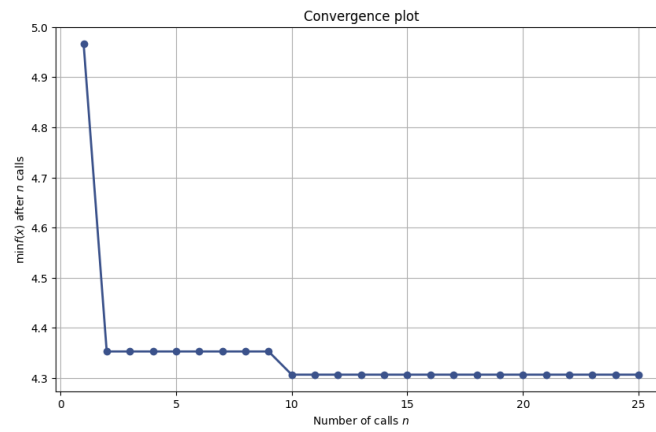
(c) The best value after the number of function calls.

Figure D.2: The visualizations obtained by the hyperparameter tuning for the final lasso model in the SciSkill case study.



(a) The distribution of the function evaluations.

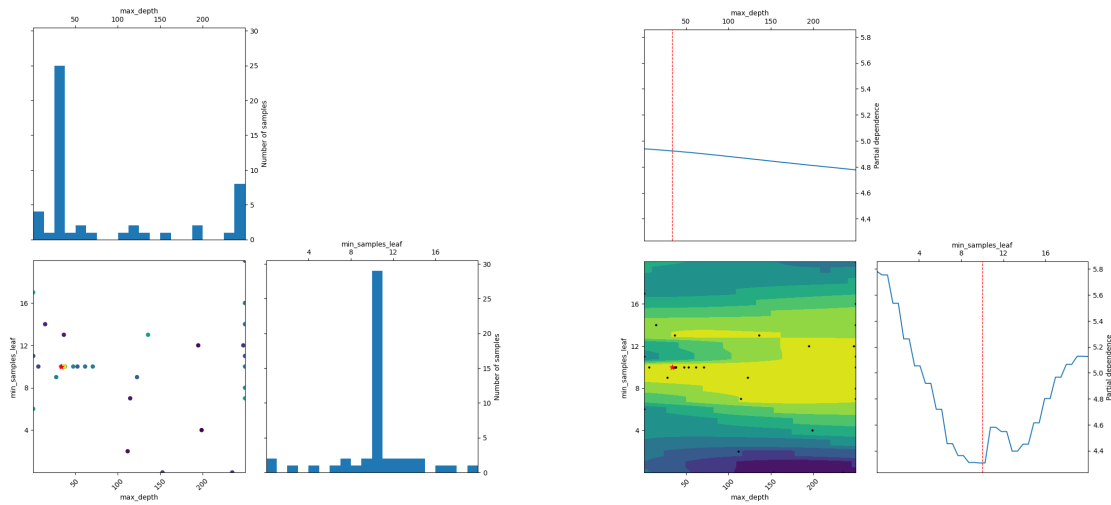
(b) The estimated objective function. The red line and the star (\*) indicate the best solution.



(c) The best value after the number of function calls.

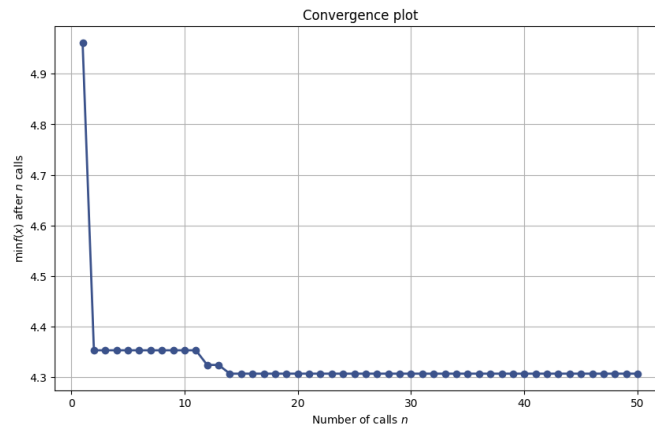
Figure D.3: The visualizations obtained by the hyperparameter tuning for the decision tree model for feature selection in the SciSkill case study.





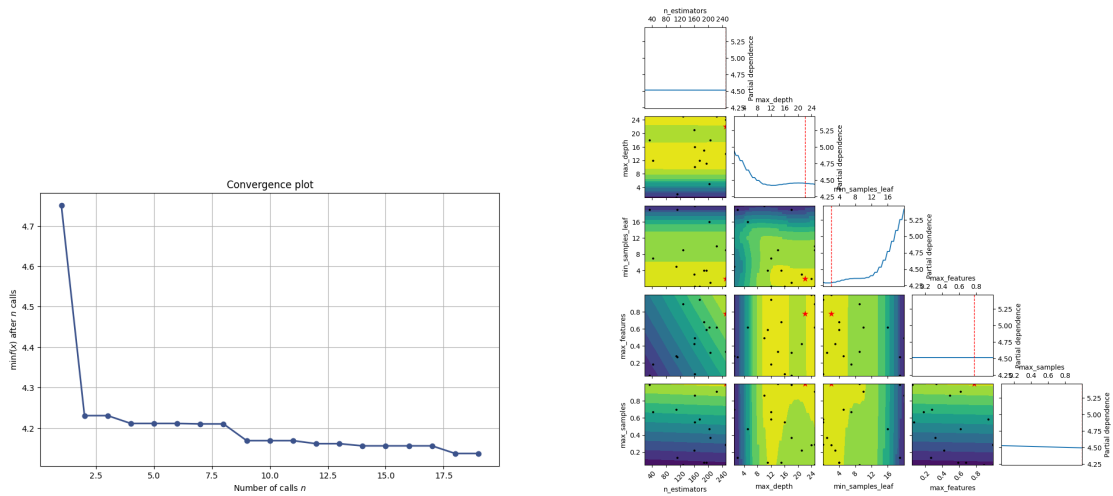
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



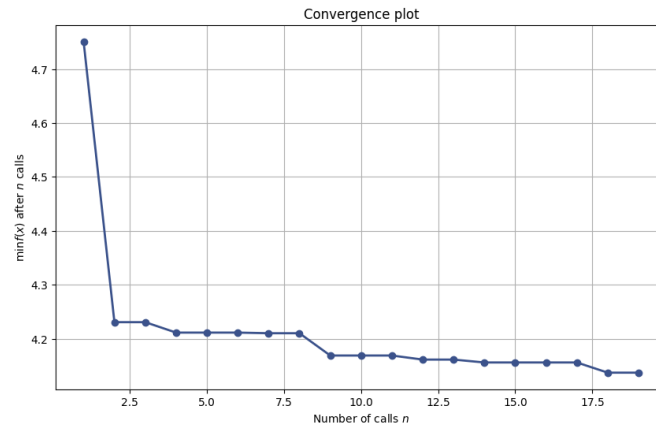
(c) The best value after the number of function calls.

Figure D.4: The visualizations obtained by the hyperparameter tuning for the final decision tree model in the SciSkill case study.



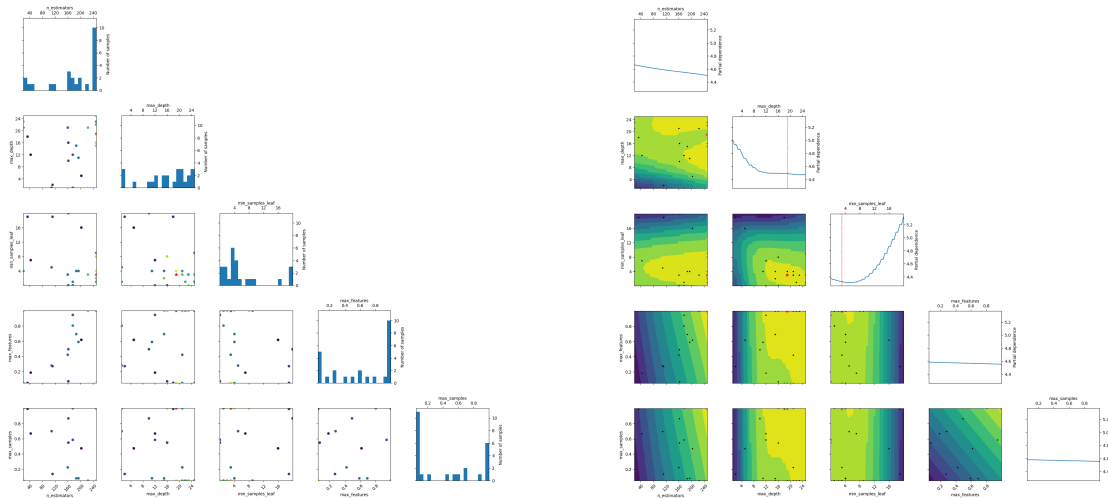
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line and the star (\*) indicate the best solution.



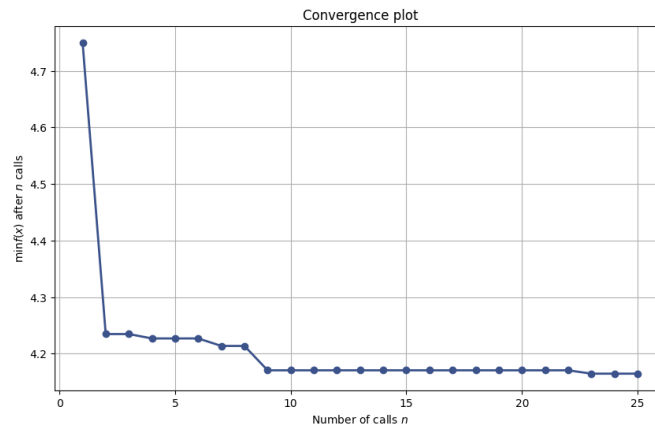
(c) The best value after the number of function calls.

Figure D.5: The visualizations obtained by the hyperparameter tuning for the random forest model for feature selection in the SciSkill case study.



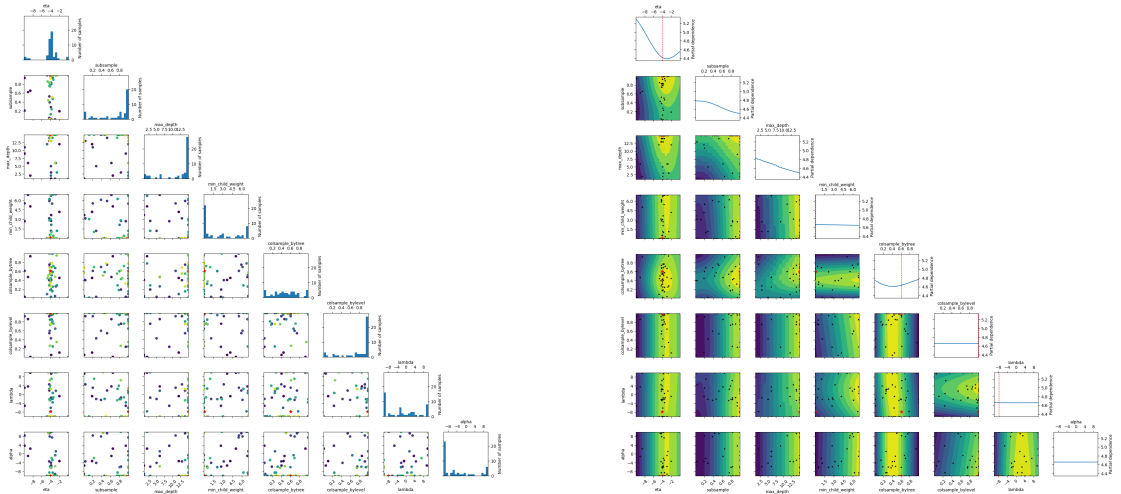
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



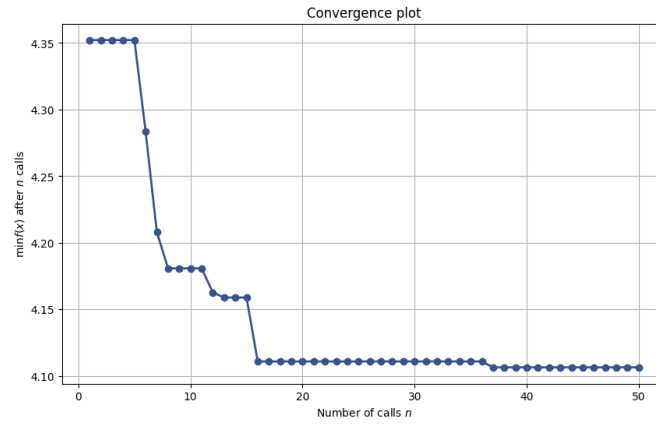
(c) The best value after the number of function calls.

Figure D.6: The visualizations obtained by the hyperparameter tuning for the final random forest model in the SciSkill case study.



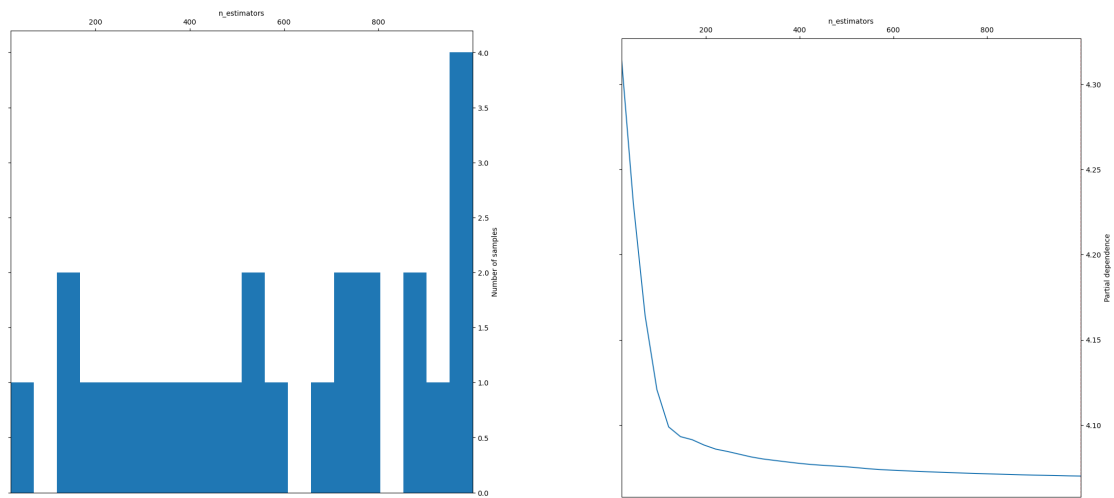
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line and the star (\*) indicate the best solution.



