Prediction of the match results from the 1st German Soccer League

Dennis Gubenko

Patrick Nitzsche

Management Information Systems Department

San Diego State University

MIS/BA 749: Business Analytics

Dr. Aaron C. Elkins

December 14, 2021

Table of Contents

1 Executive Summary 1

2 Discovery and Data Preparation 5

3 Model Planning and Building 11

4 Results and Performance 13

5 Discussion and Recommendations 16

References 17

Appendix 18

R-Code 21

1 Executive Summary

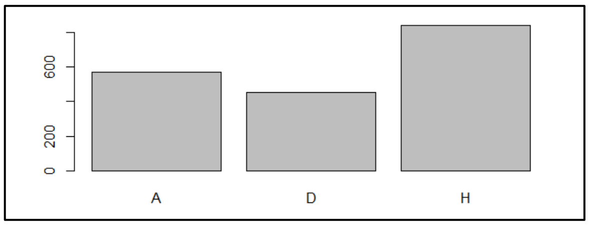
The sports betting market is very competitive and there are many different bookmakers on the market who generate good profit from the users. That means they pay out less for won bets than they earn with the stacks for lost bets over the long run. There are two reasons why this system works: The bookmakers can predict the match outcome better than the average user and they balance the odds in a way that makes the user want to bet on an outcome but still generates profit for the bookmakers. This is not possible with only human expertise. They use real time data analytics instead to automate and optimize the odd-building.

The goal of this project is to imitate the machine learning algorithm of the bookmakers by trying to predict the match outcomes of a whole football season. We don't aim to build the odds (which would just be 1/probability of each outcome), but to make predictions for the match outcomes (home win, draw, away win) and check, if this can result in a betting strategy, that can generate profit for us. In other words, we want to check how good our own predictions are compared to the predictions of the bookmakers (which are reflected in the odds). If we bet $1 on the specific odd of every predicted match outcome and we end up with a profit, it means that our predictions are better than the bookmakers and we can use the model to earn money with sports betting. This will probably not be possible as bookmakers do have bigger data, deeper information, and data sources, we don't have access to. We still want to see how close we can come to the data analysis of the bookmakers with our own machine learning model.

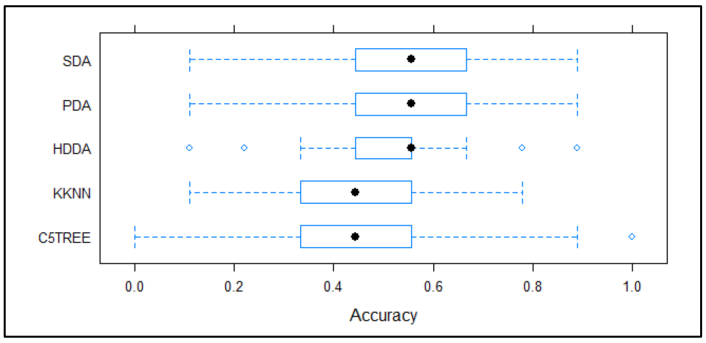
For our project we will use the historical data of 6 consecutive seasons (2014/15 – 2019/20) of the 1st German Bundesliga to build a prediction model for single match outcomes, which results in a multiclass classification problem. To test our model, we will use the 2020/21 season and see how good our prediction model performs on unseen data. Our data includes a total of 2,142 matches (18 teams in the league, 9 games per matchday, 34 matchdays per season, 7 seasons), where each row represents a single match. The columns in the dataset were filtered to only those we can be potentially used for the further analysis or preprocessing. As the stats are from the current games that we don't know

before the specific game is played, we will calculate means from these columns of previous games. We distinguish between the last 6 games of a team and the entire past of a team, as well as between all games and only the ones against the specific opponent. The listing of each variable is shown in the table in Appendix 1.

