Data mining

Final project

Jakub Laszczyk

Table of contents

Introduction

The aim of this project is to investigate the effectiveness of machine learning methods in predicting the outcomes of football matches and to understand whether it is possible to gain an advantage over betting odds using these techniques based on selected data.

The stages of the machine learning process are presented sequentially, and important metrics for evaluating model effectiveness are discussed. The dataset used in the analysis is described in detail, and three different supervised learning methods applied and compared: decision trees, random forests, and gradient boosting. These methods were chosen for their interpretability and ability to handle data with a complex structure, characteristic of sports results. The study utilized football data from publicly available datasets provided by Football-Data and Transfermarkt, selected for their extensive range of information and the reliability of the presented data.

The next step involved conducting a comprehensive data analysis, including descriptive statistics, examining correlations between variables, and identifying outliers. In the final stage of the study, the implemented methods were presented. The obtained results were compared with each other and against the betting odds.

1.1. Description of the Dataset

The following data files are included:

- Data3 - Input dataset
- Data_clean – cleaned dataset with outliers removed
- Data4 – data from one season used in the final phase of the project, after building the model and making predictions to determine the rate of return
- Season2023_s – data with added betting odds

The dataset contains statistics from all matches of the last 10 seasons of the Premier League. The match data comes from the website England Football Results Betting Odds | Premiership Results & Betting Odds (football-data.co.uk), and the team values were sourced from Transfermarkt. Analysis and model building were conducted on data from the 2013/2014 to 2021/2022 seasons, while predictions were made for the 2022/2023 season. Due to differences in data availability (initially, the data included events from each match, whereas such data would not be available for new matches during prediction), the variables were transformed to only concern previous events. The variables were divided into several categories. Each category pertains to key aspects of football match analysis. Below is a description of each category:

Form

Variables describing team form refer to the average points earned in the last 5 or 3 matches. The variables HT_AvgPointsCurrentS and AT_AvgPointsCurrentS represent the current season's table positions expressed as the average points earned in the matches played so far this season.

Variables related to team form.

Variable name	Description	Variable type
HT_pointsLast3g	average points earned by the home team in the last 3 matches	Numeric (from 0 to 3)

AT_pointsLast3g	Average points scored by the away team in the last 3 matches	Numeric (from 0 to 3)
HT_poinstLast5g	home team's average points scored in the last 5 matches	Numeric (from 0 to 3)
AT_pointsLast5g	Average points scored by the away team in the last 5 matches	Numeric (from 0 to 3)
HT_AvgPointsCurrentS	home team's average points point current season	Numeric (from 0 to 3)
AT_AvgPointsCurrentS	Average points of the visiting team in the current season	Numeric (from 0 to 3)

Historic results

To express the team's strength in a historical context and provide the model with information about the performance of individual teams in previous years, the following variables were created:

Variables about historical performance.

Variable name	Description	Variable type
HT_AvgPLastS	Last season's average points for the home team	Numeric (from 0 to 3)
AT_AvgPLastS	Average points from last season for the away team	Numeric (from 0 to 3)
HT_AvgPLast2S	Average points from the last two seasons for the home team	Numeric (from 0 to 3)

AT_AvgPLast2S	Average points from the last two seasons for the away team	Numeric (from 0 to 3)

Home field advantage

To illustrate the home advantage effect, 2 tables showing the final result of the 2021/2022 Premier League season have been compared.

Tables of the 2021/2022 Premier League season broken down into home and away matches.

Team	P	W	D	L	GF	GA	GD	Pts
1 Manchester City	19	17	1	1	60	17	+43	52
2 Manchester United	19	15	3	1	36	10	+26	48
3 Arsenal	19	14	3	2	53	25	+28	45
4 Liverpool	19	13	5	1	46	17	+29	44
5 Newcastle	19	11	6	2	36	14	+22	39
6 Aston Villa	19	12	2	5	33	21	+12	38
7 Brentford	19	10	7	2	35	18	+17	37
8 Tottenham	19	12	1	6	37	25	+12	37
9 Brighton	19	10	4	5	37	21	+16	34
10 Nottingham Forest	19	8	6	5	27	24	+3	30
11 Wolves	19	9	3	7	19	20	-1	30
12 Fulham	19	8	5	6	31	29	+2	29
13 West Ham	19	8	4	7	26	24	+2	28
14 Crystal Palace	19	7	7	5	21	23	-2	28
15 Chelsea	19	6	7	6	20	19	+1	25
16 Bournemouth	19	6	4	9	20	28	-8	22
17 Leeds	19	5	7	7	26	37	-11	22
18 Everton	19	6	3	10	16	27	-11	21
19 Leicester	19	5	4	10	23	27	-4	19
20 Southampton	19	2	5	12	19	37	-18	11

Team	P	W	D	L	GF	GA	GD	Pts
1 Arsenal	19	12	3	4	35	18	+17	39
2 Manchester City	19	11	4	4	34	16	+18	37
3 Newcastle	19	8	8	3	32	19	+13	32
4 Brighton	19	8	4	7	35	32	+3	28
5 Manchester United	19	8	3	8	22	33	-11	27
6 Fulham	19	7	2	10	24	24	0	23
7 Liverpool	19	6	5	8	29	30	-1	23
8 Tottenham	19	6	5	8	33	38	-5	23
9 Aston Villa	19	6	5	8	18	25	-7	23
10 Brentford	19	5	7	7	23	28	-5	22
11 Chelsea	19	5	4	10	18	28	-10	19
12 Crystal Palace	19	4	5	10	19	26	-7	17
13 Bournemouth	19	5	2	12	17	43	-26	17
14 Everton	19	2	9	8	18	30	-12	15
15 Leicester	19	4	3	12	28	41	-13	15
16 Southampton	19	4	2	13	17	36	-19	14
17 West Ham	19	3	3	13	16	31	-15	12
18 Wolves	19	2	5	12	12	38	-26	11
19 Leeds	19	2	3	14	22	41	-19	9
20 Nottingham Forest	19	1	5	13	11	44	-33	8

From the analysis of the above tables, it is evident that all clubs except Southampton had a better average points tally in home matches than in away matches. Notably, some clubs performed significantly better in home games than away games. For example, Nottingham Forest ranked 10th in the home matches table but 20th in the away matches table, earning only 8 points in 19 away games. The Wolves team earned the same number of points in home matches and only 2 points more in away matches. To provide information on the home-field advantage, the following data were generated:

Variables describing home field advantage.

Variable name	Description	Variable type
HT_pointsHLast3g	home team's average points scored in their last 3 home games	Numeric (from 0 to 3)
AT_pointsALast3g	Average points scored by the away team in the last 3 away games	Numeric (from 0 to 3)
HT_AvgGoalsHLastS	Average goals scored by the home team in their home matches in the previous season	Numeric (from 0,7368 to 3,3158)
AT_AvgGoalsALastS	Average goals scored by the away team in their away matches in the previous season	Numeric (from 0,5263 to 2,5263)

Offensive and defensive strength

Offensive power was included as the average of shots, average of shots on target and average of goals scored, while defensive strength as the average of goals conceded.

Variables describing offensive and defensive strength.

Variable name	Description	Variable type
HT_ShotsLast5g	average shots taken by the home team in the last 5 matches	Numeric (from 5,2 to 24,6)
AT_ShotsLast5g	average shots taken by away team in the last 5 matches	Numeric (from 4,4 to 25)
HT_ShotsOTLast5g	average shots on target by home team over last 5 matches	Numeric (from 0,8 to 10,4)

AT_ShotsOTLast5g	average shots on target by the away team in the last 5 matches	Numeric (from 1 to 10,8)
HT_Goals_Last5g	Average goals scored by home team in their last 5 matches	Numeric (from 0 to 4,8)
AT_Goals_Last5g	Average goals scored by away team in their last 5 matches	Numeric (from 0 to 4,4)
HT_GoalsConLast5	Average a goals conceded by the home team in the last 5 matches	Numeric (from 0 to 3,6)
AT_GoalsConLast5	Average goals conceded by the away team in their last 5 matches	Numeric (from 0 to 4)

Team value

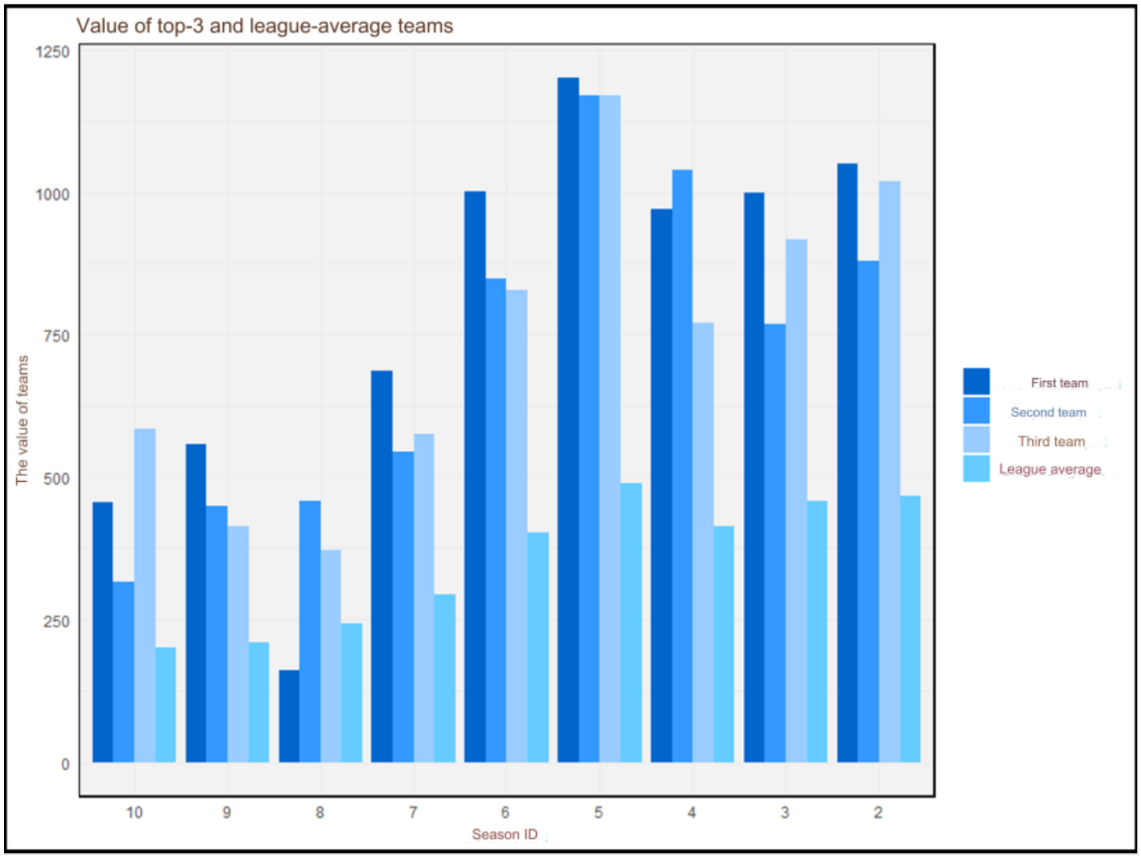
Every sports competition is characterized by the presence of favorites and so-called underdogs, which are teams deemed to have less chance of winning. During the analysis of this study, it was observed that in the vast majority of seasons, the league's top teams had significantly higher team values than the average. Additionally, this trend has increased in recent years. Over the examined 9 seasons, only once did a team finishing in the top 3 have a lower squad value than the average. Moreover, only in the 2015/2016 season (when Leicester sensationally won the league with the eleventh largest budget) did the teams that finished in the top three positions include groups outside the top 5 largest budgets in the league. In all other seasons, there is a noticeable pattern where teams finishing in the top 3 also ranked among the top 5 highest-valued squads.