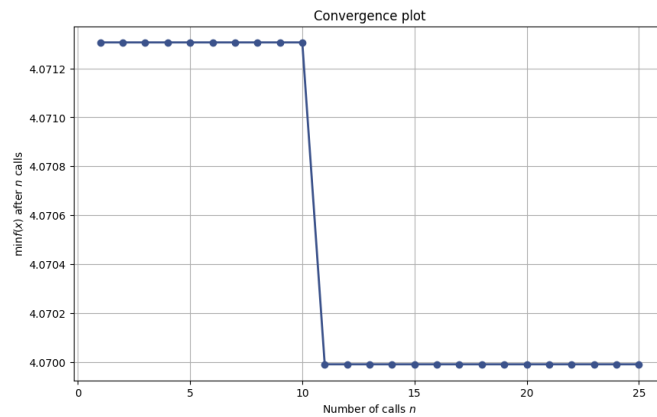
(c) The best value after the number of function calls.

Figure D.7: The visualizations obtained by the first hyperparameter tuning for the XG-Boost model for feature selection in the SciSkill case study.



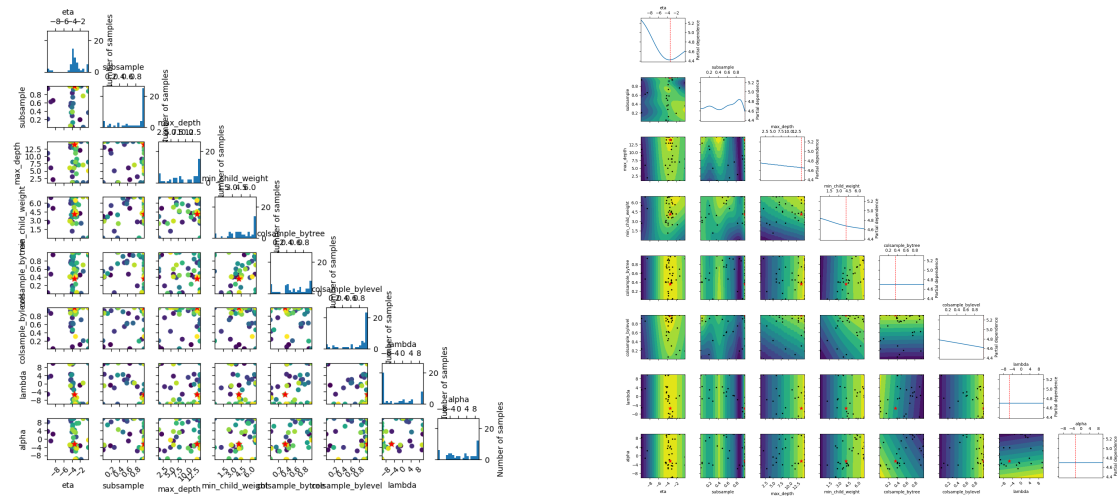
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line and the star (\*) indicate the best solution.



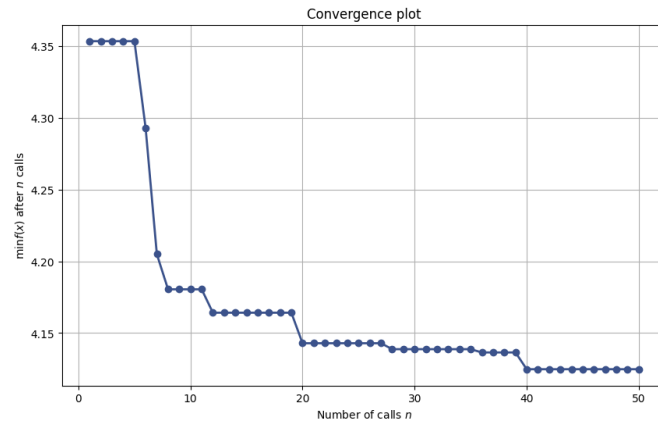
(c) The best value after the number of function calls.

Figure D.8: The visualizations obtained by the second hyperparameter tuning for the XGBoost model for feature selection in the SciSkill case study.



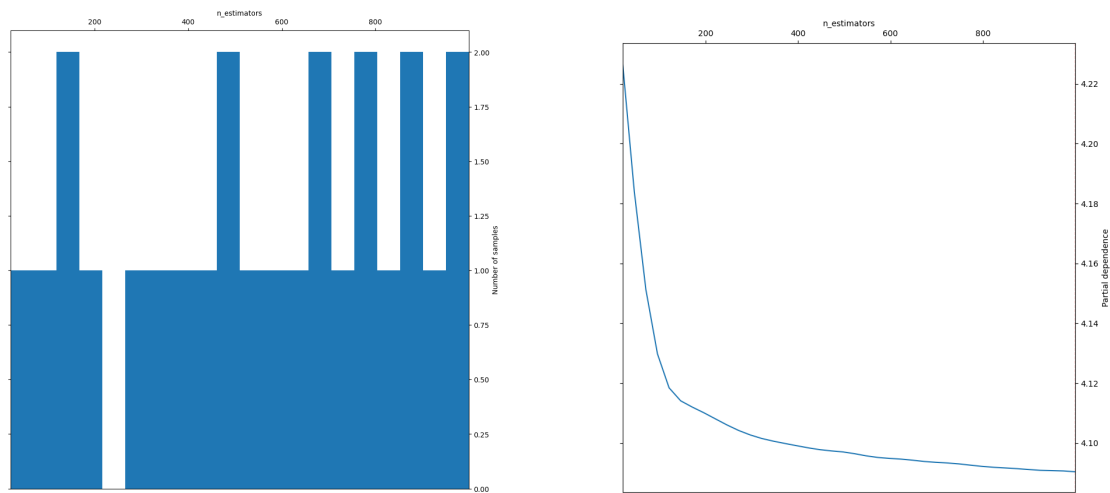
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



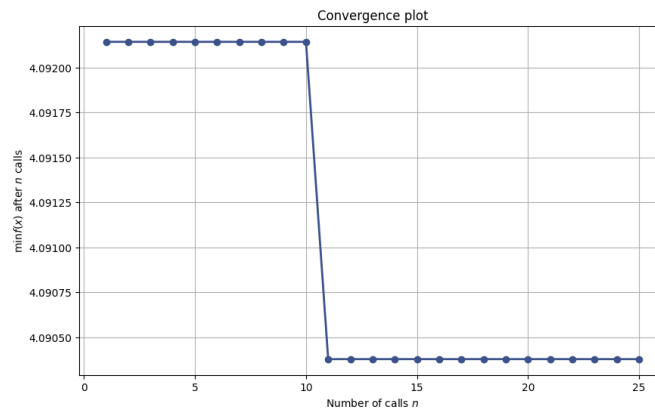
(c) The best value after the number of function calls.

Figure D.9: The visualizations obtained by the first hyperparameter tuning for the final XGBoost model in the SciSkill case study.



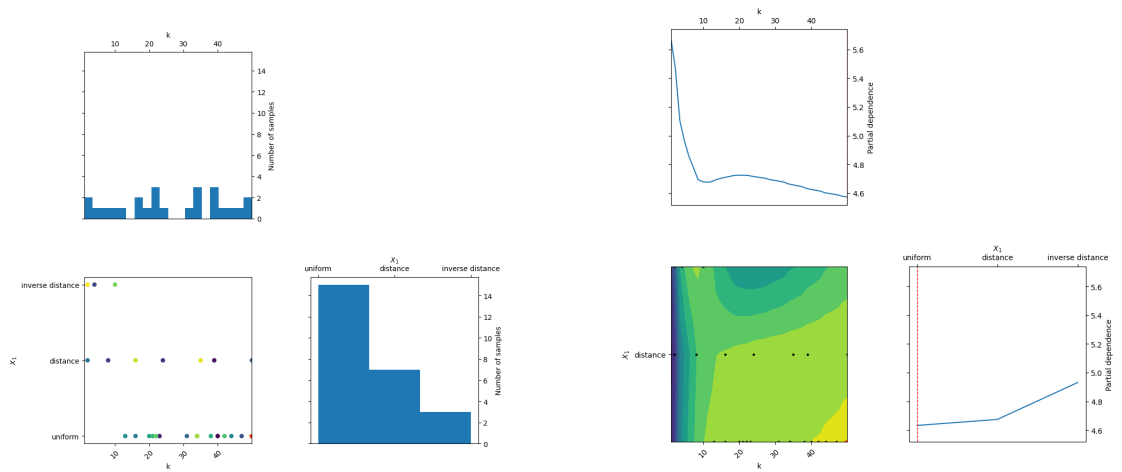
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



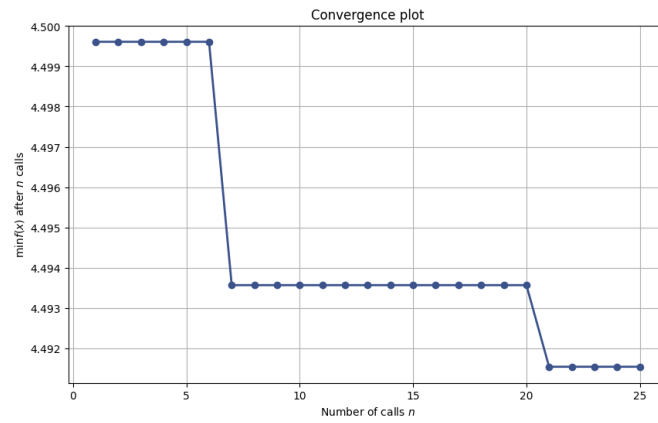
(c) The best value after the number of function calls.

Figure D.10: The visualizations obtained by the second hyperparameter tuning for the final XGBoost model in the SciSkill case study.



(a) The distribution of the function evaluations.

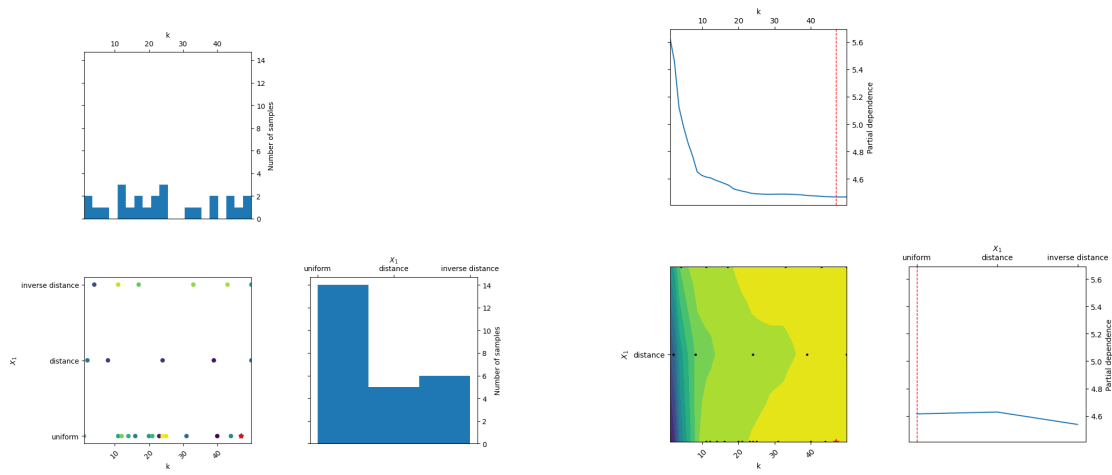
(b) The estimated objective function. The red line indicates the best solution.



(c) The best value after the number of function calls.

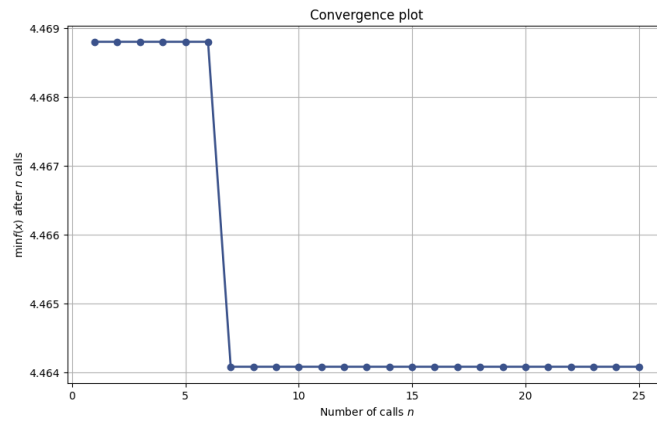
Figure D.11: The visualizations obtained by the hyperparameter tuning for the kNN model in the SciSkill case study.





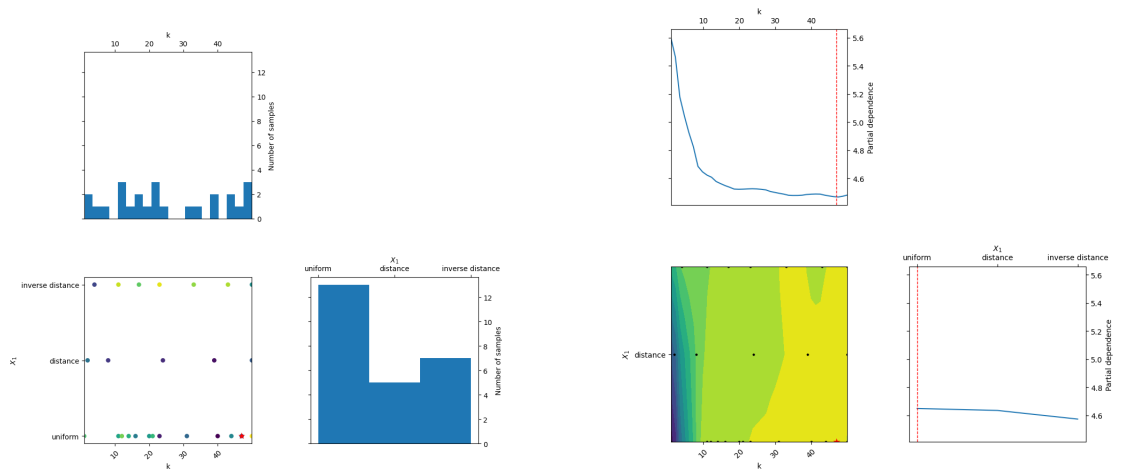
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



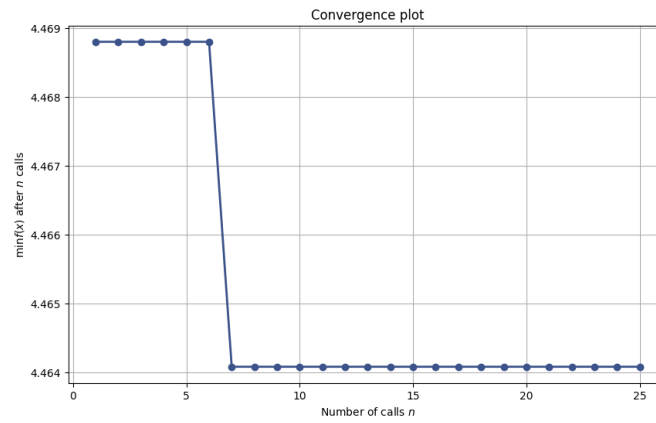
(c) The best value after the number of function calls.