First, we do a feature selection in which the multicollinearity is tested by identifying and sorting out highly correlated predictors. Then the recursive feature selection takes place, where the dataset is searched for the most relevant features that best describe the predictor variable using a 10-fold cross validation. We end up with only 12 variables left out of the initial 64. Before we go to the model planning and building, we will look on the distribution of our response variable “FTR”:



It is shown that home wins are the most common result, whereas draws are relatively rare. We should keep that class imbalance in mind for our upcoming model building. The data is split up into a training and test set, while the test set contains the last season (306 rows) and the rest is the training set (1558 rows). Before we dive into the model building, we set up the control function. As this is a timeseries dataset, where the date of a game matters, we can’t simply do a cross-validation approach. The reason for that is that we can’t take a game from 2020 to predict a 2016 game, as this would not reflect the reality of sports betting. Instead, we use the “timeslice”-method, where the first 306 rows are used as the training set and the next 9 games as the validation set. We will train the following 5 multiclass models to see which one performs the best: K-nearest Neighbors, Penalized Discriminant Analysis, Shrinkage Discriminant Analysis, High Dimensional, Discriminant Analysis, Single C5.0 Tree. While the Shrinkage Discriminant Analysis, Penalized Discriminant Analysis, and High Dimensional Discriminant Analysis models have a similar mean performance of 56 %, the SDA has the least spread and therefore should provide more constant predictions.



We then used this model to predict the match outcomes of the test set and ended up with the following confusion matrix:

		Reference		
		A	D	H
Prediction	A	45	1	53
	D	19	0	56
	H	20	0	112

The model is good at predicting home wins (85 % accuracy) but is bad at predicting away wins (44 %) and is not able to predict draws at all (0 %). By comparing our overall accuracy (51 %) to the accuracy of the bookmaker predictions (33%), we still have comparably good results. To test the real value of our prediction accuracy, we want to test our betting strategy by betting $1 on every predicted match outcome and see, how much will be left on our bank account.

We lost $13.83 over the season 2020/21, so the model is not able to generate profit for us on the betting market. As our model seems to predict home wins better than the other results, we will test another betting strategy, by only betting on home win predictions. By doing that we still end up losing $7.54. The reason for that is that the average home win odds are lower (2.2) than the draw odds (4.3) and away win odds (5.0), which means that a won bet on home wins results in less profit than on other results.

The conclusion of our project is that we are still far away from competing with the sports betting industry. It is also hard to compare our model to the odds, as they build their models not to predict the games right, but to generate profit with the odds. They also include the betting behavior of the users to build the optimal odds for them. Even though we lost money with that method, it wasn't much considered the fact that we put in a total of $306 in stacks and only lost $15.89/$8.54. To improve our model, we should also exclude the option to predict draws, as our model totally failed to predict those. Using only two class classification models, we have more models to try and therefore maximize the chance of finding the optimal model for our problem. By balancing out the class imbalance with an up- or down sampling method, we should expect even better results.

2 Discovery and Data Preparation

The goal of this project is to predict the match results of soccer matches in the 1st Bundesliga in Germany from the season 2020/2021 as accurately as possible. Specifically, we want to predict whether the home team will win, the away team will win, or whether the game will end in a draw by using history data from the seasons 2014/2015 to 2019/2020 to achieve better results than the betting providers. As there are three different possible outcomes, it’s a multiclass classification. Each line in the data set represents a single match in the 1st soccer Bundesliga, and since there are 18 teams in the league which are playing against each at in 34 match days, there are a total of 306 matches per game, resulting in 306 observations per year.

There are several providers of databases on European soccer matches on the Internet with different contents. The dataset from Football Data UK is particularly suitable for our project for various reasons. On the one hand, individual datasets exist for each Bundesliga season, which is why a train dataset, and a test dataset can be created easily. The seasons from 2014/2015 to 2019/2020 are used to train the data, the test set consists of the season 2021/2022. Second, the datasets have a good information content and already contain some of the variables needed for the project. In addition, the data provide an essential contribution for the creation of the required variables, so further variables can be derived from the data. Furthermore, another advantage is that there are no missing values in the data set, so the data quality is pretty good. The individual data sets are in CSV format and can therefore be easily read into R.

First, the goal is to create the overall data set from the data sets of each single season. Since not all columns are needed, a vector with the required columns is created first. Because in the last seasons other column labels were used and additional data were added than at the beginning, a different vector must be applied for these periods. In addition, the columns are transferred to a uniform labeling. The individual data records are then merged into one data set and individual data sets are removed after this. The data set now contains 2,142 observations and 21 total columns. When checking the data for missing values, there are no missing data.