From this analysis, we can conclude that teams with higher squad values have a greater chance of achieving favorable results. The graph below shows the values of teams finishing in the top 3 positions and the average team value in a given season. The graph indicates an accelerated increase in the average value of teams in the league since the 2016/2017 season. This trend is attributed to the record-breaking contract for the sale of

broadcasting rights for English Premier League matches. Therefore, the variables were scaled using min-max scaling for each season separately according to the given formula::

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

Histogram of team values.



Variables describing the team value.

Nazwa zmiennej	Opis	Typ zmiennej
HomeTeam_Value	The value of the home team at the beginning of the season.	Numeric (from 0 to 1)
AwayTeam_Value	the value of the away team at the beginning of the season.	Numeric (from 0 to 1)

Head-to-head record

From my observations, some teams prefer playing against certain opponents. It is worth noting that this has not been statistically confirmed - it is merely an observation I decided to investigate.

Variables describing results in direct games.

Variable name	Description	Variable type
h2h_diff	różnica w punktach w meczach bezpośrednich między drużynami.	Numeric (from -3 to 3)

Bookmakers' odds

bet365's odds for league matches were used to assess the quality of the model.

Variables describing the bookmaker's odds.

Variable name	Description	Typ zmiennej
B3651	bet365 bookmaker odds for home team to win.	Numeryczna (od 1,06 do 23)
B365x2	bet365 bookmaker's odds on a draw or away team.	Numeric (from 1,005 to 11,333)

Output variable

A variable describing the result of a match.

Zmienna opisująca wynik spotkania.

Variable name	Description	Variable type
HomeWin	A binary variable that describes the home team's victory.	Categorical (1 - the home team won the match

		0 – no win for the home team)

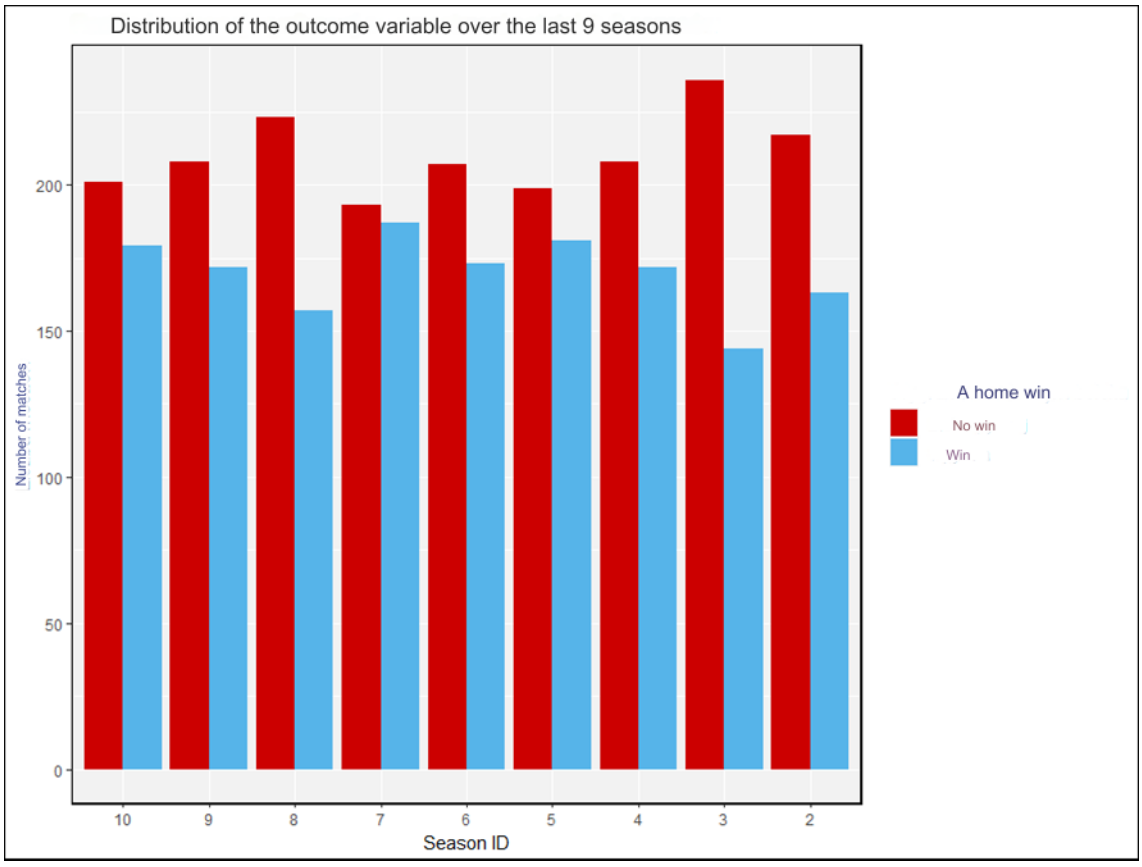
The distribution of the HomeWin variable is as follows:

Distribution of the result variable.

0 – Home team win	1 – draw or win away team
1892	1528

55.32% of the matches ended with a home team victory, while 44.68% ended in a draw or an away team victory. To illustrate the distribution of this variable across the entire dataset, broken down by season, a bar chart was generated.:

Distribution of the outcome variable over 9 seasons.



The distribution of the outcome variable is similar across the studied seasons. The largest differences were observed in the 2021/2022 and 2015/2016 seasons when the home teams won 37.9% and 41.31% of the matches, respectively.

Other variables

Other variables used.

Variable name	Description	Variable type
SeasonID	A variable describing the affiliation to the corresponding season, 1 means the latest season (2022/2023), 10 means the oldest season under study (2013/2014).	Numeric (from 1 to 10)
HomeTeam	Describes the name of the home team	text
AwayTeam	Describes the name of the away team	text

2. Presentation of the results of the empirical study

2.1. Missing data

Most of the missing data occurred due to the nature of variables based on several previous rounds. Matches from these rounds were removed. Other missing data were found in the case of newly promoted teams, i.e., teams that were promoted to the Premier League in that season. This effect was due to the presence of variables that provided information about the team's performance in previous seasons. Removing these observations would lead to a complete loss of information about these teams. Given the differences between the Premier League and the second tier of English football, as well as the historical performance of teams after promotion, it was deemed incorrect to use statistics describing the team's performance in the lower league. The best approach chosen was to fill in the missing values with the average of the corresponding statistics of newly promoted teams from the previous season. The remaining missing data concerned the variable describing the number of points earned in head-to-head matches. Again, this issue occurred with newly promoted teams. In this situation, however, it was decided to replace the missing data with the number 1, suggesting a draw in the last match played between these teams.

2.2. Analysis of outliers

Outliers are observations that significantly differ from the rest of the sample. They can be either larger or smaller than the other results. This effect leads to a different relationship between these observations and the dependent variable compared to other cases. The presence of outliers can be caused by calculation errors, incorrect data entry, the influence of rare phenomena, or other anomalies in the dataset. They bring several problems, such as changes in the sample mean, variance, and other descriptive statistics. In the case of machine learning, outliers can introduce unnecessary noise into the database, lead to incorrect predictions, or cause overfitting of the model. Although there is no single best method for detecting outliers, many techniques described in the literature help to handle this phenomenon. The IQR and z-score methods are analyzed in the following sections.

2.2.1. IQR

Detecting outliers using the interquartile range (the difference between the third and first quartile) is one of the simplest and most commonly used methods. An

observation is considered an outlier if it is below the lower limit or above the upper value. Formulas needed to use this method:

Interquartile range: $IQR = Q_3 - Q_1$

Upper limit: $Q_3 + 1{,}5 * IQR$

Lower limit: $Q_1 - 1{,}5 * IQR$

Wyniki przeprowadzonej analizy metodą IQR.

Number of observations	Lower limit	Upper limit	Variable name
0	-1	4,333333	HT_pointsHLast3g
0	-1,66667	3,666667	AT_pointsALast3g
0	-1,33333	4	HT_pointsLast3g
0	-0,5	3,5	AT_pointsLast3g
0	-0,7	3,3	HT_poinstLast5g
0	-1	3,8	AT_pointsLast5g
0	-4,5	4,3	h2h_diff
40	4,4	20,4	HT_ShotsLast5g
43	4,6	20,6	AT_ShotsLast5g
33	0,425	7,825	HT_ShotsOTLast5g
46	0,7	7,9	AT_ShotsOTLast5g
22	-0,7	3,3	HT_Goals_Last5g
29	-0,7	3,3	AT_Goals_Last5g
23	-0,2	3	HT_GoalsConLast5
14	-0,2	3	AT_GoalsConLast5
0	-0,11184	2,888158	HT_AvgPLast2S
0	-0,09211	2,855263	AT_AvgPLast2S
0	0,048246	2,75	HT_AvgPLastS
0	0,048246	2,75	AT_AvgPLastS
143	0,210526	2,736842	HT_AvgGoalsHLastS
0	-0,13158	2,605263	AT_AvgGoalsALastS
0	-0,67742	1,340136	HomeTeam_Value
0	-0,6269	1,25786	AwayTeam_Value

13	-0,1	2,833333	HT_AvgPointsCurrentS
12	-0,125	2,875	AT_AvgPointsCurrentS

2.2.2. Z-score

This method measures how far from the mean a value is in units of standard deviation. For each observation, the Z statistic is calculated:

$$Z_i = \frac{x_i - \bar{x}}{S_x}$$

Where:

x_i - value i – observation.

$\bar{x}$ - sample average.

S_x - Standard deviation from the sample.

The most commonly accepted value according to which an observation is considered an outlier is 3 standard deviations, this value was also adopted in my calculations.

Results of the z-score analysis.