Figure D.12: The visualizations obtained by the hyperparameter tuning for the kNN Mahalanobis model in the SciSkill case study.



(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



(c) The best value after the number of function calls.

Figure D.13: The visualizations obtained by the hyperparameter tuning for the kNN RReliefF model in the SciSkill case study.

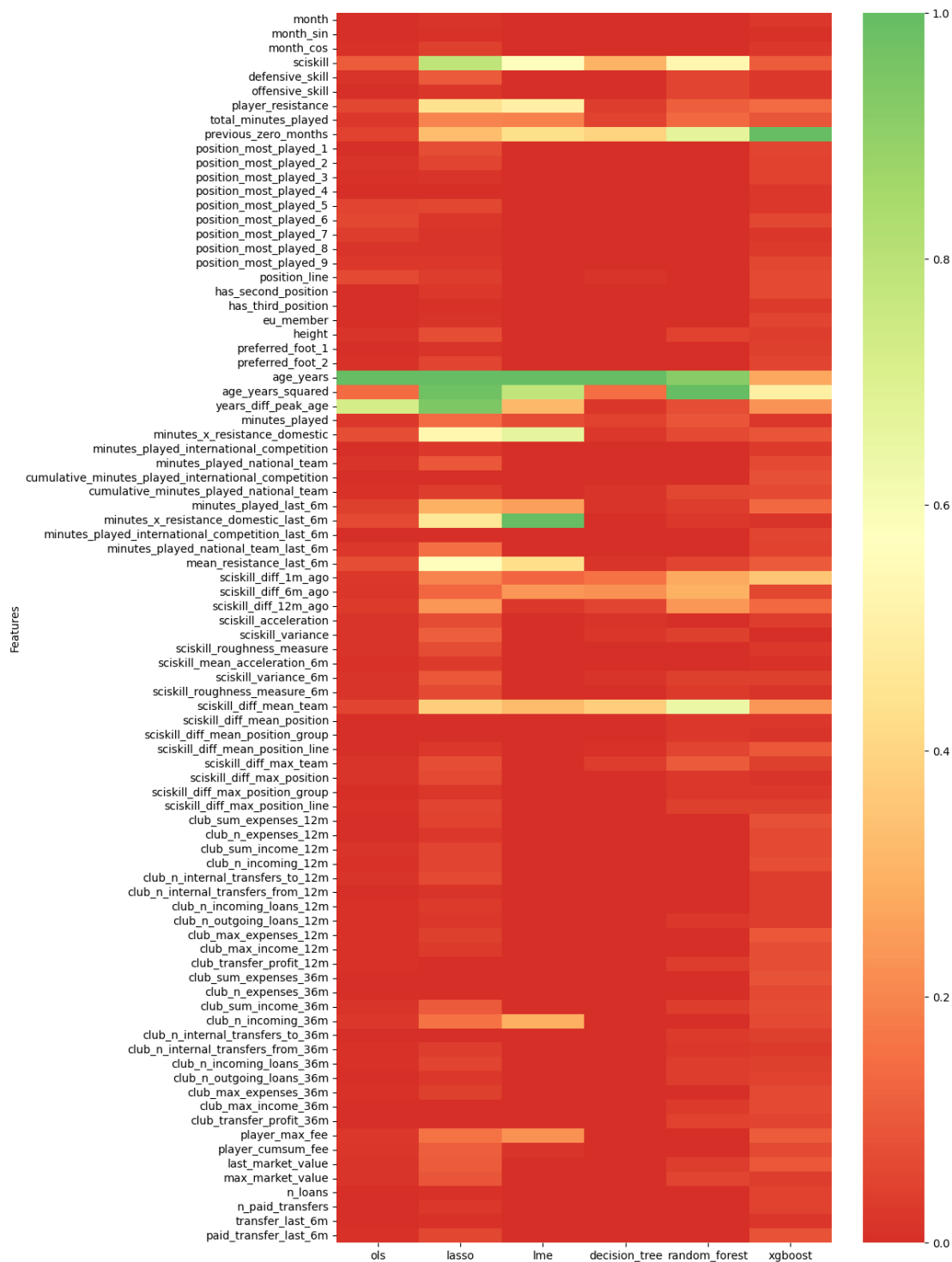
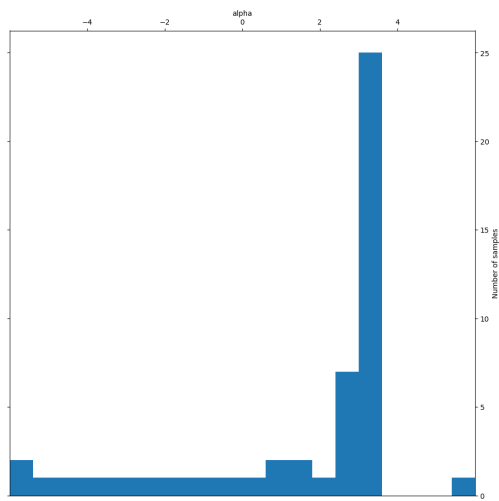
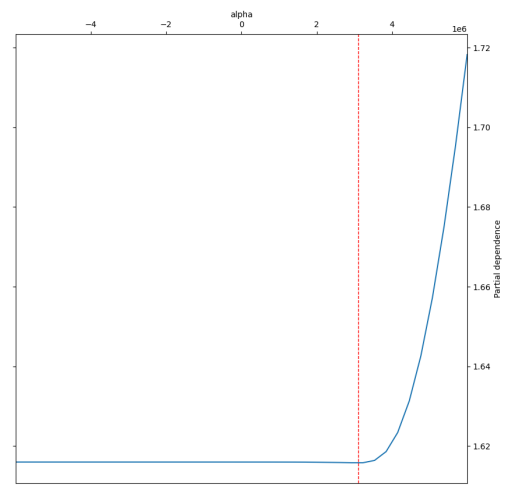


Figure D.14: An oversight of the rescaled (min-max) feature importances of each feature for the different models in the SciSkill case study.

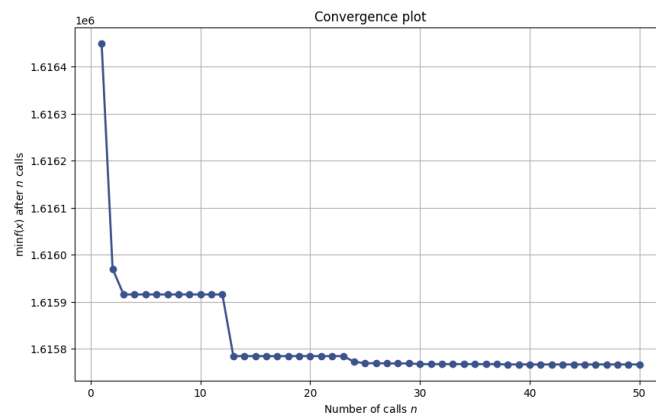
## D.2 | ETV case study



(a) The distribution of the function evaluations.

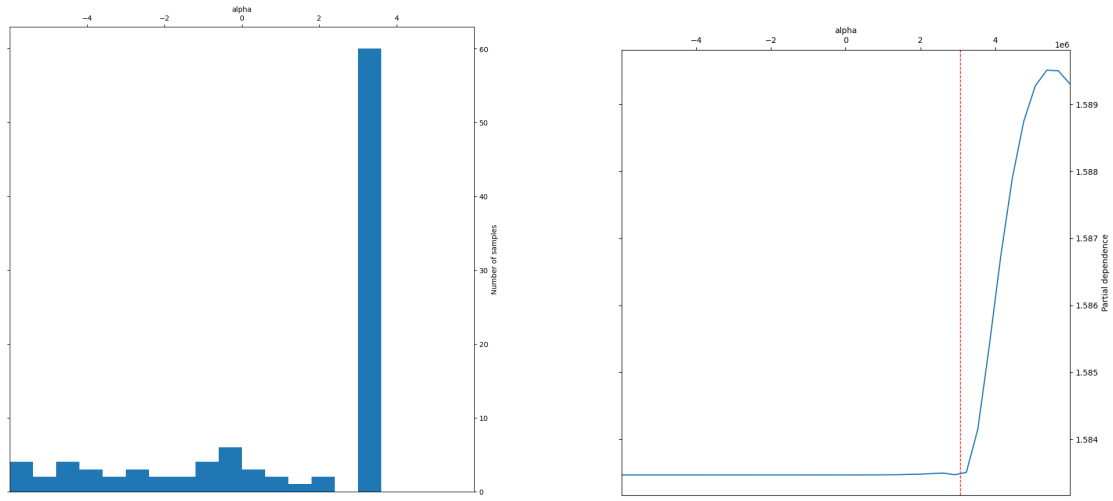


(b) The estimated objective function. The red line indicates the best solution.



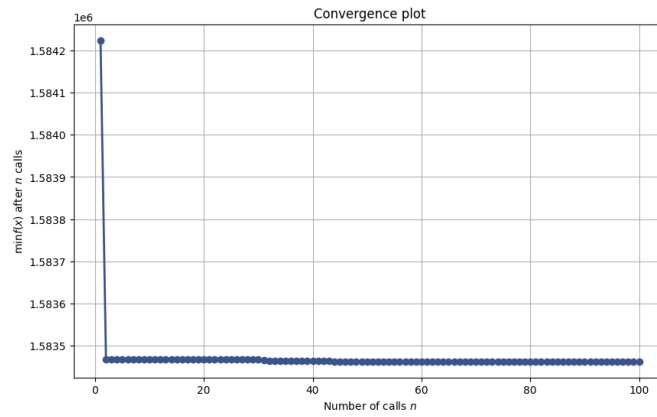
(c) The best value after the number of function calls.

Figure D.15: The visualizations obtained by the hyperparameter tuning for the lasso model for feature selection in the Estimated Transfer Value case study.



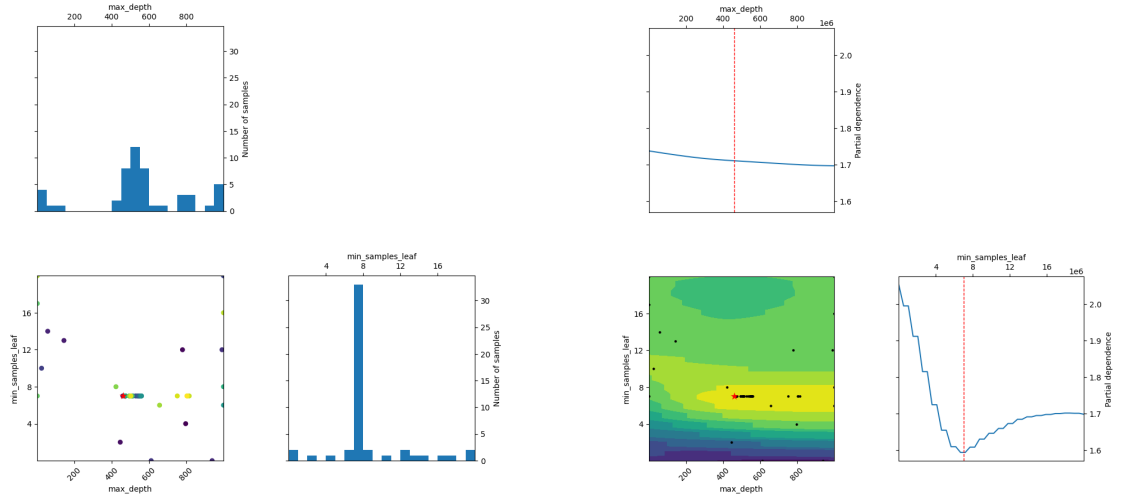
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



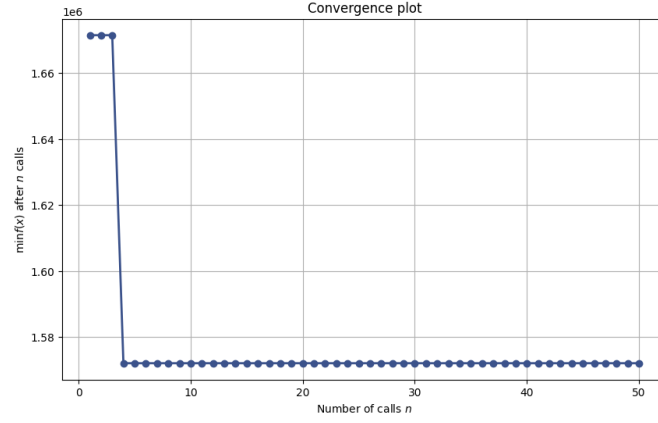
(c) The best value after the number of function calls.

Figure D.16: The visualizations obtained by the hyperparameter tuning for the final lasso model in the Estimated Transfer Value case study.