The next step is to create a function to generate the variables, which are needed to predict the outcome of the game. A distinction is made between:

- data for the home team and data for the away team
- the current form of the team in the last 6 games and the past games in total
- all matches and matches against a specific opponent

The goal is to provide the current form of the team by generating variables for the last 6 games of the respective teams and to include the form against the respective opponent and the overall form. On the other hand, variables are generated, which give information about the complete past games. This is done in 2 steps, first the respective data (goals, shots, cards, etc.) are added together, before sums or averages are calculated from them. In the form of the last 6 games, it should be noted that there are not enough past games at the beginning of the data set, so this data is omitted. Yellow and red cards are first considered separately and converted into a point system. Yellow cards are calculated with 10 points, red cards with 25 points, to be able to calculate a variable for the cards at the end for the total number of points of cards per game, thus yellow cards and red cards cumulated. For points, the same is added up and divided by the number of games to get an average number of points per game. Goals scored, goals conceded, shots and shots on goal, corners and fouls are also averaged per match. The steps are executed 2 times, once for the respective home team and once for the respective away team. This step contains all variables, which are shown in Appendix 1.

As for some matches or teams, there are no previous games (for example in the first season or for promoted teams), so there are no statistics to be computed and we just delete these games from our dataset. We end up with 1864 observations out of the initial 2142.

For the next steps it is necessary to convert the created columns into numeric columns and to round the numbers to 2 decimal places for clarity. Lines in which data are missing are also removed and are not considered in the following calculations. Columns that were necessary for the generation of the variables but are no longer needed are then sorted out. Feature selection now follows, testing for multicollinearity by identifying and sorting out highly correlated predictors (threshold = 0.75).

The following 28 variables are subsequently excluded:

[1] "HT_card_points_form_opp"

[2] "AT_Mean_card_points"

[3] "HT_shots_form_opp"

[4] "HT_Mean_card_points_opp"

[5] "AT_Mean_shots"

[6] "HT_Form_Points_opp"

[7] "AT_Form_Scored_opp"

[8] "AT_Mean_Conceded_opp"

[9] "AT_card_points_form"

[10] "AT_Mean_Points"

[11] "HT_Form_Conceded_opp"

[12] "HT_Mean_Points"

[13] "HT_Mean_shots"

[14] "AT_shots_form_opp"

[15] "HT_card_points_form"

[16] "HT_Form_Scored_opp"

[17] "AT_Form_Points_opp"

[18] "AT_Mean_Conceded"

[19] "AT_Form_Conceded_opp"

[20] "HT_Mean_Conceded"

[21] "HT_Mean_Scored_opp"

[22] "HT_Mean_Points_opp"

[23] "HT_Form_Points"

[24] "HT_shots_form"

[25] "HT_Mean_card_points"

[26] "AT_shots_form"

[27] "AT_corners_form_opp"

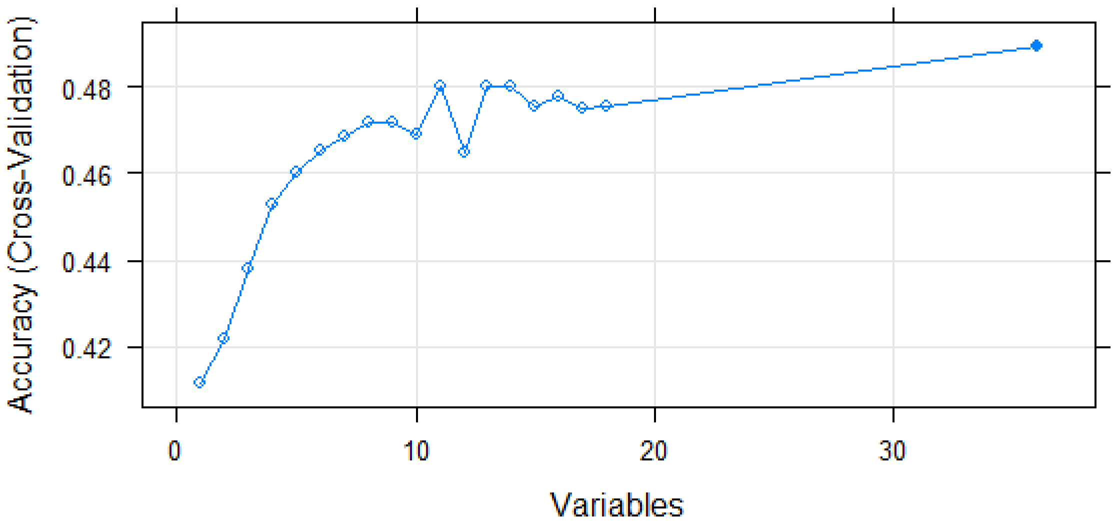
[28] "HT_fouls_form_opp"