Number of observations	Lower limit	Upper limit	Variable name
0	-0,90668	4,066846	HT_pointsHLast3g
0	-1,2956	3,660967	AT_pointsALast3g
0	-1,14665	3,838847	HT_pointsLast3g
0	-1,0965	3,938657	AT_pointsLast3g
0	-0,73307	3,454096	HT_poinstLast5g
0	-0,66911	3,480953	AT_pointsLast5g
0	-5,06366	4,896154	h2h_diff
12	3,287127	21,7346	HT_ShotsLast5g
13	3,332357	22,16848	AT_ShotsLast5g
14	0,123843	8,380196	HT_ShotsOTLast5g
14	0,127261	8,561457	AT_ShotsOTLast5g
22	-0,6457	3,32397	HT_Goals_Last5g

29	-0,62556	3,367068	AT_Goals_Last5g
23	-0,45767	3,199031	HT_GoalsConLast5
14	-0,45894	3,132062	AT_GoalsConLast5
0	0,153629	2,644183	HT_AvgPLast2S
0	0,155299	2,640915	AT_AvgPLast2S
0	0,126292	2,70023	HT_AvgPLastS
0	0,127696	2,697141	AT_AvgPLastS
32	0,008869	3,069712	HT_AvgGoalsHLastS
16	-0,07071	2,524551	AT_AvgGoalsALastS
0	-0,61602	1,267889	HomeTeam_Value
0	-0,61519	1,265777	AwayTeam_Value
1	-0,19595	2,936893	HT_AvgPointsCurrentS
4	-0,17905	2,946498	AT_AvgPointsCurrentS

In both methods, outliers were removed except for the variables HT_AvgGoalshLastS and AT_AvgGoalsALastS. Since both variables describe the average number of goals scored in the previous season, removing these variables would result in the removal of all matches of teams that scored the most goals in the previous league season. To avoid potential information loss due to the removal of these observations, it was decided to logarithmize the aforementioned variables. Logarithmization can reduce the effect of outliers while retaining the information they contain. Both datasets created by removing outliers were compared in terms of model accuracy. It was observed that the model based on data reduced using the Z-score method yielded better results. This may be due to the fewer number of removed observations compared to the other dataset. Therefore, further analysis will be based on this dataset, which contains 2,746 observations.

2.3. Analysis of basic descriptive statistics

Some variables will be described together because their values are very similar, and their characteristics pertain to the same statistics, albeit divided between the home team and the away team. The description will use the statistics of the first mentioned variable.

Variable name	Mean	Standard deviation	Median	Min	Max	Max-min	Skewness	Kurtozis	Standard Error
HT_pointsHLast3g	1,570527	0,819208	1,333333	0	3	3	0,052096	-0,70703	0,015633
AT_pointsALast3g	1,173464	0,816134	1	0	3	3	0,383789	-0,60376	0,015574
HT_pointsLast3g	1,333697	0,814536	1,333333	0	3	3	0,203288	-0,707	0,015544
AT_pointsLast3g	1,408109	0,828402	1,333333	0	3	3	0,160417	-0,76625	0,015809
HT_poinstLast5g	1,349017	0,679178	1,4	0	3	3	0,219253	-0,43822	0,012961
AT_pointsLast5g	1,39461	0,676367	1,4	0	3	3	0,147655	-0,57971	0,012907
h2h_diff	-0,08563	1,661626	0	-3	3	6	0,000428	-0,55827	0,031709
HT_ShotsLast5g	12,40517	2,948088	12	5,2	21,6	16,4	0,465505	0,01389	0,056259
AT_ShotsLast5g	12,64406	3,012967	12,4	4,4	22	17,6	0,4162	-0,10506	0,057497
HT_ShotsOTLast5g	4,201238	1,313219	4	0,8	8,2	7,4	0,427227	-0,15242	0,02506
AT_ShotsOTLast5g	4,292353	1,338289	4,2	1	8,2	7,2	0,469646	-0,13572	0,025539
HT_Goals_Last5g	1,314567	0,624525	1,2	0	3,2	3,2	0,523997	-0,08747	0,011918
AT_Goals_Last5g	1,343554	0,622613	1,2	0	3,2	3,2	0,470549	-0,0843	0,011881
HT_GoalsConLast5	1,363001	0,583489	1,4	0	3	3	0,300348	-0,30728	0,011135
AT_GoalsConLast5	1,33496	0,580918	1,2	0	3	3	0,35626	-0,19404	0,011086
HT_AvgPLast2S	1,39152	0,409665	1,223684	0,947368	2,605263	1,657895	0,851578	-0,35494	0,007818
AT_AvgPLast2S	1,391498	0,41046	1,223684	0,947368	2,605263	1,657895	0,856032	-0,34569	0,007833
HT_AvgPLastS	1,403838	0,423505	1,236842	0,894737	2,631579	1,736842	0,902132	-0,12059	0,008082
AT_AvgPLastS	1,403879	0,42373	1,236842	0,894737	2,631579	1,736842	0,906046	-0,11338	0,008086
HT_AvgGoalsHLastS	0,377209	0,304395	0,351398	-0,30538	1,198696	1,504077	0,433989	-0,2263	0,005809
AT_AvgGoalsALastS	0,137268	0,33952	0,100083	-0,64185	0,926762	1,568616	0,148344	-0,63308	0,006479
HomeTeam_Value	0,319496	0,309496	0,177001	0	1	1	0,918295	-0,53896	0,005906
AwayTeam_Value	0,319108	0,309802	0,177001	0	1	1	0,929302	-0,51777	0,005912
HT_AvgPointsCurrentS	1,357755	0,509778	1,285714	0	2,923077	2,923077	0,387062	-0,25362	0,009728
AT_AvgPointsCurrentS	1,372675	0,508543	1,291667	0	2,925926	2,925926	0,403073	-0,22565	0,009705

HT_pointsHLast3g – On average, teams earned 1.57 points in their last 3 home matches, with a standard deviation of 0.819208. Half of the data points are below 1.333. The skewness close to 0 suggests a balanced distribution of the variable around the mean.

AT_pointsALast3g – Teams earned an average of 1.173464 points in their last 3 away matches. The standard deviation is ±0.816134. Positive skewness indicates that more values are below the mean. The minimum value of 0 and the maximum value of 3 likely pertain to the early rounds of the season.

HT_pointsLast3g and AT_pointsLast3g – Teams averaged 1.333697 points in their last three matches. The median is close to this average, and the skewness is almost zero. This indicates that the data have a symmetrical distribution, meaning most teams achieved results close to the average, with an even distribution of points and no significant clustering or deviations in the data.

HT_poinstLast5g and AT_pointsLast5g – The average points earned by teams in their last 5 matches is 1.349017, close to the average points earned in the last 3 matches. Half of the data points are above 1.4, and the data are symmetrically distributed around the mean.

h2h_diff – The mean is close to the median, and the standard deviation of 1.661626 indicates a large spread of the data.

HT_ShotsLast5g and AT_ShotsLast5g – On average, teams took over 12 shots on target in the last 5 matches. The minimum value was 5.2, and the maximum was 21.6. More values were below the mean.

HT_ShotsOTLast5g and AT_ShotsOTLast5g – The minimum value was 0.8, indicating a team had fewer than one shot on target per match over 5 matches. The maximum value was 8.2, and on average, teams had 4.201238 shots on target per match.

HT_AvgGoalsHLastS – The median is close to the mean. There is little deviation from the mean.

AT_AvgGoalsALastS – Lower mean and median values than HT_AvgGoalsHLastS suggest that teams scored fewer goals on average in away matches compared to home matches.

HT_Goals_Last5g and AT_Goals_Last5g – On average, teams scored just over one goal per match. The median was 1.2, suggesting most teams achieved results close to this value. The skewness of 0.523997 indicates that more values were below the mean.

HT_GoalsConLast5 and AT_GoalsConLast5 – Teams conceded an average of 1.363001 goals per match, which is naturally close to the number of goals scored by teams. The standard deviation was 0.624525.

HT_AvgPLast2S and AT_AvgPLast2S – Over two seasons, teams averaged 1.39152 points, with a standard deviation of 0.409665. The best team averaged 2.605263 points. Skewness of 0.851578 indicates the distribution of points tends to stretch towards higher values.

HT_AvgPLastS and AT_AvgPLastS – The statistics of these variables are very close to those describing the average points from the last two seasons. This suggests that both variables likely describe similar phenomena or trends. In such a case, to avoid excessive redundancy, it may be considered to remove one of these variables.

HomeTeam_value and AwayTeam_value – The variables were standardized using the min-max method. Skewness of 0.918295 indicates a strong asymmetry towards higher values. Most observations are below the mean, with a few observations reaching very high values.

2.4 Correlation analysis

To appropriately select the variables, a Pearson correlation analysis was conducted. This analysis measures the strength of the linear relationship between variables. Its limitation is its susceptibility to outliers. There is no universally accepted strict division for the level of correlation deemed correct in the literature. The classification depends on the problem at hand and the characteristics of the data. In this case, the following classification method was applied::

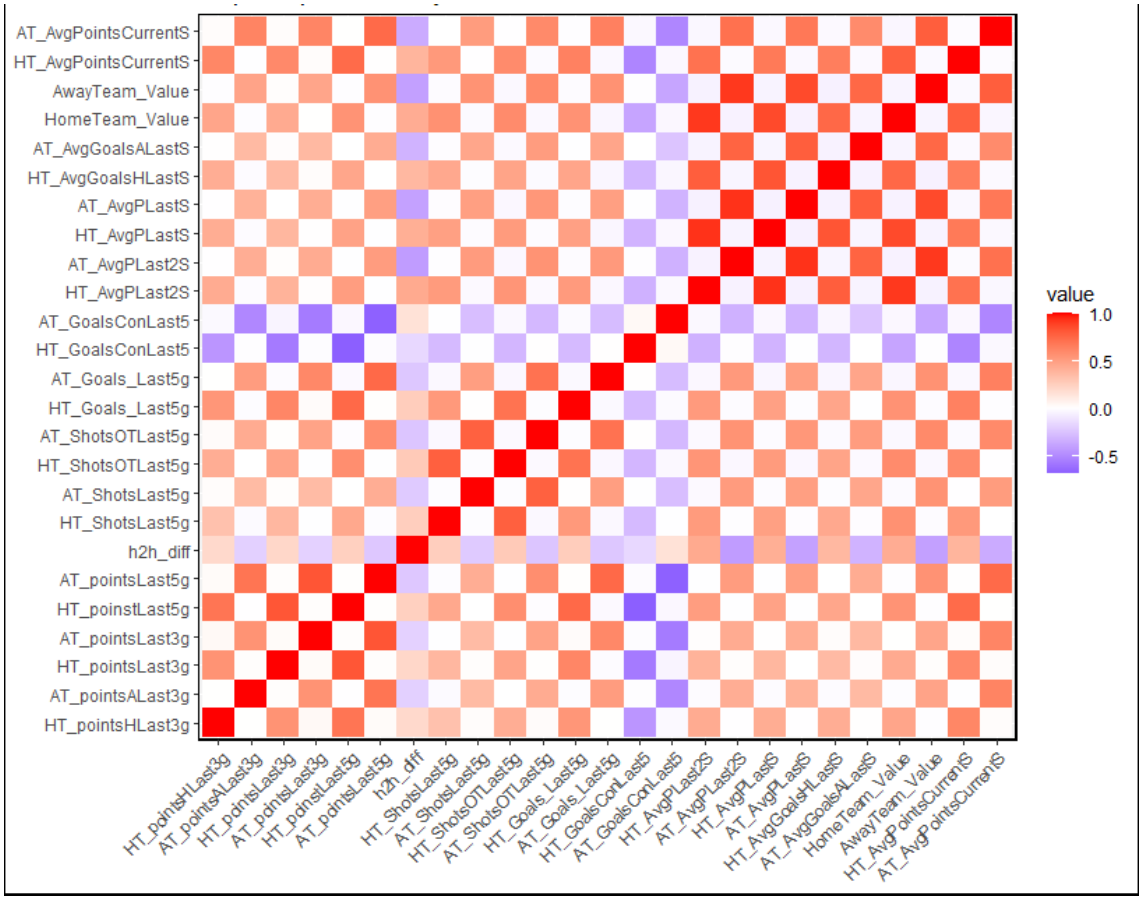
Interpretation.