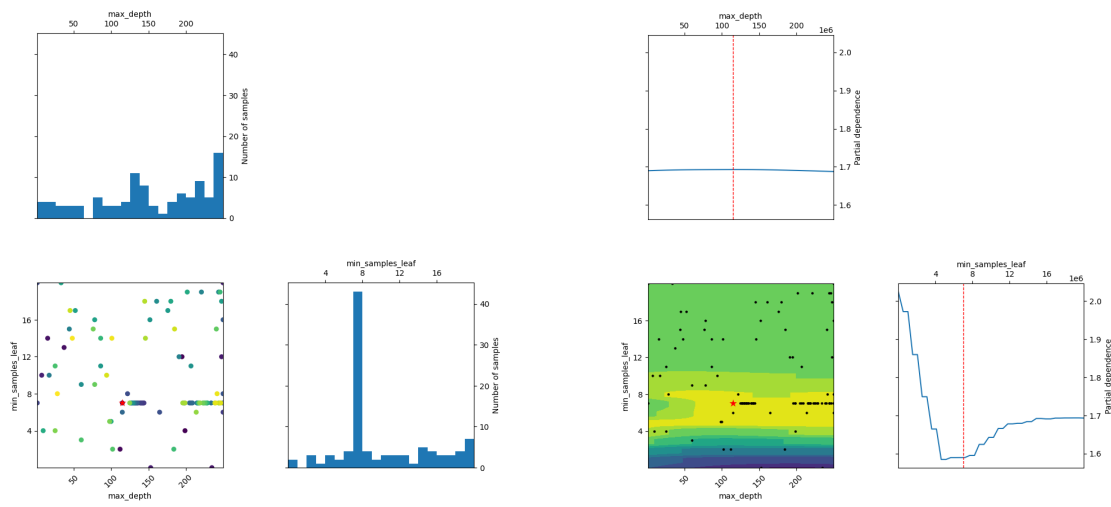
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



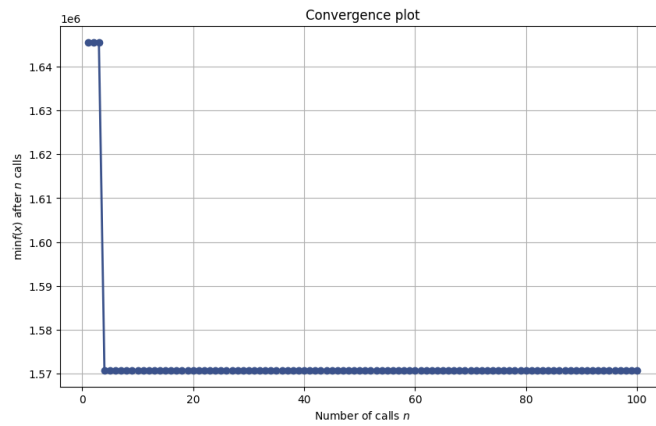
(c) The best value after the number of function calls.

Figure D.17: The visualizations obtained by the hyperparameter tuning for the decision tree model for feature selection in the Estimated Transfer Value case study.



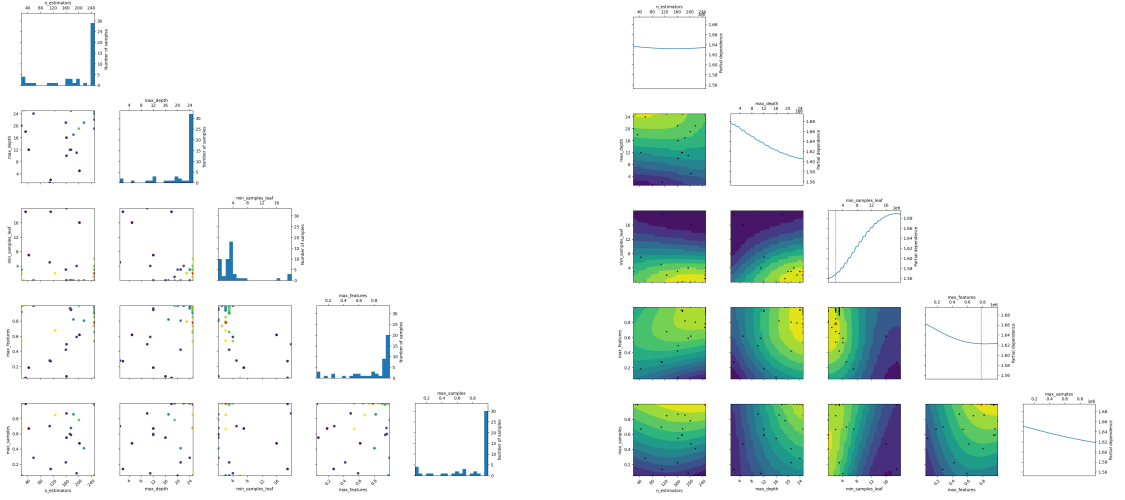
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



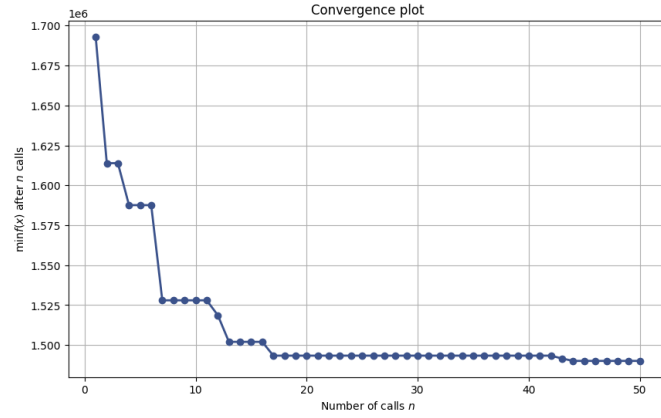
(c) The best value after the number of function calls.

Figure D.18: The visualizations obtained by the hyperparameter tuning for the final decision tree model in the Estimated Transfer Value case study. (The bounds of the hyperparameters are not right and will be adjusted.)



(a) The distribution of the function evaluations.

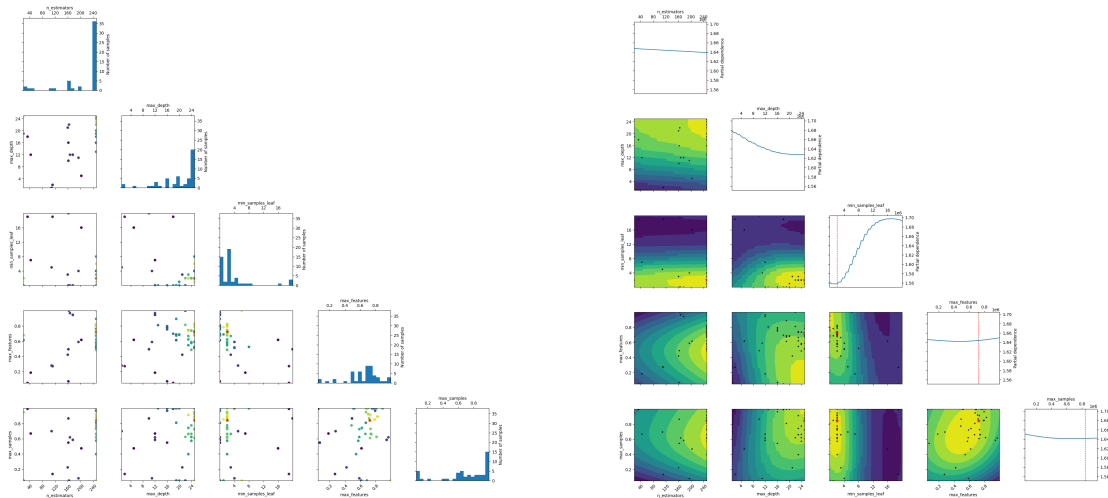
(b) The estimated objective function. The red line indicates the best solution.



(c) The best value after the number of function calls.

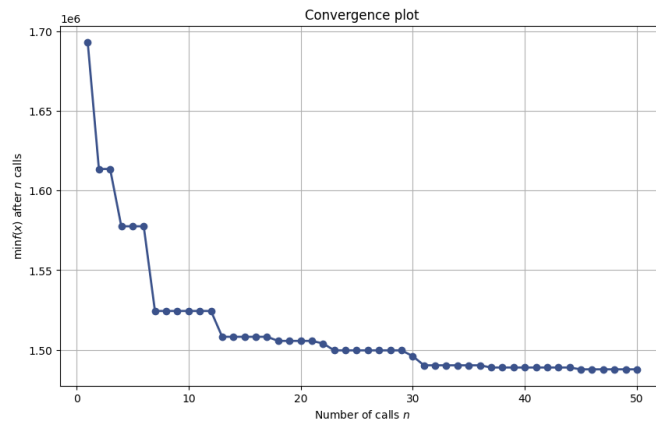
Figure D.19: The visualizations obtained by the hyperparameter tuning for the random forest model for feature selection in the Estimated Transfer Value case study.





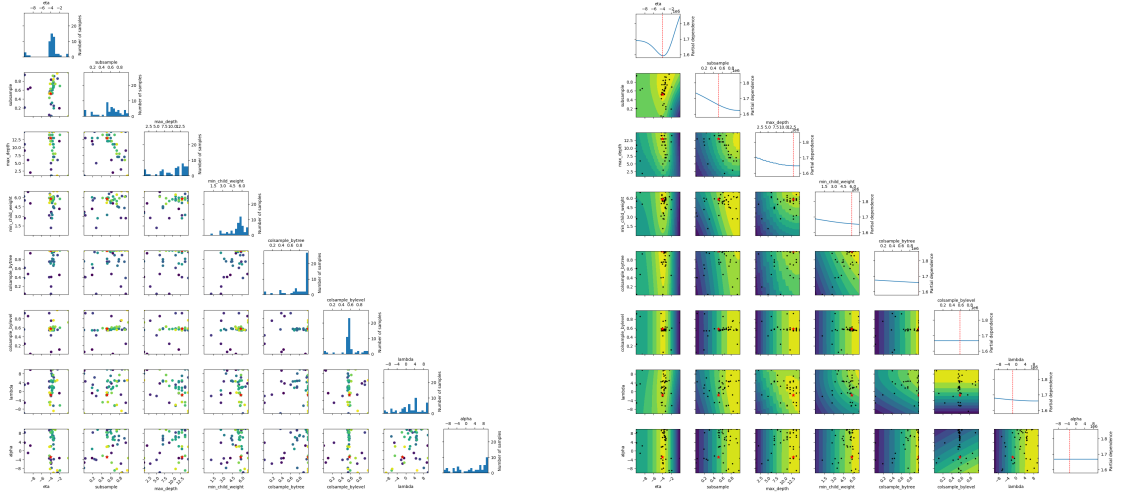
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



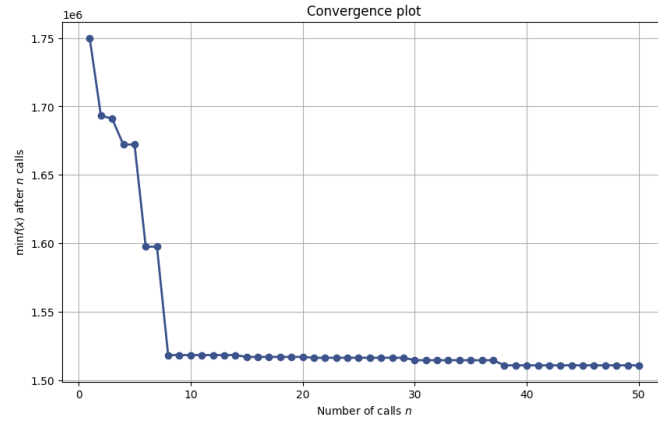
(c) The best value after the number of function calls.

Figure D.20: The visualizations obtained by the hyperparameter tuning for the final random forest model in the Estimated Transfer Value case study.



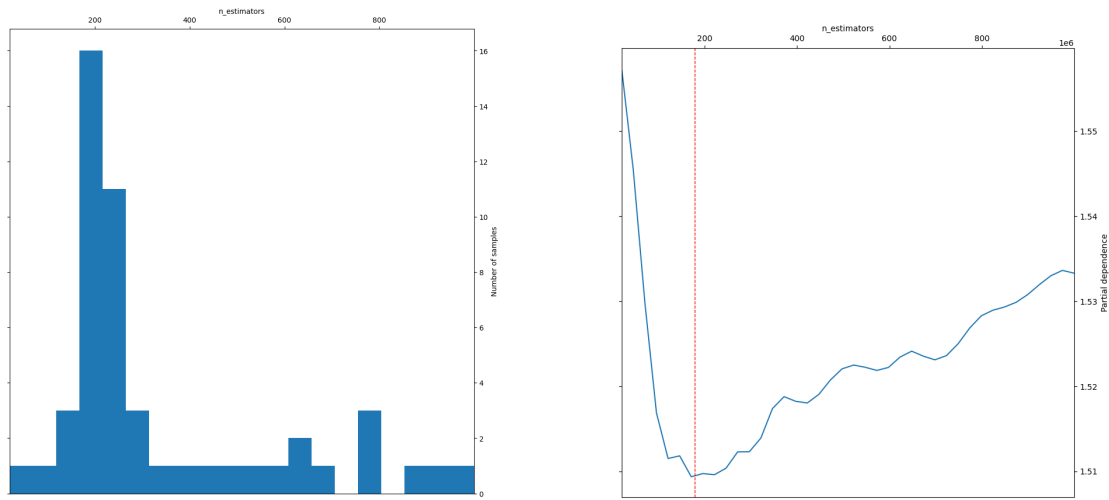
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



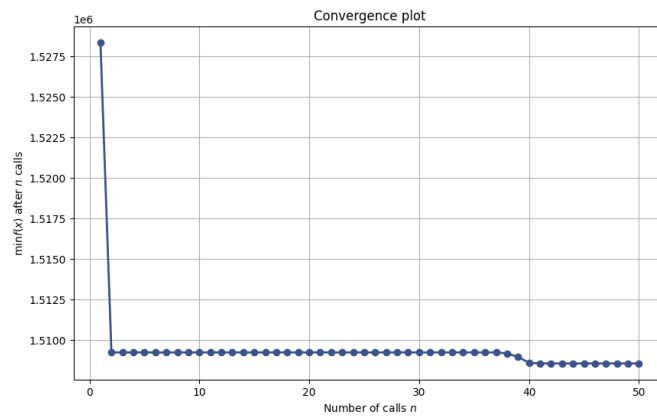
(c) The best value after the number of function calls.

Figure D.21: The visualizations obtained by the first hyperparameter tuning for the XGBoost model for feature selection in the Estimated Transfer Value case study.