Now the recursive feature selection takes place, where a seed of 101 is set, so that the result is reproducible. The recursive feature selection searches the data set for the most important features that describe the predictor variable the most. Up to 14 variables are to be retained, testing which variables achieve the highest accuracy. The result of the recursive feature elimination, including accuracy and kappa as well as the predictors are shown in the following tables:

Recursive feature selection (Outer resampling method: Cross-Validated 10-fold)
Resampling performance over subset size:

Variables	Accuracy	Kappa	AccuracySD	KappaSD
1	0.4115	0.05932	0.05574	0.08241
2	0.4217	0.08201	0.02646	0.03510
3	0.4378	0.10016	0.03374	0.05116
4	0.4528	0.12133	0.03133	0.05381
5	0.4603	0.12726	0.03044	0.05064
6	0.4651	0.12933	0.02054	0.03209
7	0.4684	0.13473	0.02039	0.03685
8	0.4716	0.13645	0.02770	0.04714
9	0.4716	0.13392	0.02683	0.04739
10	0.4689	0.13216	0.02776	0.04482
11	0.4802	0.14852	0.02044	0.03100
12	0.4646	0.12245	0.02813	0.04600
13	0.4802	0.14623	0.02430	0.04328
14	0.4801	0.14453	0.02522	0.04577
15	0.4754	0.13548	0.02328	0.03951
16	0.4775	0.14071	0.01890	0.03239
17	0.4748	0.13290	0.01997	0.03478
18	0.4754	0.13481	0.02314	0.03841
36	0.4893	0.15010	0.02464	0.04288

Predictors			
[1]	HT_Mean_shots_on_target	[2]	AT_Mean_Scored
[3]	HT_Mean_card_points	[4]	HT_Mean_Scored
[5]	AT_Mean_corners	[6]	AT_Mean_shots_on_target
[7]	AT_Mean_Scored_opp	[8]	HT_Mean_Conceded_opp
[9]	AT_Mean_shots_opp	[10]	AT_shots_on_target_form
[11]	HT_Form_Scored		



The graph shows that the accuracy increases sharply with the inclusion of the first variables up until 11 variables. After the accuracy goes down again and only exceeds the accuracy of 11 predictors with the inclusion of all 36 variables.

The next step is to split the dataset into a training and a testing set. As mentioned before, the goal is to predict the last season, which includes a total of 306 matches, while the training dataset consists of 1531 observations. The final dataset contains the following variables:

[1] "FTR"

[2] "AT_Mean_corners_opp"

[3] „AT_Mean_shots_opp"

[4] "HT_Mean_Scored"

[5] „AT_Mean_shots_on_target"

[6] "AT_Mean_Scored"

[7] „AT_Mean_corners"

[8] "AT_Mean_fouls"

[9] „HT_Mean_fouls"

[10] "HT_Mean_shots_on_target"

[11] "AT_Mean_fouls_opp"

3 Model Planning and Building

First, the descriptive statistics are performed. After evaluating the dependent variable FTR (final time result), it shows that in total 840 times the home team won, 573 times the away team won, and 451 games ended in a draw. That home teams perform better on average is normal in that they have the crowd and are more familiar with the conditions in their own stadium than the away team. The descriptive statistics (minimum, 1st quantile, median, mean, 3rd quantile, maximum) of the independent variables are shown in the following table, using the column numbers from the list above:

	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
Min.	2.980	0.500	7.78	0.530	3.00	2.820	0.000	0.000	3.00	4.00	0.000
1st Qu.	4.140	1.200	18.40	1.190	4.28	4.150	1.000	1.000	10.50	23.00	6.000
Median	4.500	1.350	20.46	1.350	4.57	4.510	1.330	1.330	12.89	28.00	8.000
Mean	4.697	1.452	20.04	1.447	4.77	4.711	1.457	1.457	13.00	28.39	8.751
3rd Qu.	5.120	1.640	21.90	1.640	5.20	5.140	2.000	2.000	15.30	34.00	11.00
Max.	7.560	2.750	29.33	2.800	8.12	7.590	6.000	6.000	31.00	56.00	27.00

Home teams, like away teams, scored an average of 1.5 goals per game. For that, both home teams and away teams needed an average of 4.7 shots on goal per game. So, there is not much difference in shots on goal and goals scored. On average, there were 4.7 corners per game for the away team, both if you look at the statistics against all teams and against their respective opponents. Home and away teams are also about equal when it comes to fouls, with an average of 13.5 fouls per game.

Now the models are trained. First a control function is set up with the "timeslice"-method to make rolling forecasts for every matchday (9 games) based on the whole data before the respective matchday. As this is a timeseries dataset, where the date of a game matters, we can't simply do a cross-validation approach. The reason for that is that we can't take a game from 2020 to predict a 2016 game, as this would not reflect the reality of sports betting. So instead, we use the "timeslice"-method, where the first 306 rows are used as the training set and the next 9 games as the validation set. The next iteration takes the first 315 (=306+9) games as the training set and again the next 9 games as the validation set. In that way we always only use previous games to predict upcoming matches.

The following 5 multiclass individual models are calculated:

- K-nearest Neighbors
- Penalized Discriminant Analysis
- Shrinkage Discriminant Analysis
- High Dimensional Discriminant Analysis
- Single C5.0 Tree

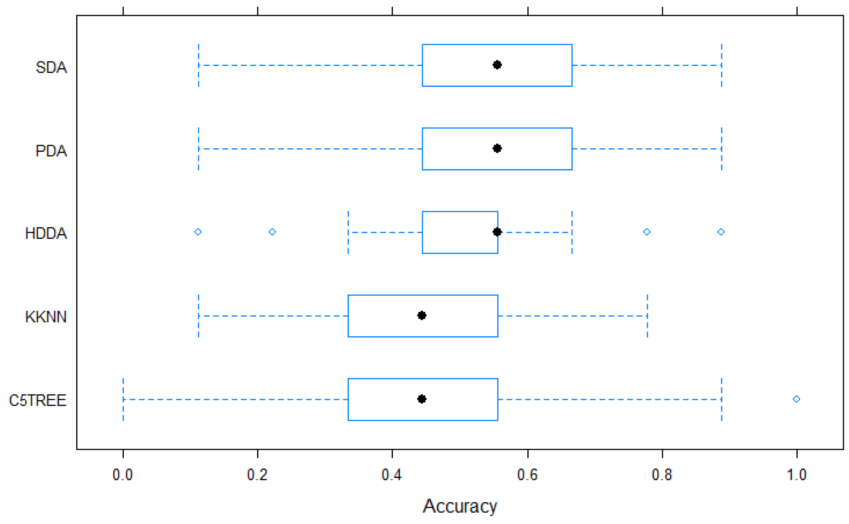
KNN is a simple algorithm that assumes that similar things are near each other. So, if a data point is near another data point, it is assumed that both belong to similar classes. Linear discriminant analysis is a data analytic tool used to examine the relationship between a set of predictors and a categorical response. Shrinkage is a form of regularization used to improve the estimation of covariance matrices in situations where the number of training samples is small compared to the number of features. HDDA reduces the dimension for each class independently and adjusts the conditional covariance matrices of the classes to fit the Gaussian framework to high-dimensional data. The C5.0 algorithm has become a standard for producing decision trees, because it does well for most types of problems directly out of the box.