Correlation size	Interpretation
0,90 to 1,00 (-0,90 to -1,00)	Very strong positive (negative) correlation
0,70 to 0,90 (-0,70 to -0,90)	Strong positive (negative) correlation
0,50 do 0,70 (-0,50 do -0,70)	Moderate positive (negative) correlation

0,30 to 0,50 (-0,30 to -0,50)	Low positive (negative) correlation
0,00 to 0,30 (0,00 to -0,30)	Little or no correlation

The results were presented using a heatmap, where higher correlation values are represented by warmer shades. A strong positive correlation is depicted in intense red, while a strong negative correlation is shown in purple. In cases where the color fades, there is a weak correlation or none at all

Correlation Heatmap.



The heatmap clearly shows clusters of highly correlated variables. One such area is visible in the bottom left corner. The variables in this part of the map relate to the team's recent form, goals scored, and points earned. A higher number of goals scored typically leads to a higher average number of points earned. Therefore, a high correlation between these variables can be expected. Another area where we observe a strong positive correlation between several variables is the top right corner. This area includes variables describing team strength, such as team value, points earned in previous seasons, and the average

points earned in the current season. We see a strong correlation between team value and points earned in the previous season, which is consistent with the analysis conducted in the description of the team value variable. We can also observe that variables show significant correlation in categories related to the home team and the away team. Below are all pairs with a correlation greater than 0.7.Pary zmiennych z korelacją wynoszącą powyżej 0,7.

Variable 1	Variable 2	Size of correlation
HT_poinstLast5g	HT_pointsLast3g	0,819793
AT_pointsLast5g	AT_pointsLast3g	0,823328
HT_Goals_Last5g	HT_poinstLast5g	0,742126
HT_AvgPointsCurrentS	HT_poinstLast5g	0,735597
AT_Goals_Last5g	AT_pointsLast5g	0,744286
AT_AvgPointsCurrentS	AT_pointsLast5g	0,738733
HT_ShotsOTLast5g	HT_ShotsLast5g	0,779908
AT_ShotsOTLast5g	AT_ShotsLast5g	0,778182
HT_Goals_Last5g	HT_ShotsOTLast5g	0,703178
AT_Goals_Last5g	AT_ShotsOTLast5g	0,705234
HT_AvgPLastS	HT_AvgPLast2S	0,931322
HT_AvgGoalsHLastS	HT_AvgPLast2S	0,787951
HomeTeam_Value	HT_AvgPLast2S	0,915052
HT_AvgPointsCurrentS	HT_AvgPLast2S	0,707621
AT_AvgPLastS	AT_AvgPLast2S	0,929697
AT_AvgGoalsALastS	AT_AvgPLast2S	0,766475
AwayTeam_Value	AT_AvgPLast2S	0,915225
AT_AvgPointsCurrentS	AT_AvgPLast2S	0,710383
HT_AvgGoalsHLastS	HT_AvgPLastS	0,824391
HomeTeam_Value	HT_AvgPLastS	0,862683
AT_AvgGoalsALastS	AT_AvgPLastS	0,787689
AwayTeam_Value	AT_AvgPLastS	0,861707
HomeTeam_Value	HT_AvgGoalsHLastS	0,746098
AwayTeam_Value	AT_AvgGoalsALastS	0,749849

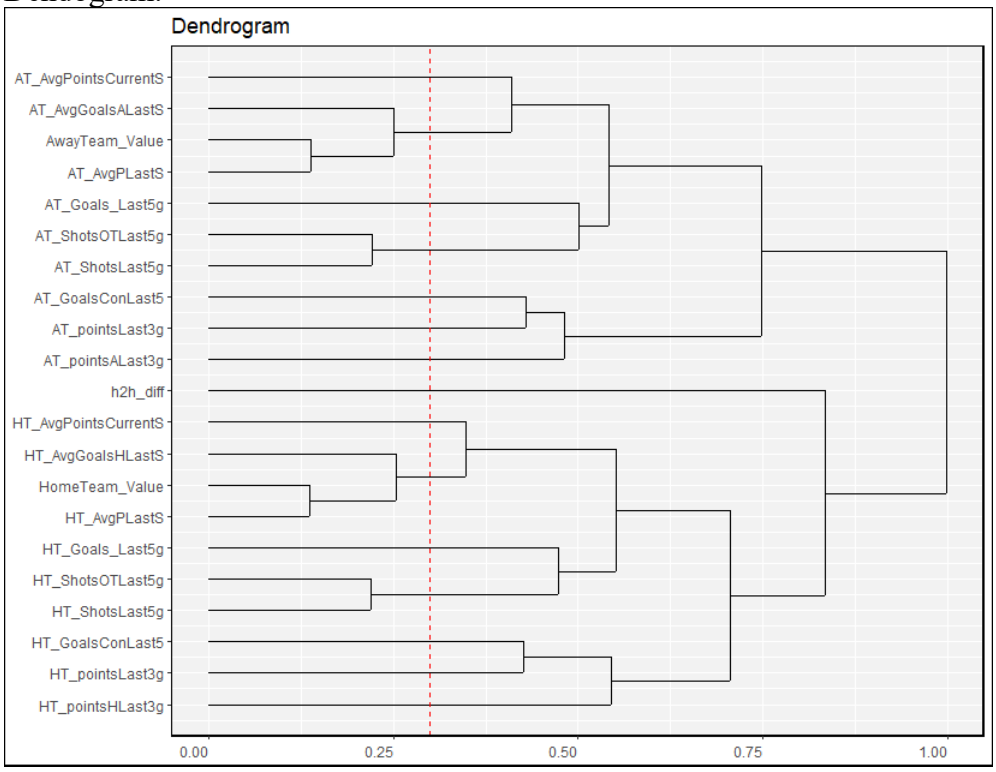
HT_AvgPointsCurrentS	HomeTeam_Value	0,782218
AT_AvgPointsCurrentS	AwayTeam_Value	0,787619

Due to their high correlations with other variables, the following variables were removed: HT_AvgPLast2S, AT_AvgPLast2S, AT_pointsLast5g, and HT_pointsLast5g. The remaining variables with strong correlations will be considered in further analysis; it may be necessary to remove them at a later stage.

2.5. Cluster analysis

To select an appropriate set of variables for the model, a hierarchical cluster analysis was performed. The goal of this analysis is to group variables characterized by strong correlations within groups and weak correlations between groups. The DIANA (Divisive Analysis) algorithm was chosen for the cluster analysis. This algorithm is based on divisive hierarchical clustering. It starts by creating one large cluster. In subsequent steps, the most heterogeneous cluster is selected and recursively divided into two. The algorithm ends when all data points belong to appropriate clusters or when a specific stopping criterion is met. The results of the algorithm are presented in a dendrogram. The cutoff threshold corresponds to a correlation of 0.7.

Dendogram.



As a result of the cluster analysis, the data was divided into 15 clusters:

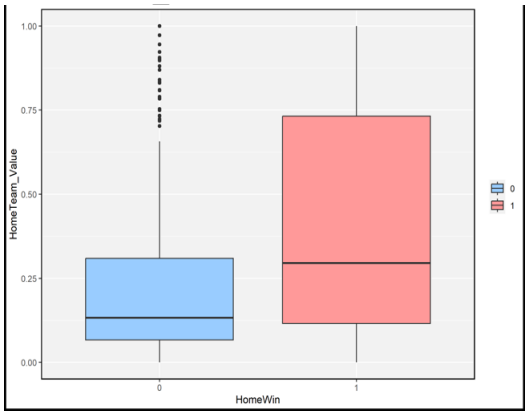
Variable	Cluster
HT_pointsHLast3g	1
AT_pointsALast3g	2
HT_pointsLast3g	3
AT_pointsLast3g	4
h2h_diff	5
HT_ShotsLast5g	6
HT_ShotsOTLast5g	6
AT_ShotsLast5g	7
AT_ShotsOTLast5g	7
HT_Goals_Last5g	8
AT_Goals_Last5g	9
HT_GoalsConLast5	10
AT_GoalsConLast5	11
HT_AvgPLastS	12
HT_AvgGoalsHLastS	12
HomeTeam_Value	12
AT_AvgPLastS	13
AT_AvgGoalsALastS	13
AwayTeam_Value	13
HT_AvgPointsCurrentS	14
AT_AvgPointsCurrentS	15

2.6 Selecting an output dataset

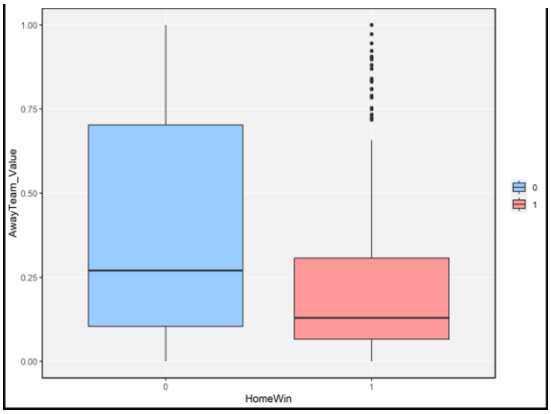
To gain a better understanding of the relative predictive ability of individual variables, an analysis was conducted using box plots. These plots illustrate the distribution characteristics of the values of individual variables in the context of different values of the dependent variable, i.e., "HomeWin". The analysis considered several key aspects. First, it examined whether the variables showed similar median values for both categories of the dependent variable ("HomeWin"). This helped determine if there was a

clear difference in the tendency of the variables between the two groups. Additionally, special attention was paid to the interquartile range, which indicates the distribution of data between the first and third quartiles. Analyzing this range allowed for assessing in which value ranges there is greater variability between the variable values for different "HomeWin" values. This approach helped identify variables that potentially have a significant impact on predictive ability concerning the "HomeWin" dependent variable. The variables considered the best in the context of prediction are: HomeTeam_Value, AwayTeam_Value, HT_AvgGoalsHLastS, HT_AvgPointsCurrentS, and AT_pointsALast3g. Their box plots are shown below:

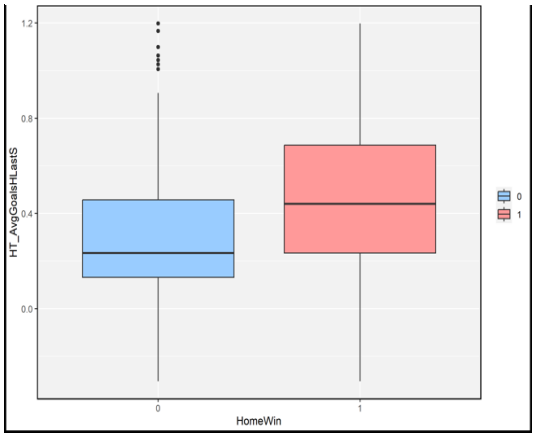
Boxplot of HomeTeam_Value.