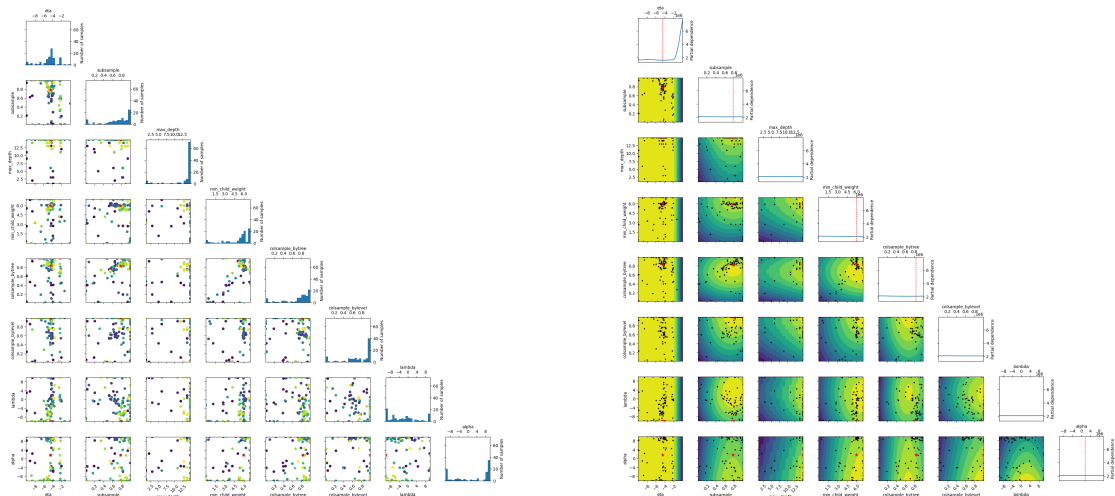
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



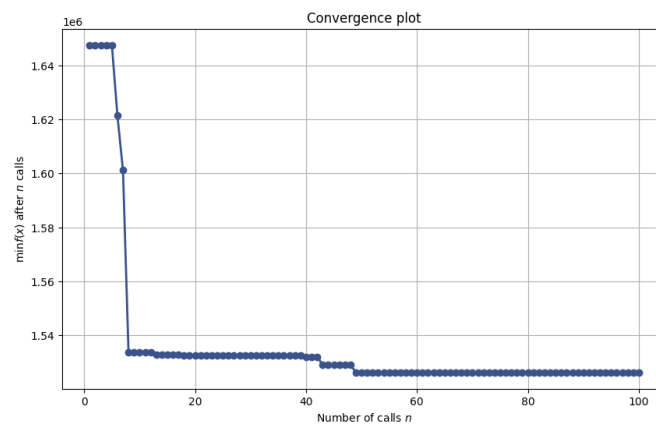
(c) The best value after the number of function calls.

Figure D.22: The visualizations obtained by the second hyperparameter tuning for the XGBoost model for feature selection in the Estimated Transfer Value case study.



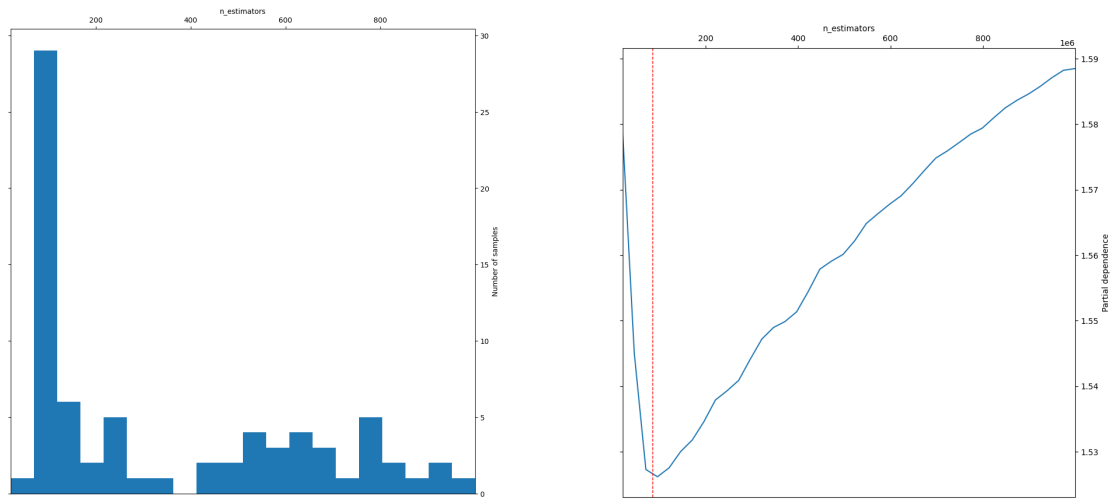
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



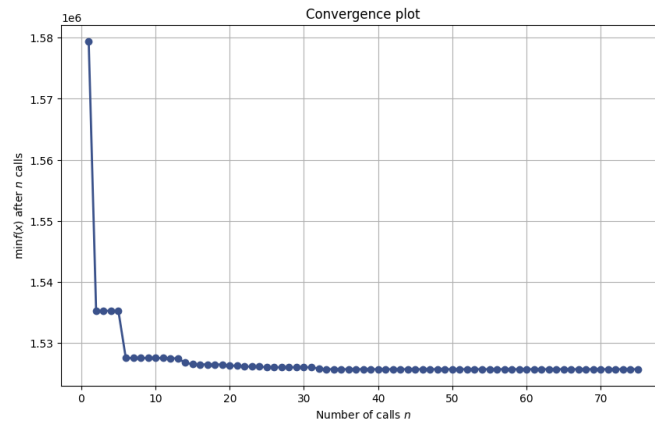
(c) The best value after the number of function calls.

Figure D.23: The visualizations obtained by the first hyperparameter tuning for the final XGBoost model in the Estimated Transfer Value case study.



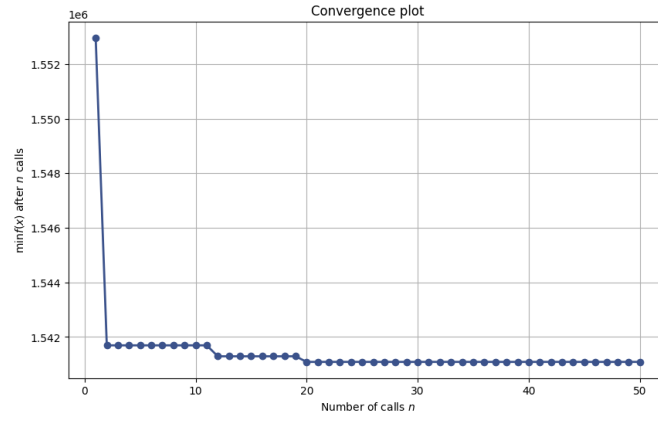
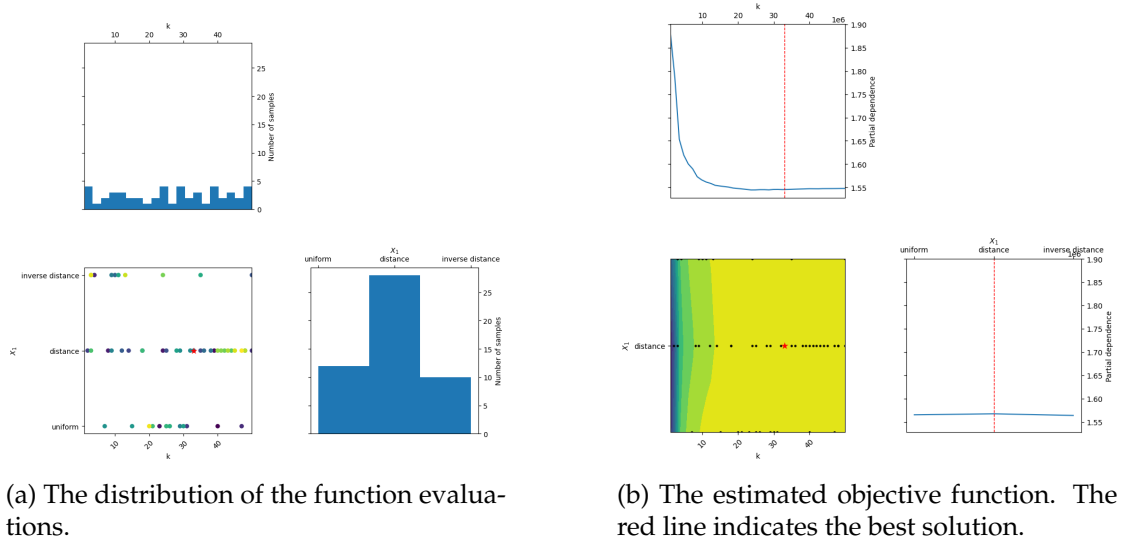
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



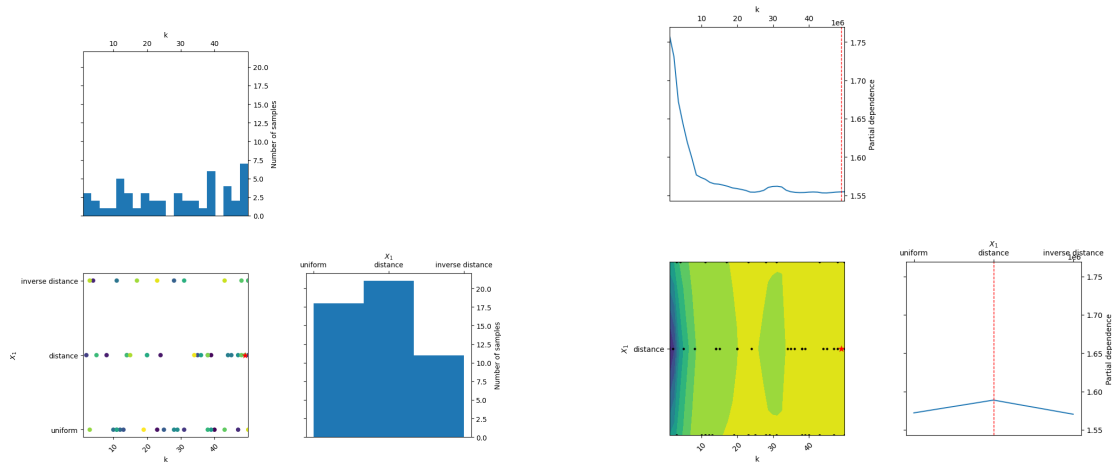
(c) The best value after the number of function calls.

Figure D.24: The visualizations obtained by the second hyperparameter tuning for the final XGBoost model in the Estimated Transfer Value case study.



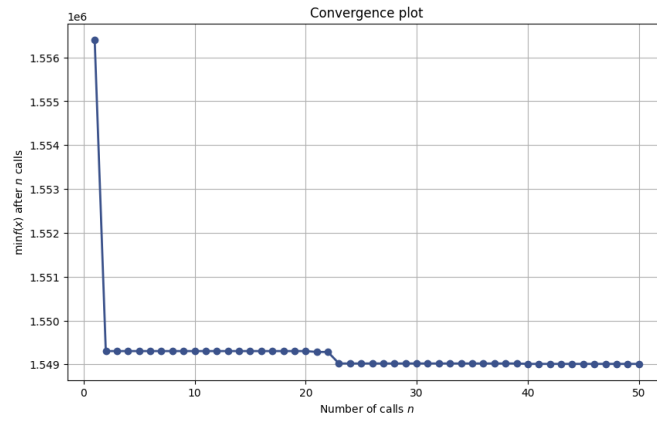
(c) The best value after the number of function calls.

Figure D.25: The visualizations obtained by the hyperparameter tuning for the final kNN model in the Estimated Transfer Value case study.



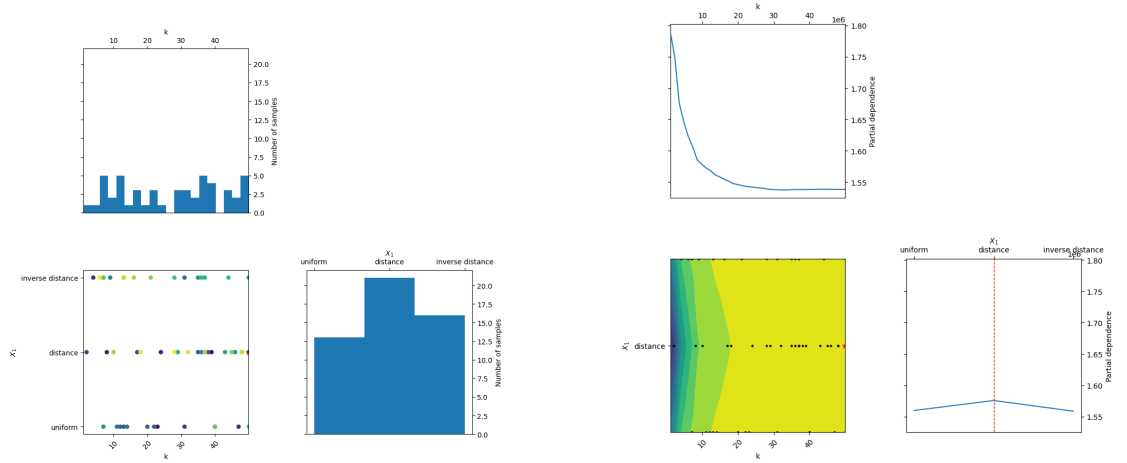
(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



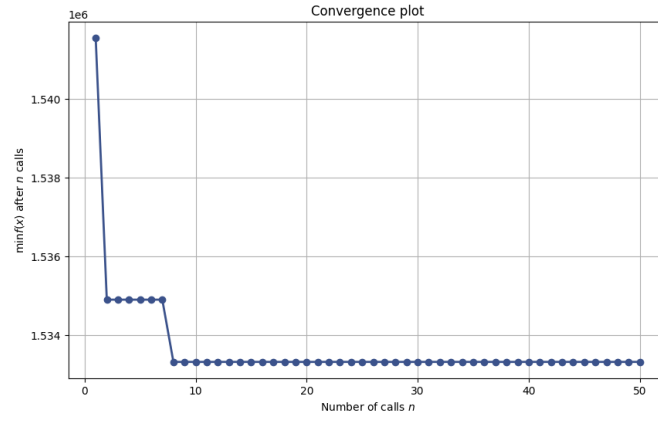
(c) The best value after the number of function calls.

Figure D.26: The visualizations obtained by the hyperparameter tuning for the final kNN model with Mahalanobis distance in the Estimated Transfer Value case study.



(a) The distribution of the function evaluations.

(b) The estimated objective function. The red line indicates the best solution.



(c) The best value after the number of function calls.

Figure D.27: The visualizations obtained by the hyperparameter tuning for the final kNN model with RReliefF weights in the Estimated Transfer Value case study.



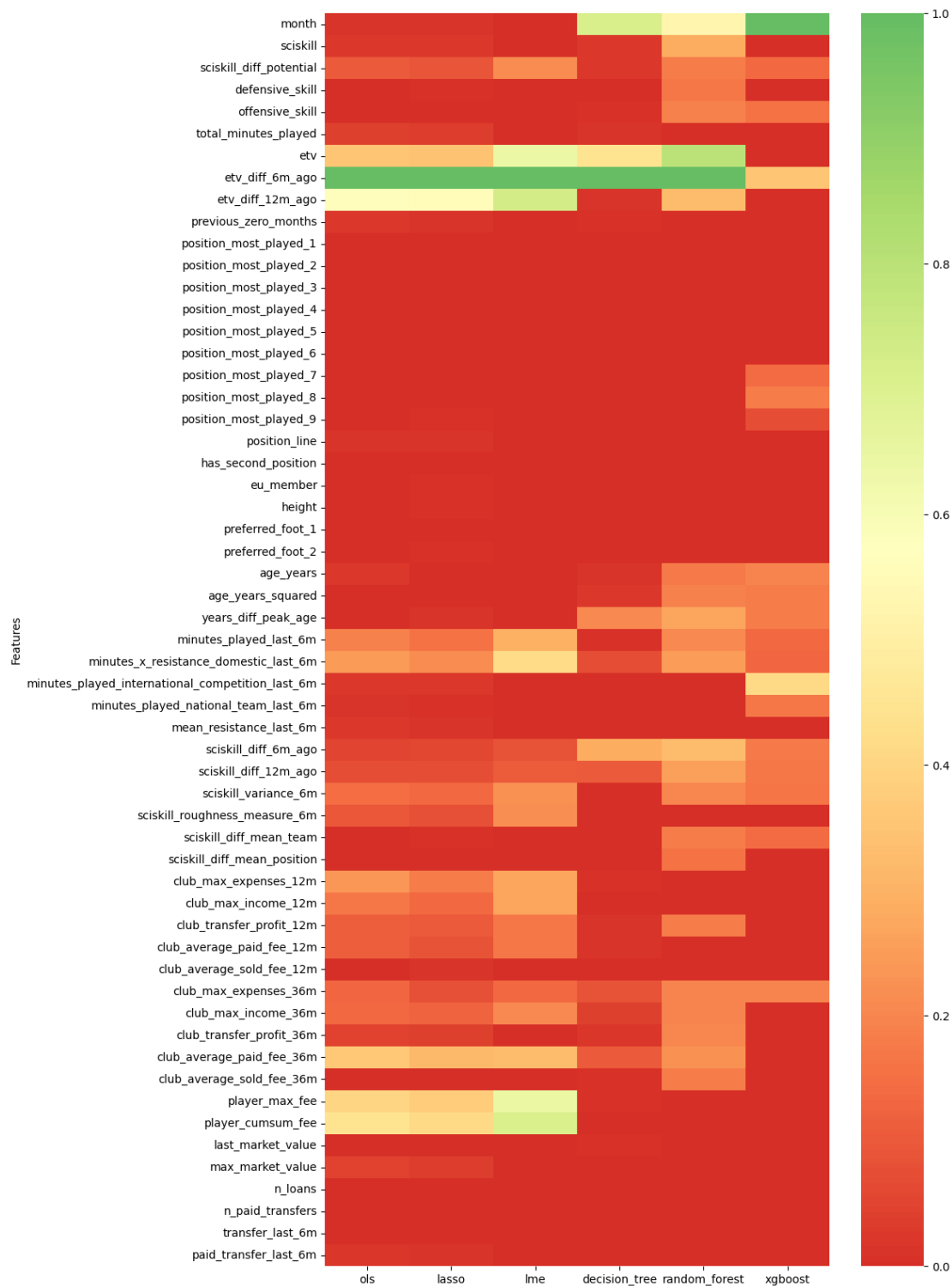


Figure D.28: An oversight of the rescaled (min-max) feature importances of each feature for the different models in the Estimated Transfere Value case study.



## Descriptions of data sets

### E.1 | SciSkill case study

Table E.1: Label description of the SciSkill case study

Label Name	Type	Description
sciskill_diff_12m_future	Float	The difference in SciSkill at current moment and one year later

Table E.2: Feature descriptions of the SciSkill case study

Feature Name	Type	Description
month	Integer	Month of the year (1 to 12)
month_sin	Float	Sine transformation of the month for seasonality
month_cos	Float	Cosine transformation of the month for seasonality
sciskill	Float	SciSkill of the player
defensive_skill	Float	Player's defensive skill level
offensive_skill	Float	Player's offensive skill level
player_resistance	Float	The resistance factor attributed to the player
total_minutes_played	Integer	Total number of minutes played by the player in his career

Continued on next page

Table E.2 – continued from previous page

Feature Name	Type	Description
previous_zero_months	Integer	Number of previous months with zero minutes played
position_most_played_1	Boolean	Whether the player mostly played as left back
position_most_played_2	Boolean	Whether the player mostly played as right back
position_most_played_3	Boolean	Whether the player mostly played as centre back
position_most_played_4	Boolean	Whether the player mostly played as defensive midfielder
position_most_played_5	Boolean	Whether the player mostly played as centre midfielder
position_most_played_6	Boolean	Whether the player mostly played as attacking midfielder
position_most_played_7	Boolean	Whether the player mostly played as left winger
position_most_played_8	Boolean	Whether the player mostly played as right winger
position_most_played_9	Boolean	Whether the player mostly played as centre forward
position_line	Integer	Line of the position the player mostly played (0 for goalkeeper, 1 for defense, 2 for mid-field, 3 for attack)
has_second_position	Boolean	Whether the player has a secondary position
has_third_position	Boolean	Whether the player has a tertiary position
eu_member	Boolean	Whether the player is a member of the European Union
height	Integer	Height of the player in centimeters
preferred_foot_1	Boolean	Whether the player prefers to use left foot
preferred_foot_2	Boolean	Whether the player prefers to use right foot
age_years	Float	Age of the player in years
age_years_squared	Float	Age of the player squared

Continued on next page

Table E.2 – continued from previous page

Feature Name	Type	Description
years_diff_peak_age	Float	Difference in years from the peak age at the player's position
minutes_played	Integer	Minutes played in the last month
minutes_x_resistance_domestic	Float	Product of the minutes played and resistance in domestic competitions in the last month
minutes_played_international_competition	Integer	Minutes played in international competitions in the last month
minutes_played_national_team	Integer	Minutes played for the national team in last month
cumulative_minutes_played_international_competition	Integer	Cumulative minutes played in the international competitions
cumulative_minutes_played_national_team	Integer	Cumulative minutes played for the national team in career
minutes_played_last_6m	Integer	Minutes played in the last 6 months
minutes_x_resistance_domestic_last_6m	Float	Product of minutes played and resistance in domestic competitions over the last 6 months
minutes_played_international_competition_last_6m	Integer	Minutes played in international competitions over the last 6 months
minutes_played_national_team_last_6m	Integer	Minutes played for the national team over the last 6 months
mean_resistance_last_6m	Float	Mean resistance level of the competition of the player over the last 6 months
sciskill_diff_1m_ago	Float	Difference in SciSkill from 1 month ago
sciskill_diff_6m_ago	Float	Difference in SciSkill from 6 months ago
sciskill_diff_12m_ago	Float	Difference in SciSkill from 12 months ago
sciskill_acceleration	Float	Numeric backwards second derivative of the SciSkill
sciskill_variance	Float	Squared difference of SciSkill value of the current and previous month

Continued on next page

Table E.2 – continued from previous page

Feature Name	Type	Description
sciskill_roughness_measure	Float	Squared numeric backwards second derivative of the SciSkill
sciskill_mean_acceleration_6m	Float	Mean numeric backwards second derivative of the SciSkill over the last 6 months
sciskill_variance_6m	Float	Mean squared difference of subsequent SciSkill values over the last 6 months
sciskill_roughness_measure_6m	Float	Mean squared numeric backwards second derivative of the SciSkill over the last 6 months
sciskill_diff_mean_team	Float	Difference in SciSkill from the team average of the last half year
sciskill_diff_mean_position	Float	Difference in SciSkill from the average for the position in his team of the last half year
sciskill_diff_mean_position_group	Float	Difference in SciSkill from the average for the position group in his team of the last half year
sciskill_diff_mean_position_line	Float	Difference in SciSkill from the average for the position line in his team of the last half year
sciskill_diff_max_team	Float	Difference in SciSkill from the team's maximum of the last half year
sciskill_diff_max_position	Float	Difference in SciSkill from the position's maximum in the team of the last half year
sciskill_diff_max_position_group	Float	Difference in SciSkill from the position group's maximum in the team of the last half year
sciskill_diff_max_position_line	Float	Difference in SciSkill from the position line's maximum in the team of the last half year
club_sum_expenses_12m	Integer	Sum of club expenses over the last 12 months
club_n_expenses_12m	Integer	Number of club expenses over the last 12 months

Continued on next page

**Table E.2 – continued from previous page**

<b>Feature Name</b>	<b>Type</b>	<b>Description</b>
club_sum_income_12m	Integer	Sum of club income over the last 12 months
club_n_incoming_12m	Integer	Number of club incoming transfers over the last 12 months
club_n_internal_transfers_to_12m	Integer	Number of internal transfers to the club over the last 12 months
club_n_internal_transfers_from_12m	Integer	Number of internal transfers from the club over the last 12 months
club_n_incoming_loans_12m	Integer	Number of incoming loans to the club over the last 12 months
club_n_outgoing_loans_12m	Integer	Number of outgoing loans from the club over the last 12 months
club_max_expenses_12m	Integer	Maximum single expense by the club over the last 12 months
club_max_income_12m	Integer	Maximum single income for the club over the last 12 months
club_transfer_profit_12m	Integer	Transfer profit of the club over the last 12 months (income minus expenses)
club_sum_expenses_36m	Integer	Sum of club expenses over the last 36 months
club_n_expenses_36m	Integer	Number of club expenses over the last 36 months
club_sum_income_36m	Integer	Sum of club income over the last 36 months
club_n_incoming_36m	Integer	Number of club incoming transfers over the last 36 months
club_n_internal_transfers_to_36m	Integer	Number of internal transfers to the club over the last 36 months
club_n_internal_transfers_from_36m	Integer	Number of internal transfers from the club over the last 36 months
club_n_incoming_loans_36m	Integer	Number of incoming loans to the club over the last 36 months
club_n_outgoing_loans_36m	Integer	Number of outgoing loans from the club over the last 36 months

Continued on next page

Table E.2 – continued from previous page

Feature Name	Type	Description
club_max_expenses_36m	Integer	Maximum single expense by the club over the last 36 months
club_max_income_36m	Integer	Maximum single income for the club over the last 36 months
club_transfer_profit_36m	Integer	Transfer profit of the club over the last 36 months (income minus expenses)
player_max_fee	Integer	Maximum historical transfer fee paid for the player
player_cumsum_fee	Integer	Cumulative historical transfer fees paid for the player
last_market_value	Integer	Last recorded market value of the player
max_market_value	Integer	Maximum historical market value of the player
n_loans	Integer	Number of times the player was loaned
n_paid_transfers	Integer	Number of paid transfers for the player
transfer_last_6m	Boolean	Whether the player was transferred in the last 6 months
paid_transfer_last_6m	Boolean	Whether the player had a paid transfer in the last 6 months

## E.2 | Estimated Transfer Value case study

Table E.3: Label description of the Estimated Transfer Value case study

Label Name	Type	Description
etv_diff_12m_future	Float	The difference in Estimated Transfer Value at current moment and one year later



Table E.4: Feature descriptions of the Estimated Transfer Value case study

Feature Name	Type	Description
month	Integer	Month of the year (1 or 7)
sciskill	Float	Current skill level of the player
sciskill_diff_potential	Float	Difference in SciSkill potential
defensive_skill	Float	Player's defensive skill level
offensive_skill	Float	Player's offensive skill level
total_minutes_played	Integer	Total minutes played by the player in the month
etv	Integer	Current Estimated Transfer Value
etv_diff_6m_ago	Integer	Difference in Estimated Transfer Value from 6 months ago
etv_diff_12m_ago	Integer	Difference in Estimated Transfer Value from 12 months ago
previous_zero_months	Integer	Number of previous months with zero minutes played
position_most_played_1	Boolean	Whether the player mostly played as left back
position_most_played_2	Boolean	Whether the player mostly played as right back
position_most_played_3	Boolean	Whether the player mostly played as centre back
position_most_played_4	Boolean	Whether the player mostly played as defensive midfielder
position_most_played_5	Boolean	Whether the player mostly played as centre midfielder
position_most_played_6	Boolean	Whether the player mostly played as attacking midfielder
position_most_played_7	Boolean	Whether the player mostly played as left winger
position_most_played_8	Boolean	Whether the player mostly played as right winger
position_most_played_9	Boolean	Whether the player mostly played as centre forward

Continued on next page

Table E.4 – continued from previous page

Feature Name	Type	Description
position_line	Integer	Line of the position the player mostly played (0 for goalkeeper, 1 for defense, 2 for mid-field, 3 for attack)
has_second_position	Boolean	Whether the player has a secondary position
eu_member	Boolean	Whether the player is a member of the European Union
height	Integer	Height of the player in centimeters
preferred_foot_1	Boolean	Whether the player prefers to use left foot
preferred_foot_2	Boolean	Whether the player prefers to use right foot
age_years	Float	Age of the player in years
age_years_squared	Float	Age of the player squared
years_diff_peak_age	Float	Difference in years from the peak age at the player's position
minutes_played_last_6m	Integer	Minutes played in the last 6 months
minutes_x_resistance_domestic_last_6m	Float	Product of minutes played and resistance in domestic competitions over the last 6 months
minutes_played_international_competition_last_6m	Integer	Minutes played in international competitions over the last 6 months
minutes_played_national_team_last_6m	Integer	Minutes played for the national team over the last 6 months
mean_resistance_last_6m	Float	Mean resistance level of the competition of the player over the last 6 months
sciskill_diff_6m_ago	Float	Difference in SciSkill from 6 months ago
sciskill_diff_12m_ago	Float	Difference in SciSkill from 12 months ago
sciskill_variance_6m	Float	Mean squared difference of subsequent SciSkill values over the last 6 months
sciskill_roughness_measure_6m	Float	Mean squared numeric backwards second derivative of the SciSkill over the last 6 months