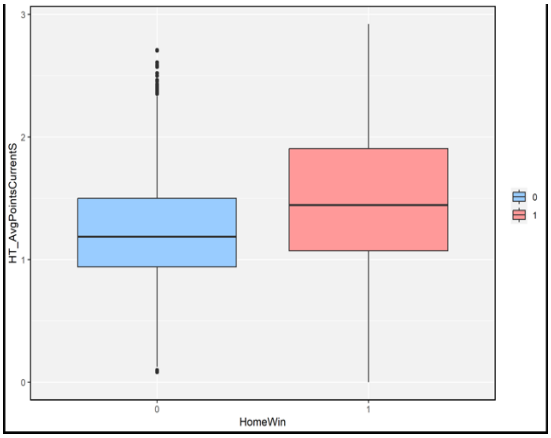
. Boxplot of AwayTeam_value.



Boxplot of HT_AvgGoalsHlastS.

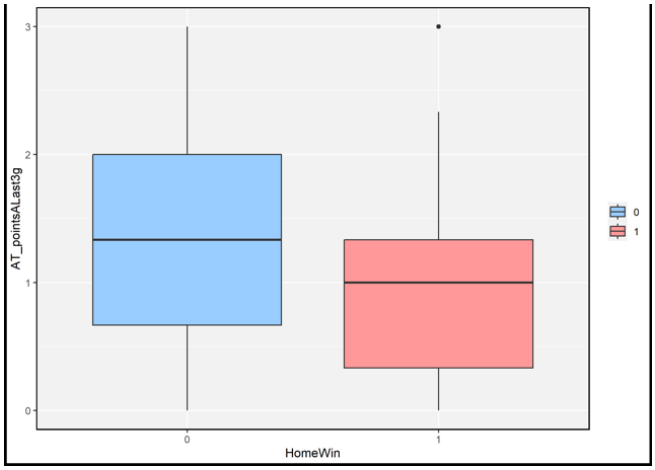


Boxplot of HT_AvgPointsCurrentS.



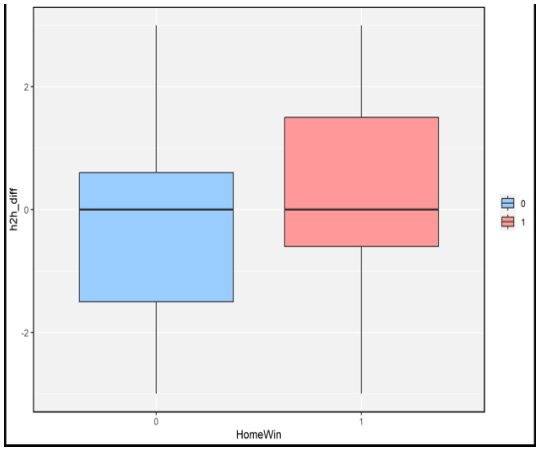
.

Boxplot of AT_pointsALast3g.

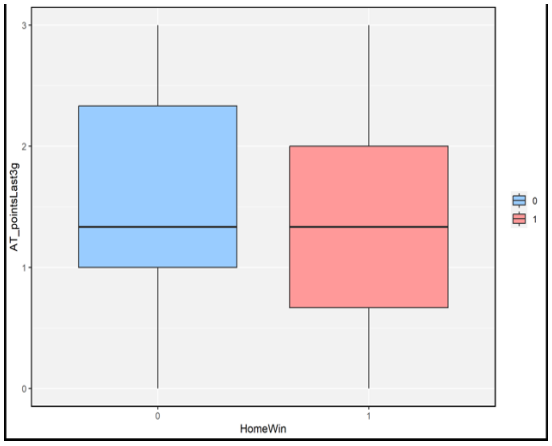


As a result of the analysis, variables that may have limited predictive ability were identified. These variables are: HT_pointsLast3g, AT_pointsLast3g, and h2h_diff..

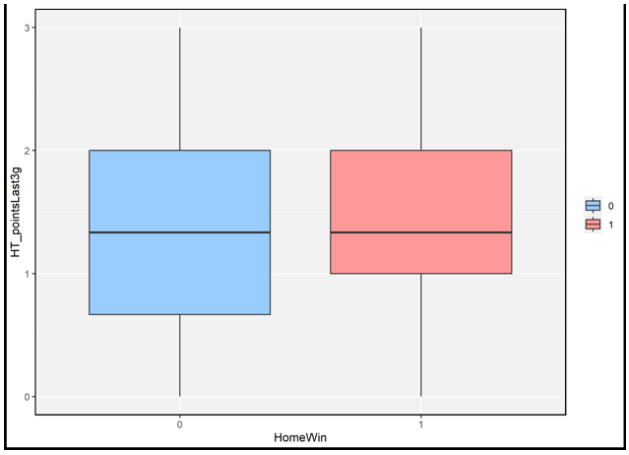
Boxplot of h2h_diff.



Boxplot of AT_pointsLast3g.



Boxplot of HT_pointslast3g.



The variables HT_pointsLast3g and AT_pointsLast3g will be excluded from further analysis and will not be considered in the modeling process. However, due to the presence of a significant number of missing data in the variable h2h_diff, it will be subjected to further observations. To fill in these missing data, various methods will be implemented, which will allow for a more accurate understanding of the characteristics of this variable and its potential impact on the analyzed process.

As a result of the above analysis, the basic set of input data was selected:

- HT_pointsHLast3g,
- AT_pointsALast3g,
- h2h_diff,
- HT_ShotsOTLast5g,
- AT_ShotsOTLast5g,
- HT_Goals_Last5g,
- AT_Goals_Last5g,
- HT_GoalsConLast5,
- AT_GoalsConLast5,
- HomeTeam_Value,
- AwayTeam_Value,
- HT_AvgPointsCurrentS,
- AT_AvgPointsCurrentS,

3.7. Source code presentation

In order to create the code, the R programming language was used, in the RStudio environment.

3.7.1 Decision Tree

In addition to the built-in Rstudio libraries, additional libraries were used to create the classification tree model:

Rpart() –Create, train, and pruning a model,

Mlr()- Model creation and parameter tuning,

Caret() – model evaluation, generation of confusion matrix,

Rpart.plot() – Generating tree charts,

pROC()- to calculate the ROC curve and generate the plot,

First, the data were loaded and split into a training set and a test set in a ratio of 70% (1929 observations) to 30% (817 observations). This was done using the `sample()` function, which randomly assigns indices to the data in the appropriate proportions. To ensure the results are repeatable, a random seed was set.

```
set.seed(123)
ind <- sample(2, nrow(dane_tree1), replace = TRUE, prob = c(0.7, 0.3))
train_set <- dane_tree1[ind==1,]
test_set <- dane_tree1[ind==2,]
```

Next, a basic decision tree was created using the initial dataset and default parameter values.

```
tree1 = rpart(HomeWin ~ .,
              data=train_set,
              method = 'class')
```

Default values for decision tree parameters.

Parameter	Value
Minsplit	20
Minbucket	-
Cp	0,01
Maxcompete	4
Maxsurrogate	5
Usesurrogate	2
Surrogatestyle	0
Maxdepth	30
Xval	10
Kryterium podziału	Gini

A graph showing the diagram of the resulting tree was also generated and its ability to predict on a test set was checked.

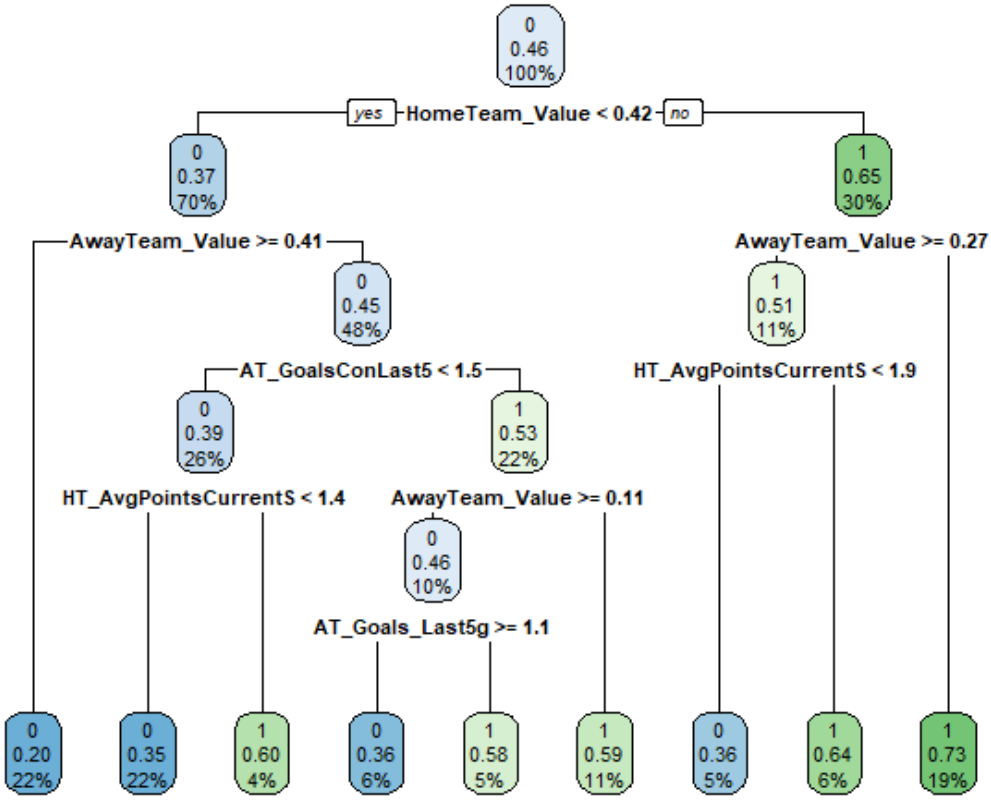
Source code for predictions for model 1.

```
rpart.plot(tree1)

predicted <- predict(tree1, train_set, type = 'class')
confusionMatrix(predicted,train_set$HomeWin)

predicted1 <- predict(tree1, test_set, type = 'class')
confusionMatrix(predicted1,test_set$HomeWin)
```

Graphical representation of the model 1.



.