Continued on next page

Table E.4 – continued from previous page

Feature Name	Type	Description
sciskill_diff_mean_team	Float	Difference in SciSkill from the team average of the last half year
sciskill_diff_mean_position	Float	Difference in SciSkill from the average for the position in his team of the last half year
club_max_expenses_12m	Integer	Maximum single expense by the club over the last 12 months
club_max_income_12m	Integer	Maximum single income for the club over the last 12 months
club_transfer_profit_12m	Integer	Transfer profit of the club over the last 12 months (income minus expenses)
club_average_paid_fee_12m	Integer	Average transfer fee paid by the club over the last 12 months
club_average_sold_fee_12m	Integer	Average transfer fee received by the club over the last 12 months
club_max_expenses_36m	Integer	Maximum single expense by the club over the last 36 months
club_max_income_36m	Integer	Maximum single income for the club over the last 36 months
club_transfer_profit_36m	Integer	Transfer profit of the club over the last 36 months (income minus expenses)
club_average_paid_fee_36m	Integer	Average transfer fee paid by the club over the last 36 months
club_average_sold_fee_36m	Integer	Average transfer fee received by the club over the last 36 months
player_max_fee	Integer	Maximum historical transfer fee paid for the player
player_cumsum_fee	Integer	Cumulative historical transfer fees paid for the player
last_market_value	Integer	Last recorded market value of the player
max_market_value	Integer	Maximum historical market value of the player
n_loans	Integer	Number of times the player was loaned

Continued on next page

**Table E.4 – continued from previous page**

<b>Feature Name</b>	<b>Type</b>	<b>Description</b>
n_paid_transfers	Integer	Number of paid transfers for the player
transfer_last_6m	Boolean	Whether the player was transferred in the last 6 months
paid_transfer_last_6m	Boolean	Whether the player had a paid transfer in the last 6 months

---

## References

- Mustafa A. Al-Asadi and Sakir Tasdemir. Predict the Value of Football Players Using FIFA Video Game Data and Machine Learning Techniques. *IEEE Access*, 10:22631–22645, 2022. ISSN 2169-3536. doi: 10.1109/access.2022.3154767. URL <http://dx.doi.org/10.1109/ACCESS.2022.3154767>.
- Konstantinos Apostolou and Christos Tjortjis. Sports Analytics algorithms for performance prediction. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE, July 2019. doi: 10.1109/iisa.2019.8900754. URL <http://dx.doi.org/10.1109/IISA.2019.8900754>.
- Halvard Arntzen and Lars Magnus Hvattum. Predicting match outcomes in association football using team ratings and player ratings. *Statistical Modelling*, 21(5):449–470, July 2020. ISSN 1477-0342. doi: 10.1177/1471082x20929881. URL <http://dx.doi.org/10.1177/1471082x20929881>.
- Ayşe Elvan Aydemir, Tugba Taskaya Temizel, Alptekin Temizel, Kliment Preshlenov, and Daniel M. Strahinov. A Dimension Reduction Approach to Player Rankings in European Football. *IEEE Access*, 9: 119503–119519, 2021. ISSN 2169-3536. doi: 10.1109/access.2021.3107585. URL <http://dx.doi.org/10.1109/ACCESS.2021.3107585>.
- Ayşe Elvan Aydemir, Tugba Taskaya Temizel, and Alptekin Temizel. A Machine Learning Ensembling Approach to Predicting Transfer Values. *SN Computer Science*, 3(3), March 2022. ISSN 2661-8907. doi: 10.1007/s42979-022-01095-z. URL <http://dx.doi.org/10.1007/s42979-022-01095-z>.
- Ali Baouan, Elsa Bismuth, Aurèle Bohbot, Sébastien Coustou, Mathieu Lacome, and Mathieu Rosenbaum. What should clubs monitor to predict future value of football players, 2022. URL <https://arxiv.org/abs/2212.11041>.
- Donald Barron, Graham Ball, Matthew Robins, and Caroline Sunderland. Artificial neural networks and player recruitment in professional soccer. *PLOS ONE*, 13(10):e0205818, October 2018. ISSN 1932-6203. doi: 10.1371/journal.pone.0205818. URL <http://dx.doi.org/10.1371/journal.pone.0205818>.
- Iman Behravan and Seyed Mohammad Razavi. A novel machine learning method for estimating football players’ value in the transfer market. *Soft Computing*, 25(3):2499–2511, October 2020. ISSN 1433-7479. doi: 10.1007/s00500-020-05319-3. URL <http://dx.doi.org/10.1007/s00500-020-05319-3>.

- Tom L. G. Bergkamp, A. Susan M. Niessen, Ruud. J. R. den Hartigh, Wouter G. P. Frencken, and Rob R. Meijer. Methodological Issues in Soccer Talent Identification Research. *Sports Medicine*, 49(9):1317–1335, June 2019. ISSN 1179-2035. doi: 10.1007/s40279-019-01113-w. URL <http://dx.doi.org/10.1007/s40279-019-01113-w>.
- Manuele Bicego and Marco Loog. Weighted k-nearest neighbor revisited. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1642–1647, Cancún, Mexico, December 2016. IEEE. doi: 10.1109/ICPR.2016.7899872. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7899872>.
- Leo Breiman. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/a:1010933404324. URL <http://dx.doi.org/10.1023/A:1010933404324>.
- Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification And Regression Trees*. Chapman and Hall/CRC, 1 edition, 1984. ISBN 9781315139470. doi: 10.1201/9781315139470. URL <http://dx.doi.org/10.1201/9781315139470>.
- Joel Brooks, Matthew Kerr, and John Guttig. Developing a data-driven player ranking in soccer using predictive model weights. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*. ACM, August 2016. doi: 10.1145/2939672.2939695. URL <http://dx.doi.org/10.1145/2939672.2939695>.
- Jason Brownlee. *XGBoost With Python: Gradient Boosted Trees With XGBoost and Scikit-Learn*. 1.10 edition, 2018.
- Mattia Cefis and Maurizio Carpita. The higher-order PLS-SEM confirmatory approach for composite indicators of football performance quality. *Computational Statistics*, October 2022. ISSN 1613-9658. doi: 10.1007/s00180-022-01295-4. URL <http://dx.doi.org/10.1007/s00180-022-01295-4>.
- Sanjay Chawla, Joël Estephan, Joachim Gudmundsson, and Michael Horton. Classification of passes in football matches using spatiotemporal data. *ACM Trans. Spat. Algorithms Syst.*, 3(2):1–30, June 2017.
- Victor Chazan-Pantazalis and Christos Tjortjis. Sports Analytics for Football League Table and Player Performance Prediction. 10 2020.
- Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*. ACM, August 2016. doi: 10.1145/2939672.2939785. URL <http://dx.doi.org/10.1145/2939672.2939785>.
- Francois Chollet. *Deep learning with python*. Manning Publications, New York, NY, 2017. ISBN 9781617294433.
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989. ISSN 1435-568X. doi: 10.1007/bf02551274. URL <http://dx.doi.org/10.1007/BF02551274>.
- Tom Decroos and Jesse Davis. Interpretable prediction of goals in soccer. In *Proceedings of the AAAI-20 Workshop on Artificial Intelligence in Team Sports, AITS*. AI in Team Sports Organising Committee, dec 2020.

- Tom Decroos, Lotte Bransen, Jan Van Haaren, and Jesse Davis. 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. ACM, July 2019. doi: 10.1145/3292500.3330758. URL <http://dx.doi.org/10.1145/3292500.3330758>.
- Craig A. Depken and Tomislav Globan. Football transfer fee premiums and europe's big five. *Southern Economic Journal*, 87(3):889–908, November 2020. ISSN 2325-8012. doi: 10.1002/soej.12471. URL <http://dx.doi.org/10.1002/soej.12471>.
- Stephen Dobson and John Goddard. *The Economics of Football*. 01 2001. ISBN 9780511493225. doi: 10.1017/CBO9780511493225.
- Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.
- Jordi Duch, Joshua S. Waitzman, and Luís A. Nunes Amaral. Quantifying the performance of individual players in a team activity. *PLoS ONE*, 5(6):e10937, June 2010. ISSN 1932-6203. doi: 10.1371/journal.pone.0010937. URL <http://dx.doi.org/10.1371/journal.pone.0010937>.
- Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327, April 1976. ISSN 0018-9472. doi: 10.1109/tsmc.1976.5408784. URL <http://dx.doi.org/10.1109/TSMC.1976.5408784>.
- Xudong Fan, Xijin Zhang, and Xiong Bill Yu. Uncertainty quantification of a deep learning model for failure rate prediction of water distribution networks. *Reliability Engineering and System Safety*, 236: 109088, August 2023. ISSN 0951-8320. doi: 10.1016/j.ress.2023.109088. URL <http://dx.doi.org/10.1016/j.ress.2023.109088>.
- Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.
- Maxence Franceschi, Jean-François Brocard, Florian Follert, and Jean-Jacques Gouguet. Football players in light of economic value theory: Critical review and conceptualisation. *Managerial and Decision Economics*, n/a(n/a). doi:  $\ddot{A}$ ğ.
- Maxence Franceschi, Jean-François Brocard, Florian Follert, and Jean-Jacques Gouguet. Determinants of football players' valuation: A systematic review. *Journal of Economic Surveys*, February 2023. ISSN 1467-6419. doi: 10.1111/joes.12552. URL <http://dx.doi.org/10.1111/joes.12552>.
- Bernd Frick. The football players' labor market: Empirical evidence from the major European leagues. *Scottish Journal of Political Economy*, 54(3):422–446, June 2007. ISSN 1467-9485. doi: 10.1111/j.1467-9485.2007.00423.x. URL <http://dx.doi.org/10.1111/j.1467-9485.2007.00423.x>.
- Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010. ISSN 1548-7660. doi: 10.18637/jss.v033.i01. URL <http://dx.doi.org/10.18637/jss.v033.i01>.

- Guillaume Garrigos and Robert M. Gower. Handbook of Convergence Theorems for (Stochastic) Gradient Methods, 2023. URL <https://arxiv.org/abs/2301.11235>.
- Garry A. Gelade and Lars Magnus Hvattum. On the relationship between  $+/ -$  ratings and event-level performance statistics. *Journal of Sports Analytics*, 6(2):85–97, June 2020. ISSN 2215-0218. doi: 10.3233/jsa-200432. URL <http://dx.doi.org/10.3233/JSA-200432>.
- Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, March 2006. ISSN 1573-0565. doi: 10.1007/s10994-006-6226-1. URL <http://dx.doi.org/10.1007/s10994-006-6226-1>.
- Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL [https://proceedings.neurips.cc/paper\\_files/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf).
- N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, January 2011. ISSN 1095-7200. doi: 10.1137/090771806. URL <http://dx.doi.org/10.1137/090771806>.
- Dongkwon Han, Sunil Kwon, Jeongwoo Kim, Wooseong Jin, and Hanam Son. Comprehensive analysis for production prediction of hydraulic fractured shale reservoirs using proxy model based on deep neural network. In *Day 4 Thu, October 29, 2020. SPE*, October 2020.
- Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- Miao He, Ricardo Cachucho, and Arno Knobbe. Football player’s performance and market value. In *Proceedings of the 2nd workshop of sports analytics, ECML PKDD*, 09 2015.
- Qixiang He, Duarte Araújo, Keith Davids, Ying Hwa Kee, and John Komar. Adaptability of performance to different contextual constraints as a predictor of development and success in competitive football: A systematic review. *Movement amp; Sport Sciences - Science amp; Motricité*, (121):37–58, 2023. ISSN 2118-5743. doi: 10.1051/sm/2023011. URL <http://dx.doi.org/10.1051/sm/2023011>.
- Wenchong He and Zhe Jiang. A comprehensive survey on uncertainty quantification for deep learning, 2023. URL <https://arxiv.org/abs/2302.13425>.
- Steffen Herm, Hans-Markus Callsen-Bracker, and Henning Kreis. When the crowd evaluates soccer players’ market values: Accuracy and evaluation attributes of an online community. *Sport Management Review*, 17(4):484–492, October 2014. ISSN 1839-2083. doi: 10.1016/j.smr.2013.12.006. URL <http://dx.doi.org/10.1016/j.smr.2013.12.006>.
- Mat Herold, Floris Goes, Stephan Nopp, Pascal Bauer, Chris Thompson, and Tim Meyer. Machine learning in men’s professional football: Current applications and future directions for improving attacking play. *Int. J. Sports Sci. Coach.*, 14(6):798–817, December 2019.



- Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, February 1970. ISSN 1537-2723. doi: 10.1080/00401706.1970.10488634. URL <http://dx.doi.org/10.1080/00401706.1970.10488634>.
- Ondrej Hubáček, G Sourek, and F Zelezny. Score-based soccer match outcome modeling—an experimental review. *MathSport International*, 2019.
- Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 754–762, Beijing, China, 22–24 Jun 2014. PMLR. URL <https://proceedings.mlr.press/v32/hutter14.html>.
- Lars Magnus Hvattum. Offensive and Defensive Plus–Minus Player Ratings for Soccer. *Applied Sciences*, 10(20):7345, October 2020. ISSN 2076-3417. doi: 10.3390/app10207345. URL <http://dx.doi.org/10.3390/app10207345>.
- Lars Magnus Hvattum and Halvard Arntzen. Using ELO ratings for match result prediction in association football. *International Journal of Forecasting*, 26(3):460–470, July 2010. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2009.10.002. URL <http://dx.doi.org/10.1016/j.ijforecast.2009.10.002>.
- Lars Magnus Hvattum and Garry A. Gelade. Comparing bottom-up and top-down ratings for individual soccer players. *International Journal of Computer Science in Sport*, 20(1):23–42, May 2021. ISSN 1684-4769. doi: 10.2478/ijcss-2021-0002. URL <http://dx.doi.org/10.2478/ijcss-2021-0002>.
- Lars Magnus Hvattum and Olav Sæbø. Evaluating the efficiency of the association football transfer market using regression based player ratings. 01 2015.
- Tarak Kharrat, Ian G. McHale, and Javier López Peña. Plus–minus player ratings for soccer. *European Journal of Operational Research*, 283(2):726–736, June 2017. ISSN 0377-2217. doi: 10.1016/j.ejor.2019.11.026. URL <http://dx.doi.org/10.1016/j.ejor.2019.11.026>.
- Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, and Dmitry Gorinevsky. An interior-point method for large-scale -regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, December 2007. ISSN 1941-0484. doi: 10.1109/jstsp.2007.910971. URL <http://dx.doi.org/10.1109/JSTSP.2007.910971>.
- Igor Kononenko. Estimating attributes: Analysis and extensions of relief. In Francesco Bergadano and Luc De Raedt, editors, *Machine Learning: ECML-94*, pages 171–182, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-48365-6.
- Nicolas Leifheit and Florian Follert. Financial player valuation from the perspective of the club: the case of football. *Managing Sport and Leisure*, 28(6):618–637, June 2021. ISSN 2375-0480. doi: 10.1080/23750472.2021.1944821. URL <http://dx.doi.org/10.1080/23750472.2021.1944821>.
- Mary J. Lindstrom and Douglas M. Bates. Newton-raphson and em algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*, 83(404):1014, December 1988. ISSN 0162-1459. doi: 10.2307/2290128. URL <http://dx.doi.org/10.2307/2290128>.

- Daniel Link, Steffen Lang, and Philipp Seidenschwarz. Real time quantification of dangerousity in football using spatiotemporal tracking data. *PLOS ONE*, 11(12):e0168768, December 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0168768. URL <http://dx.doi.org/10.1371/journal.pone.0168768>.
- Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, August 1989. ISSN 1436-4646. doi: 10.1007/bf01589116. URL <http://dx.doi.org/10.1007/BF01589116>.
- Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred Hero, and Pramod K. Varshney. A primer on zeroth-order optimization in signal processing and machine learning, 2020. URL <https://arxiv.org/abs/2006.06224>.
- Yu-Ren Liu, Yi-Qi Hu, Hong Qian, Chao Qian, and Yang Yu. Zoopt: a toolbox for derivative-free optimization. *Science China Information Sciences*, 65(10), September 2022. ISSN 1869-1919. doi: 10.1007/s11432-021-3416-y. URL <http://dx.doi.org/10.1007/s11432-021-3416-y>.
- Gilles Louppe and Manoj Kumar. Bayesian optimization with skopt. [https://scikit-optimize.github.io/stable/auto\\_examples/bayesian-optimization.html#sphx-glr-auto-examples-bayesian-optimization-py](https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html#sphx-glr-auto-examples-bayesian-optimization-py), 2016. Reformatted by Holger Nahrstaedt in 2020.
- Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017. URL <https://arxiv.org/abs/1705.07874>.
- Jiajie Luo. Analysis on the Impact of COVID-19 on the Football Industry and Corresponding Strategies. *SHS Web of Conferences*, 163:03002, 2023. ISSN 2261-2424. doi: 10.1051/shsconf/202316303002. URL <http://dx.doi.org/10.1051/shsconf/202316303002>.
- Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer Berlin Heidelberg, 2005. ISBN 9783540277521. doi: 10.1007/978-3-540-27752-1. URL <http://dx.doi.org/10.1007/978-3-540-27752-1>.
- Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47–68, January 2011. ISSN 1063-5203. doi: 10.1016/j.acha.2010.02.003. URL <http://dx.doi.org/10.1016/j.acha.2010.02.003>.
- Francesca Matano, Lee F. Richardson, Taylor Pospisil, Collin Eubanks, and Jining Qin. Augmenting Adjusted Plus-Minus in Soccer with FIFA Ratings, 2018. URL <https://arxiv.org/abs/1810.08032>.
- Ian G. McHale and Benjamin Holmes. Estimating transfer fees of professional footballers using advanced performance metrics and machine learning. *European Journal of Operational Research*, 306(1):389–399, April 2023. ISSN 0377-2217. doi: 10.1016/j.ejor.2022.06.033. URL <http://dx.doi.org/10.1016/j.ejor.2022.06.033>.
- Goeffrey J McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999.
- Daniel Memmert and Dominik Raabe. *Data Analytics in Football: Positional Data Collection, Modelling and Analysis*. Routledge, May 2018. ISBN 9781351210164. doi: 10.4324/9781351210164. URL <http://dx.doi.org/10.4324/9781351210164>.

- Mrs Vidushi Mishra, Smt. Manisha Agarwal, and Neha Puri. Comprehensive and comparative analysis of neural network. *International Journal of Computer Application*, 2(8), 2018. ISSN 2250-1797. doi: 10.26808/rs.ca.i8v2.15. URL <http://dx.doi.org/10.26808/rs.ca.i8v2.15>.
- John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Li. *Applied Linear Statistical Models*. McGraw-Hill/Irwin, Boston, MA, 5th edition, 2004. ISBN 978-0073108742.
- Edward Nsolo, Patrick Lambrix, and Niklas Carlsson. *Player Valuation in European Football*, page 42–54. Springer International Publishing, 2019. ISBN 9783030172749. doi: 10.1007/978-3-030-17274-9\_4. URL [http://dx.doi.org/10.1007/978-3-030-17274-9\\_4](http://dx.doi.org/10.1007/978-3-030-17274-9_4).
- G. Pantuso and L. M. Hvattum. Maximizing performance with an eye on the finances: a chance-constrained model for football transfer market decisions. *TOP*, 29(2):583–611, October 2020. ISSN 1863-8279. doi: 10.1007/s11750-020-00584-9. URL <http://dx.doi.org/10.1007/s11750-020-00584-9>.
- Luca Pappalardo, Paolo Cintia, Paolo Ferragina, Emanuele Massucco, Dino Pedreschi, and Fosca Gianotti. PlayeRank: Data-driven Performance Evaluation and Player Ranking in Soccer via a Machine Learning Approach. *ACM Transactions on Intelligent Systems and Technology*, 10(5):1–27, September 2019. ISSN 2157-6912. doi: 10.1145/3343172. URL <http://dx.doi.org/10.1145/3343172>.
- Vineet M. Payyappalli and Jun Zhuang. A data-driven integer programming model for soccer clubs’ decision making on player transfers. *Environment Systems and Decisions*, 39(4):466–481, March 2019. ISSN 2194-5411. doi: 10.1007/s10669-019-09721-7. URL <http://dx.doi.org/10.1007/s10669-019-09721-7>.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Raffaele Poli, Roger Besson, and Loïc Ravenel. Econometric Approach to Assessing the Transfer Fees and Values of Professional Football Players. *Economies*, 10(1):4, December 2021. ISSN 2227-7099. doi: 10.3390/economies10010004. URL <http://dx.doi.org/10.3390/economies10010004>.
- Paul Power, Hector Ruiz, Xinyu Wei, and Patrick Lucey. Not all passes are created equal: Objectively measuring the risk and reward of passes in soccer from tracking data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’17. ACM, August 2017. doi: 10.1145/3097983.3098051. URL <http://dx.doi.org/10.1145/3097983.3098051>.
- Vitor Ayres Principe, Rodrigo Gomes de Souza Vale, Juliana Brandão Pinto de Castro, Luiz Marcelo Carvano, Roberto André Pereira Henriques, Victor José de Almeida e Sousa Lobo, and Rodolfo de Alkmim Moreira Nunes. A computational literature review of football performance analysis through probabilistic topic modeling. *Artificial Intelligence Review*, 55(2):1351–1371, April 2021. ISSN 1573-7462. doi: 10.1007/s10462-021-09998-8. URL <http://dx.doi.org/10.1007/s10462-021-09998-8>.
- Robert Rein and Daniel Memmert. Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. *Springerplus*, 5(1):1410, August 2016.

- Marko Robnik-Šikonja and Igor Kononenko. An adaptation of relief for attribute estimation in regression. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97*, page 296–304, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1558604863.
- Marko Robnik-Šikonja and Igor Kononenko. *Machine Learning*, 53(1/2):23–69, 2003. ISSN 0885-6125. doi: 10.1023/a:1025667309714. URL <http://dx.doi.org/10.1023/A:1025667309714>.
- Alessio Rossi, Luca Pappalardo, and Paolo Cintia. A Narrative Review for a Machine Learning Application in Sports: An Example Based on Injury Forecasting in Soccer. *Sports*, 10(1):5, December 2021. ISSN 2075-4663. doi: 10.3390/sports10010005. URL <http://dx.doi.org/10.3390/sports10010005>.
- Maaike Van Roy, Pieter Robberechts, Tom Decroos, and Jesse Davis. Valuing On-the-Ball Actions in Soccer: A Critical Comparison of xT and VAEP. 2020. URL <https://api.semanticscholar.org/CorpusID:226289375>.
- Sarah Rudd. A Framework for Tactical Analysis and Individual Offensive Production Assessment in Soccer Using Markov Chains. In *Proceedings of the Conference on Sports Analytics*, 2011. Presentation.
- Annibal Parracho Sant’Anna, Helder Gomes Costa, and Valdecy Pereira. CPP-TRI: a sorting method based on the probabilistic composition of preferences. *International Journal of Information and Decision Sciences*, 7(3):193, 2015. ISSN 1756-7025. doi: 10.1504/ijids.2015.071372. URL <http://dx.doi.org/10.1504/IJIDS.2015.071372>.
- Wojciech Sałabun, Andrii Shekhovtsov, Dragan Pamučar, Jarosław Wątróbski, Bartłomiej Kizielewicz, Jakub Więckowski, Darko Bozanić, Karol Urbaniak, and Bartosz Nyczaj. A Fuzzy Inference System for Players Evaluation in Multi-Player Sports: The Football Study Case. *Symmetry*, 12(12):2029, December 2020. ISSN 2073-8994. doi: 10.3390/sym12122029. URL <http://dx.doi.org/10.3390/sym12122029>.
- SciSports. SciSkill Index - why and how. <https://www.scisports.com/sciskill-index-why-and-how/> #, October 2020. Accessed: 27 November 2023.
- SciSports. Player valuation model, 2024. URL <https://www.scisports.com/player-valuation-model/>. Accessed: 2024-05-30.
- Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference, SciPy*. SciPy, 2010. doi: 10.25080/majora-92bf1922-011. URL <http://dx.doi.org/10.25080/Majora-92bf1922-011>.
- Peter Sloane. The Economics of Professional Football: The Football Club as a Utility Maximiser. *Scottish Journal of Political Economy*, 18:121–46, 02 1971.
- Vinscent Steve Arrul, Preethi Subramanian, and Raheem Mafas. Predicting the Football Players’ Market Value Using Neural Network Model: A Data-Driven Approach. In *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*. IEEE, April 2022. doi: 10.1109/icdcece53908.2022.9792681. URL <http://dx.doi.org/10.1109/ICDCECE53908.2022.9792681>.
- Michael Stöckl, Thomas Seidl, Daniel Marley, and Paul Power. Making Offensive Play Predictable -Using a Graph Convolutional Network to Understand Defensive Performance in Soccer. 04 2021.

- Connor Sullivan and Christopher Cronin. Improving elo rankings for sports experimenting on the English premier league. *Virginia Tech CSx824/ECEx424 technical report, VA, USA*, 2016.
- Łukasz Szczepański and Ian McHale. Beyond Completion Rate: Evaluating the Passing Ability of Footballers. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 179(2):513–533, April 2015. ISSN 1467-985X. doi: 10.1111/rssa.12115. URL <http://dx.doi.org/10.1111/rssa.12115>.
- Sean J Taylor and Benjamin Letham. Forecasting at scale. September 2017. doi: 10.7287/peerj.preprints.3190v2. URL <http://dx.doi.org/10.7287/peerj.preprints.3190v2>.
- Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, January 1996. ISSN 1467-9868. doi: 10.1111/j.2517-6161.1996.tb02080.x. URL <http://dx.doi.org/10.1111/j.2517-6161.1996.tb02080.x>.
- Koen van Arem and Mirjam Bruinsma. Extended xThreat: an explainable quality assessment method for actions in football using game context. In *Proceedings of the 15th International Conference on the Engineering of Sport*. ISEA, July 2024. Forthcoming.
- Jan Van Haaren. "Why Would I Trust Your Numbers?" On the Explainability of Expected Values in Soccer, 2021. URL <https://arxiv.org/abs/2105.13778>.
- Stefan Wager, Trevor J. Hastie, and Bradley Efron. Confidence intervals for random forests: the jackknife and the infinitesimal jackknife. *Journal of machine learning research : JMLR*, 15 1:1625–1651, 2013. URL <https://api.semanticscholar.org/CorpusID:2185396>.
- Edward Wakelam, Volker Steuber, and James Wakelam. The collection, analysis and exploitation of footballer attributes: A systematic review. *Journal of Sports Analytics*, 8(1):31–67, March 2022. ISSN 2215-0218. doi: 10.3233/jsa-200554. URL <http://dx.doi.org/10.3233/JSA-200554>.
- Stephan Wolf, Maximilian Schmitt, and Björn Schuller. A football player rating system. *Journal of Sports Analytics*, 6(4):243–257, January 2021. ISSN 2215-0218. doi: 10.3233/jsa-200411. URL <http://dx.doi.org/10.3233/JSA-200411>.
- Yanxiang Yang, Joerg Koenigstorfer, and Tim Pawlowski. Predicting transfer fees in professional European football before and during COVID-19 using machine learning. *European Sport Management Quarterly*, page 1–21, December 2022. ISSN 1746-031X. doi: 10.1080/16184742.2022.2153898. URL <http://dx.doi.org/10.1080/16184742.2022.2153898>.
- Ahmet Talha Yigit, Baris Samak, and Tolga Kaya. An XGBoost-lasso ensemble modeling approach to football player value assessment. *Journal of Intelligent amp; Fuzzy Systems*, 39(5):6303–6314, November 2020. ISSN 1875-8967. doi: 10.3233/jifs-189098. URL <http://dx.doi.org/10.3233/JIFS-189098>.
- Xiaozhe Yin, Masoud Fallah-Shorshani, Rob McConnell, Scott Fruin, Yao-Yi Chiang, and Meredith Franklin. Quantile extreme gradient boosting for uncertainty quantification, 2023. URL <https://arxiv.org/abs/2304.11732>.
- Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021. doi: 10.48550/arXiv.1611.03530. URL <https://arxiv.org/abs/1611.03530v2>.