To check for potential overfitting at this stage, the obtained accuracy was compared with the prediction accuracy on the training set, which was 68.48%. A difference of only 4 percentage points does not suggest excessive overfitting of the model. The next step was to fine-tune the appropriate model parameters. Particular attention was paid to the following parameters:

Maxdepth – The "maxdepth" parameter regulates the maximum depth of the tree, in other words, it specifies how many levels the tree can have from the root before its growth is halted. Choosing a large value for maxdepth leads to generating a complex tree that captures the relationships in the training data well, but this may weaken the tree's ability to generalize to new data. Conversely, selecting a too small value may risk underfitting the model.

Minsplit – The "minsplit" parameter defines the minimum number of observations that must be in a node for a split to occur. If the number of observations in a node does not

exceed the minsplit value, the split will not be performed, and the node will become a leaf.

Cp – The complexity parameter "cp" controls the size of the tree and prunes unnecessary branches. The higher the cp value, the more restrictive the pruning criteria. Initially, when building the tree, the goal is to create a more complex model that considers as many relationships in the training data as possible, so the initial cp value is set to 0. The optimal value of this parameter will be chosen during the tree pruning stage.

All other parameters were kept at their default values.

To fine-tune the model parameters, a classification task object was created, the cross-validation method was specified, the metric for model verification was determined, and a parameter grid to be tested was defined. As a result of the parameter grid search, all combinations of maxdepth (1:10), cp = 0, minsplit (1:30) were tested with 10-fold cross-validation.

Source code for the hyperparameter tuning process.

```
getParamSet("classif.rpart")
tree2 <- makeClassifTask(
  data=train_set,
  target="HomeWin"
)

control_grid = makeTuneControlGrid()
resample = makeResampleDesc("CV",iter = 5,predict = "both")
measure = acc

param_grid <- makeParamSet(
  makeDiscreteParam("maxdepth", values=1:10),
  makeDiscreteParam("cp", values = 0),
  makeDiscreteParam("minsplit", values=1:30),
  makeDiscreteParam('xval',value =10)
)
```

As a result of the above code, the best set of parameters was determined, for which predictions were made on the test set.

```
best_params = setHyperPars(
  makeLearner("classif.rpart", predict.type = "prob"),
  par.vals = dt_tuneparam_multi$x
)

best_model_multi <- mlr::train(best_params, tree2)
best_tree_model <- best_model_multi$learner.model
rpart.plot(best_tree_model)
predicted2 <- predict(best_tree_model, newdata = test_set, type = "class")
confusionMatrix(predicted2, test_set$HomeWin)
```

At this stage, individual variables that were initially not utilized were also added. If adding a particular variable improved accuracy, it was included in the model. However, if the addition of a variable had no impact on or weakened the model's predictive capability, such a variable was excluded. The combination of variables that yielded the highest accuracy was the basic set of variables augmented with the variable HT_AvgGoalsHLastS.

After examining the weights for individual variables in the model, it was decided to remove the variables HT_pointsHlast3g, AT_pointsAlast3g, and HT_ShotsOTLast5g as they did not contribute to the model. The parameter tuning process resulted in the following parameter values:

Maxdepth = 5,

Minsplit = 27,

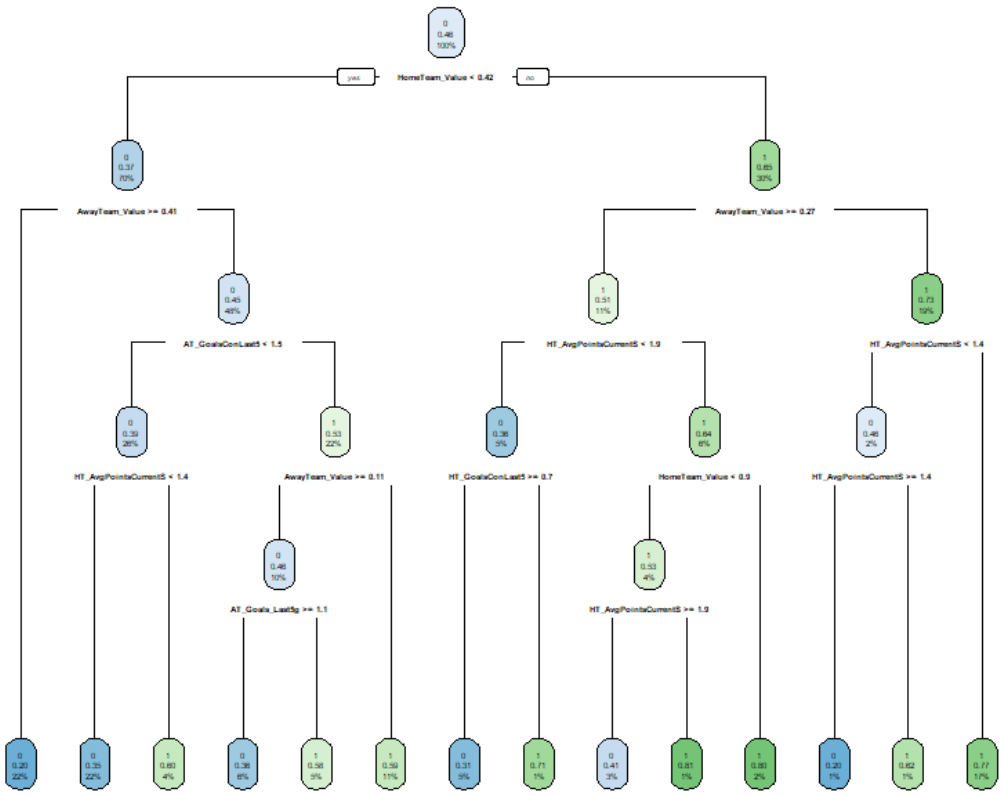
The impact of each variable on the decision-making process was also examined. The variable that proved to be the most significant was the value of the away team. Other important variables related to the current season's form of the teams and the value of the home team. In addition to the previously mentioned variables, the ones with the least importance in the model were HT_GoalsConLast5 and HT_Goals_Last5g

.

Model 3 parameter weights.

Zmienna	Waga
AwayTeam_Value	119,697
AT_AvgPointsCurrentS	89,4049
HT_AvgPointsCurrentS	88,79269
HomeTeam_Value	77,02568
HT_AvgGoalsHLastS	74,88642
AT_Goals_Last5g	37,76713
AT_GoalsConLast5	36,45533
AT_ShotsOTLast5g	30,03777
h2h_diff	13,07669
HT_GoalsConLast5	7,285428
HT_Goals_Last5g	3,288901

Graphical representation of model 2.



Next, the tree pruning process was applied using the `prune()` function and the optimal cp level (0.005113636) determined based on minimizing the cross-validation error..

Source code for the tree pruning process.

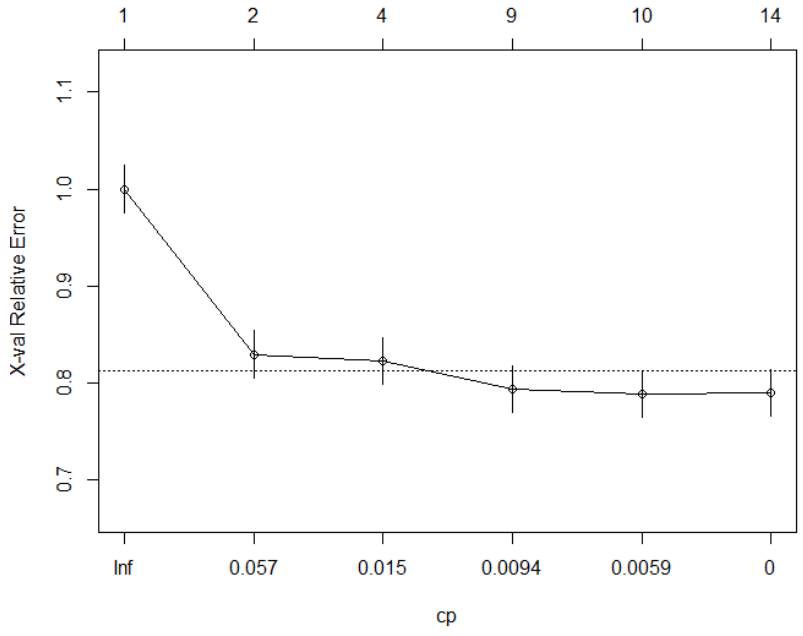
```
best_tree_model$cptable
best_cp <- best_tree_model$cptable[which.min(best_tree_model$cptable[,"xerror"]),"CP"]
plotcp(best_tree_model)
pruned_tree <- prune(best_tree_model, cp = best_cp)
print(pruned_tree)
rpart.plot(pruned_tree)
predicted3 <- predict(pruned_tree, newdata = test_set, type = "class")
confusionMatrix(predicted3, test_set$HomeWin)
```

Table of cp values for model 2.

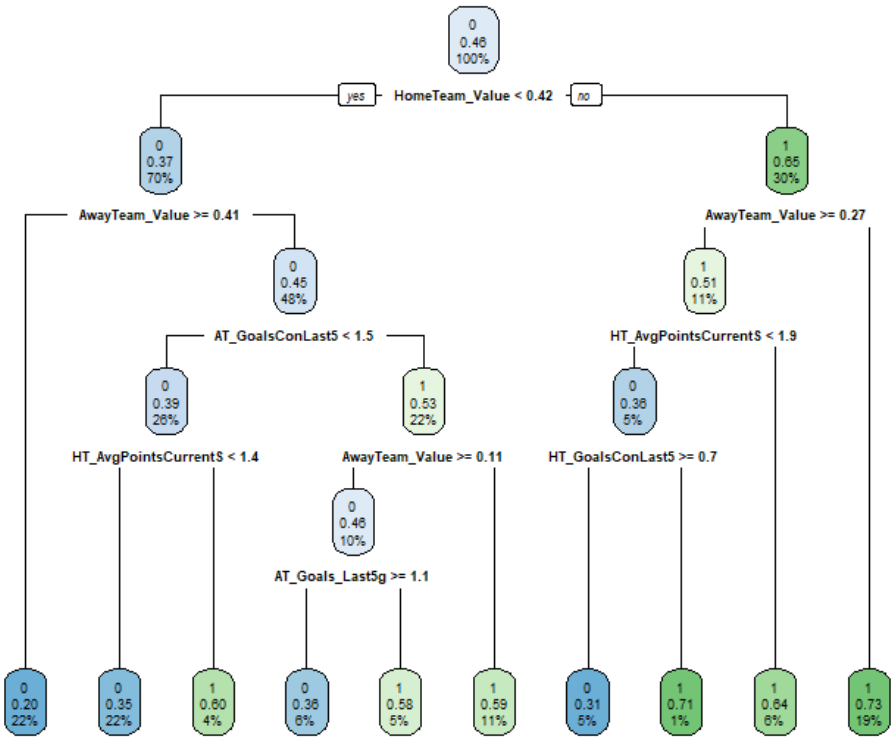
	cp	nsplit	rel error	xerror	xstd
1	0,195455	0	1	1	0,024859
2	0,016477	1	0,804545	0,829545	0,024206
3	0,013068	3	0,771591	0,826136	0,024186
4	0,006818	8	0,690909	0,796591	0,024005
5	0,005114	9	0,684091	0,781818	0,023907
6	0	13	0,663636	0,782955	0,023915

In order to better understand the relationship of the cross-classification error to cp and tree size, a graph has been generated.

Graph of the relationship between cp and tree size and cross-validation error.



Graphical representation of model 3.



The model building process ends with verification of the quality of the trimmed tree.

3.7.2 Random forest

Another method used was the random forest algorithm. The following functions were used to create the model:

Ranger() – building, training the model and tuning parameters

Caret()- evaluation of the quality of the model, generation of the confusion matrix

pROC()- generating an ROC curve

ggplot2()- charts

During the model training process, the data was split according to the previous approach, and then a basic model was created that had default parameter values.

Source code for model creation and prediction.

```
model <- ranger(HomeWin ~ .,
                data = train_set)
p1 <- predict(model, train_set)
confusionMatrix(p1$predictions, train_set$HomeWin)
p2 <- predict(model, test_set)
confusionMatrix(p2$predictions, test_set$HomeWin)
```

This model achieved 100% accuracy on the training set and 65.85% accuracy on the test set. Such high accuracy on the training set suggests that the model is overfitted to these data, so it is necessary to select appropriate parameter values. When building random forests, there are three key parameters to consider:

1. Number of Trees (num.trees): Specifies the number of trees in the forest. Generally, a larger number of trees provides better results, but training such a model consumes more computational power.

2. Number of Features (mtry): In each node, the random forest algorithm randomly selects a specified number of variables (without replacement) determined by mtry. The best of these variables is used to make the split. By default, mtry is set to the square root of the number of variables. A large mtry value introduces less randomness into the model because variables with high predictive value are more frequently included in the selected variables. This can result in most trees in the forest being similar to each other. Conversely, a small mtry value can lead to very poor predictive power in some trees.

3. Maximum Depth (max.depth): Controls the maximum depth of each tree in the forest. The default value is set to 0, which means trees can grow to the maximum possible extent allowed by the data set. However, a model with a low max.depth value is more prone to overfitting. Training such a model also takes more time.

The first parameter checked was the number of trees in the forest. The ranger function and a loop were used to examine how the prediction error behaves with different values of num.trees. The number of trees was tested in the range from 50 to 1000. The mtry value was set to 4, and the Gini index was used as the splitting criterion. Then, a plot was generated to show the results of the loop.

Source code for tree count selection.

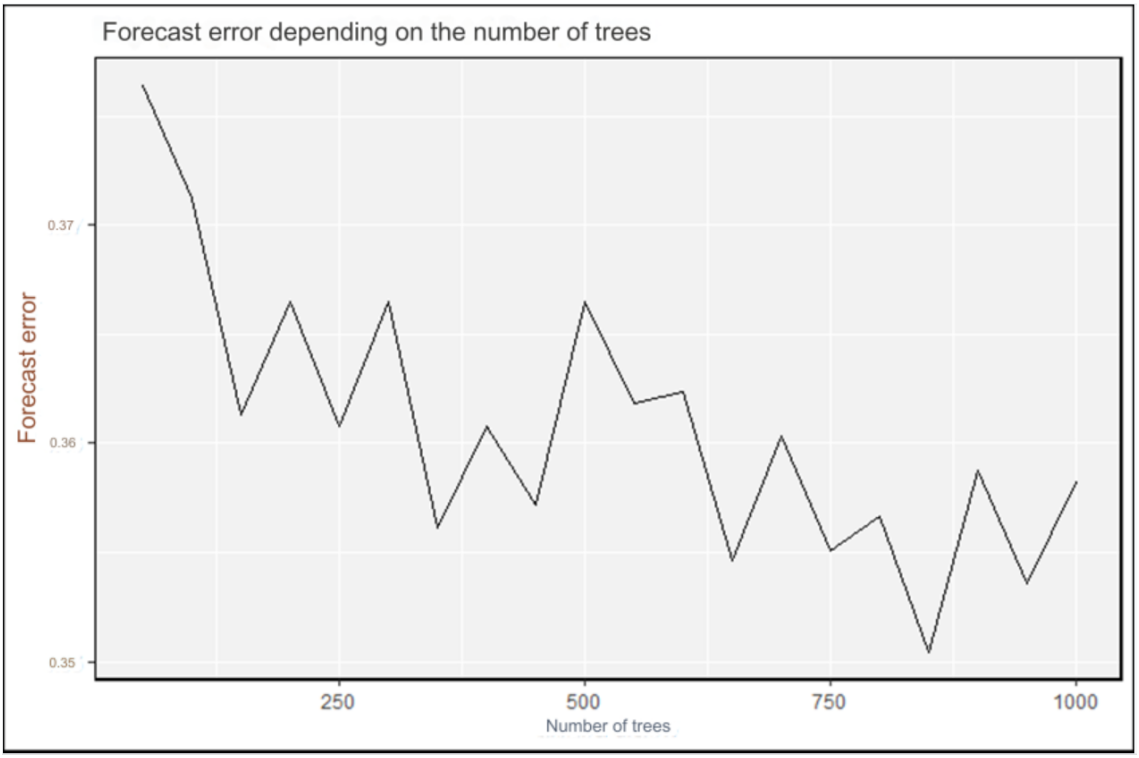
```r
num.trees <- seq(50, 1000, by = 50)
oob.error <- vector("numeric", length(num.trees))

for (i in seq_along(num.trees)) {
  model <- ranger(HomeWin ~ .,
                  data = train_set,
                  num.trees = num.trees[i],
                  mtry = 4,
                  importance = 'impurity')
  oob.error[i] <- model$prediction.error
}

df <- data.frame(NumTrees = num.trees, OOBE = oob.error)
ggplot(df, aes(x = NumTrees, y = OOBE)) +
  geom_line() +
  labs(title = "OOB Error vs Liczba drzew", x = "Liczba drzew", y = "OOB Error")+
  theme(panel.border = element_rect(color = 'black',fill = NA,size = 1),
        panel.background = element_rect(fill='gray95'),
        plot.background = element_rect(color = 'black', size = 1),
        axis.title.x = element_blank(),
        axis.title.y = element_blank())

min.error.index <- which.min(oob.error)
best.num.trees <- num.trees[min.error.index]
```

Forecast error plot relative to tree count.



The next parameters that needed to be tuned were mtry and max.depth. For this purpose, a grid of variables was created, containing all combinations of mtry from 1 to 10 and max.depth from 1 to 15. The parameter values were checked in a manner similar to the number of trees. The num.trees value was set to the best value determined in the previous step..

Source code for creating and applying a parameter grid.

```
tune_grid <- expand.grid(max.depth = seq(1, 15, by = 1),
                         mtry.size = seq(1, 10, by = 1))

for(i in 1:nrow(tune_grid)){
  max_depth_value <- tune_grid$max.depth[i]
  mtry_value <- tune_grid$mtry.size[i]

  model <- ranger(HomeWin ~ .,
                  data = train_set,
                  num.trees = best.num.trees,
                  mtry = mtry_value,
                  max.depth = max_depth_value,
                  importance = 'impurity')
  oob.error[i] <- model$prediction.error
}

min.error.index <- which.min(oob.error)
best_params <- tune_grid[min.error.index,]
```

The last stage was to build a random forest using previously selected parameters, and evaluate the model.

Source code for the application of the random forest model.

```
model <- ranger(HomeWin ~ .,
                data = train_set,
                num.trees = best.num.trees,
                mtry = best_params$mtry.size,
                max.depth = best_params$max.depth,
                importance = 'impurity')

print(model)
attributes(model)
p3 <- predict(model, train_set)
confusionMatrix(p3$predictions, train_set$HomeWin)
p4 <- predict(model, test_set)
confusionMatrix(p4$predictions, test_set$HomeWin)
```

Again, the process began by testing the model on the basic dataset. Then, additional variables were added, and the quality of the prediction was verified. The model with the highest accuracy included the basic set of variables augmented with

HT_AvgGoalsHLastS and AT_ShotsLast5g. The values selected during the tuning process are presented in the table below.

Parameter values for the model 4.

Parameter	Value
Num.trees	850
Mtry	4
Max.depth	7

Weights of variables in the model 4.

Variable	Weight
HT_AvgPointsCurrentS	45,73125
HomeTeam_Value	45,57244
AwayTeam_Value	45,02436
AT_AvgPointsCurrentS	37,0539
HT_AvgGoalsHLastS	32,03984
AT_ShotsLast5g	26,01121
HT_ShotsOTLast5g	22,16368
h2h_diff	20,6538
AT_GoalsConLast5	19,06619
AT_ShotsOTLast5g	19,03663
AT_Goals_Last5g	14,22887
HT_GoalsConLast5	13,3657
HT_Goals_Last5g	13,24854
HT_pointsHLast3g	9,746709
AT_pointsALast3g	9,224401

We can see that the weights of the variables are more similar to each other than in the case of a decision tree algorithm. Three variables were of greatest importance to the model: HT_AvgPointsCurrentS, HomeTeam_Value, AwayTeam_Value.

3.7.3. Extreme Gradient boosting

The last algorithm used is the XGBoost (Extreme Gradient boosting).

The following functions were used to create the model:

Xboost() – build and train a model,

Mlr() – Tuning model hyperparameters,

Caret() – Model quality,

Description of the parameters used in the model XGBoost:

Nrounds: Specifies the number of boosting rounds. In each round, a new base model is created.

Eta: This is the learning rate, responsible for adjusting the weights in subsequent models. A low value means smaller adjustments, which can help with overfitting but lengthens the model training process.

Max.depth: Controls the depth of each tree.

Subsample: Controls the number of observations passed to each tree. A lower value means greater randomness in selecting observations.

Colsample_bytree: Specifies the number of variables passed to each tree. Like subsample, a value less than 1 indicates greater randomness.

Min.child.weight: If the sum of the weights of the "children" is less than this parameter, the node is not created.

At the beginning, the data was divided into a test and training set in the same way as in the previous examples. Both collections have been converted to the format xgb.DMatrix and data labels have been extracted.

Source code for the division into training and test sets.

```
set.seed(123)
ind <- sample(2, nrow(danexboost), replace = TRUE, prob = c(0.7, 0.3))
train_set <- danexboost[ind==1,]
test_set <- danexboost[ind==2,]

dtrain <- xgb.DMatrix(data = as.matrix(train_set[, -length(test_set)]), label = train_set[,length(test_set)])
dtest <- xgb.DMatrix(data = as.matrix(test_set[, -length(test_set)]), label = test_set[,length(test_set)])
```

Then, the parameters of the goal function and evaluation metrics were determined, and a list of objects containing the training and test sets was created.

Source code for specifying parameters.

```
xgb_params <- list("objective" = "binary:logistic",
                   "eval_metric" = "logloss"
                   )
watchlist <- list(train = dtrain, test = dtest)
```

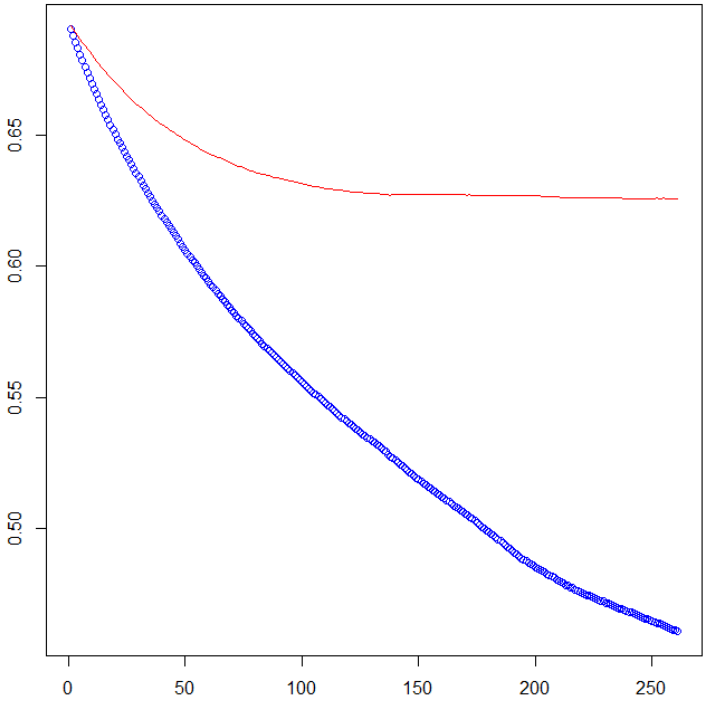
The next step was to create a model with basic parameter values and, based on it, determine the appropriate values for the number of boosting rounds (nrounds) and eta. This selection was based on minimizing the logarithmic logistic error. The table below shows the initial parameter values used to start training the model. To ensure reproducibility of the results in the model, a random seed was set.

Initial parameter values.

Parameter	Value
Max.depth	6
Subsample	0,8
Colsample_bytree	1
Min.child.weight	1
Nrounds	500
Eta	0,1

In order to better understand the relationship between the eta and nround values, a graph was generated showing how the logarithmic logistic error changes in the test (red) and training (blue) sets.

A graph of the distribution of error on the training and test set.



In order to determine the optimal nround value for the determined eta level, cross-validation was performed using the xgb.cv function.

Source code for selecting nround values.

```
params <- list(booster = "gbtree", objective = "binary:logistic", eta=0.01,
               gamma=0, max_depth=6, min_child_weight=1, subsample=0.8, colsample_bytree=1)

xgbcv <- xgb.cv( params = params, data = xgtrain, nrounds = 500, nfold = 5,
                 showsd = T, stratified = T, print_every_n = 10, early_stop_round = 20, maximize = F)


xgbcv$evaluation_log[xgbcv$evaluation_log$test_logloss_mean==min(xgbcv$evaluation_log$test_logloss_mean),]
```

Then, the basic model was verified on the test and training data.

Source code for model validation.

```
preds <- predict(xg_model, xgtest)
preds <- ifelse(preds > 0.5, 1, 0)
preds <- as.factor(preds)
x <- as.factor(test_set[,length(test_set)])

table(pred = preds, true = test_set[,length(test_set)])
confusionMatrix(preds,x)

preds <- predict(xg_model, xgtrain)
preds <- ifelse(preds > 0.5, 1, 0)
preds <- as.factor(preds)
x <- as.factor(train_set[,length(train_set)])
table(pred = preds, true = train_set[,length(train_set)])
confusionMatrix(preds,x)
```

In the next step, hyperparameter optimization was carried out using the **mlr** package. For this purpose, classification tasks were created based on the training and test data, and a learning object utilizing XGBoost classification algorithms was set up. Within the learning object, the previously determined values for nrounds and eta were also defined. Additionally, a parameter grid was configured to identify the optimal hyperparameter values. This grid contained various combinations of parameters to be tested for their impact on the model's quality. Cross-validation was used to evaluate and compare the model's results..

Source code for hyperparameter selection.

```
test_set$HomeWin <- as.factor(test_set$HomeWin)
train_set$HomeWin <- as.factor(train_set$HomeWin)
traintask <- makeClassifTask (data = train_set,target = "HomeWin")
testtask <- makeClassifTask (data = test_set,target = "HomeWin")

lrn <- makeLearner("classif.xgboost",predict.type = "response")
lrn$par.vals <- list( objective="binary:logistic", eval_metric="error", nrounds=271, eta=0.01,set.seed(123))


params <- makeParamSet(makeIntegerParam("max_depth",lower = 3L,upper = 10L),
                       makeNumericParam("min_child_weight",lower = 1L,upper = 10L),
                       makeNumericParam("subsample",lower = 0.5,upper = 1),
                       makeNumericParam("colsample_bytree",lower = 0.5,upper = 1))


resample <- makeResampleDesc("CV",stratify = T,iters=5L)
ctrl <- makeTuneControlRandom(maxit = 10L)
tune_params <- tuneParams(learner = lrn, task = traintask,
                     resampling = resample, measures = acc,
                     par.set = params ,control = ctrl,show.info = T)

tune_params$y
```

The final step involved assigning the determined parameter values to the model, followed by lowering the eta value while simultaneously increasing the nrounds value, and testing different data sets. Below is a table showing the distribution of variable importance weights in the model.

Weights of variables in the model 5.

Variable	Weight
AwayTeam_Value	0,18372917
HT_AvgPointsCurrentS	0,16975184
HomeTeam_Value	0,14523383
AT_AvgPointsCurrentS	0,10601171
HT_AvgGoalsHLastS	0,09426241
HT_ShotsOTLast5g	0,07103434
AT_GoalsConLast5	0,05761435

h2h_diff	0,04167142
AT_ShotsOTLast5g	0,03987216
HT_Goals_Last5g	0,02431244
AT_Goals_Last5g	0,02328586
HT_pointsHLast3g	0,01859008
HT_GoalsConLast5	0,01494303
AT_pointsALast3g	0,00968736

Again, the best results were achieved with the primary dataset plus the variable HT_AvgGoalsHLastS.

4. Presentation of results

Model 1 – A model using a decision tree algorithm created from a basic set of variables and default parameter values.

Model 2 – model 1 with selected hyperparameter values.

Model 3 – model 2 after applying the tree trimming process.

Model 4 – model using the random forest algorithm with the parameters presented in the table above.

Models 5,6,7 have been created on the basis of the XGBoost algorithm, their parameters are presented in the table below.

Model hyperparameters XGBoost.

	Model 5	Model 6	Model 7
Booster	gbtree	gbtree	gbtree
Nrounds	547	134	152
Eta	0,01	0,04	0,03
Max depth	4	3	3
Subsample	0,842	0,704	0,704
Colsample by tree	0,764	0,992	0,992
Min child weight	1	4,98	4,98

Confusion matrix model 1.

Predicted value	Actual value	
	1	0
1	232	145
0	142	298

Confusion matrix model 2.

Predicted value	Actual value	
	1	0
1	216	131
0	158	312

Confusion matrix model 3.

Predicted value	Actual value	
	1	0
1	235	147
0	139	296

Confusion matrix model 4.

Predicted value	Actual value	
	1	0
1	181	79
0	193	364

Confusion matrix model 5.

Predicted value	Actual value	
	1	0
1	205	103
0	169	340

Confusion matrix model 6.

Predicted value	Actual value	
	1	0
1	207	106
0	167	337

Confusion matrix model 7.

Predicted value	Actual value	
	1	0
1	211	103
0	163	340

Comparison of measures calculated from the error matrix.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	64,87%	64,63%	64,99%	66,71%	66,71%	66,59%	67,44%
Precision	61,54%	62,24%	61,52%	69,61%	66,56%	66,13%	67,2%
Sensitivity	67,27%	70,43%	66,82%	82,17%	76,75%	76,07%	76,75%
Specificity	62,03%	57,75%	62,83%	48,40%	54,81%	55,35%	56,42%
Classification Error	35,13%	35,37%	35,01%	33,29%	33,29%	33,41%	32,56%
AUC	69,64%	67,44%	69,59%	69,59%	72,1%	71,49%	71,54%

Profitability of models based on bookmakers' odds.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Profit/Loss	-3051,834	-2966,27	-2813,834	-520,227	2034,916	1764,577	-542,6445
% Profitability	-9,54%	-9,27%	-8,79%	-1,63%	6,36%	5,51%	-1,70%

4.1 Overview of the results

In addition to the classical measures of model quality assessment, the predictive ability of each model was tested on 320 matches from the 2022/2023 English Premier League season. Due to missing data for the initial rounds, data from the first six rounds were not used for prediction. The profit of the model represents the sum obtained by betting on each match with an amount of 100 units. In the case of a correct prediction, the profit from a single match is calculated according to the formula:

$$profit = odds * amount - amount$$

In the case of an incorrect prediction, a loss equal to the staked amount is recorded. The profitability percentage is calculated as the sum of profit divided by the total invested amount. In the examined discipline, the best experts achieve profitability levels of 5%-15%. As expected, the simplest decision tree algorithms achieved the weakest results. Interestingly, parameter tuning in this case only slightly improved both accuracy and profitability. The random forest algorithm achieved the same accuracy as the fifth model created using the XGBoost algorithm, but the profitability of the two models differed significantly, with model 5 yielding a profit of 6.36%, the best result among all applied models.

Comparing the confusion matrices of all models reveals differences in the tendency to predict outcomes. Models based on decision tree algorithms predicted a home win in an average of 45% of cases, while the random forest and XGBoost algorithms tended to

favor the absence of a home win. Model 4 predicted this outcome in 68.18% of cases, successfully forecasting the highest percentage of all matches ending with such a result (82.17%). However, this did not translate into a higher profit. Model 5, on the other hand, predicted the absence of a home win in 63% of cases, with 66.56% being correct. Since models 4-7 achieved much better profitability results, it can be assumed that a good approach is to place more emphasis on correctly predicting matches ending in an away team win or a draw.

Model 7 achieved the highest accuracy among all developed models. Nevertheless, it is worth noting that despite its high accuracy, this model showed negative profitability. This may suggest that the model correctly predicted the outcomes of more matches with a clear favorite, which is associated with lower betting odds. The same issue likely affected the decision tree algorithm. Model 3 correctly predicted the highest percentage of home wins; however, an analysis of the tree structure and variable importance indicates that it considered variables characterizing match favorites, such as team value and average points scored in the current season, to a much greater extent than other models..

4.2. Summary and steps for the future

Predicting the outcomes of football matches is challenging due to the large number of independent factors that influence the final result. However, the obtained results provide a basis for asserting that models based on random forests and publicly available data can assist in accurately predicting football matches. The most promising results were obtained using the XGBoost technique, where the best model achieved an accuracy of 66.71%. It is worth noting that this model yielded a profit of 6.36%, which is a satisfactory result considering that the typical return rate for experts in this field ranges from 5% to 15%.

However, since not all models created using this method were profitable, it would be worthwhile in the future to enhance the model with a larger number of variables such as injuries to key players, more detailed match statistics, and expert opinions..