4 Results and Performance

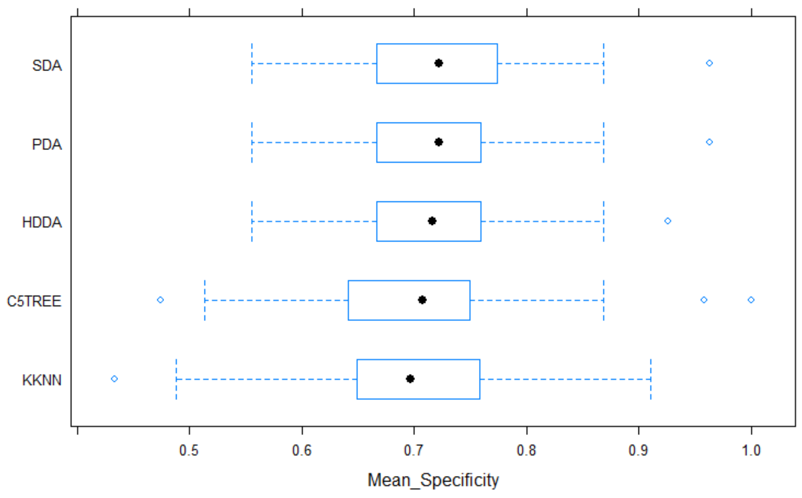
To summarize the results of each model and compare them, the accuracy of the single models is shown in the following tables:

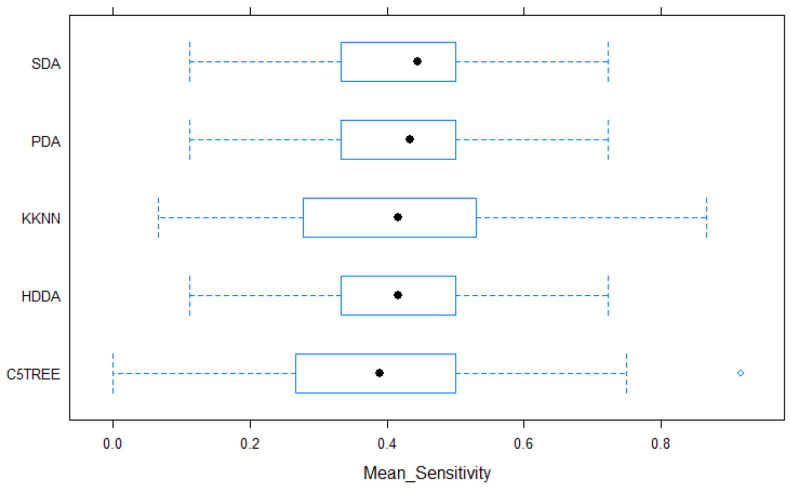
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
KKNN	0.1111111	0.3333333	0.4444444	0.4426667	0.5555556	0.7777778	0
PDA	0.1111111	0.4444444	0.5555556	0.5057778	0.6666667	0.8888889	0
SDA	0.1111111	0.4444444	0.5555556	0.5075556	0.6666667	0.8888889	0
HDDA	0.1111111	0.4444444	0.5555556	0.5031111	0.5555556	0.8888889	0
C5TREE	0.0000000	0.3333333	0.4444444	0.4400000	0.5555556	10.000.000	0

The table above and the graph below show, that While the Shrinkage Discriminant Analysis, Penalized Discriminant Analysis, and High Dimensional Discriminant Analysis models have a similar mean performance of 51 %. Therefore it is hard to decide, which model to choose, so we pick the SDA as it has the slightly better accuracy.



Also, the SDA has the better Sensitivity than PDA as it can be seen in the boxplot below.





In the next step, the results of the test set are predicted, and the confusion matrix is created again. The test accuracy is by 51%. Again, the performance of the home matches is predicted significantly better than away matches and draws.

		Reference		
		A	D	H
Prediction	A	45	1	53
	D	19	0	56
	H	20	0	112

The model is good at predicting home wins (85 % accuracy) but is bad at predicting away wins (44 %) and is not able to predict draws at all (0 %). The next step is to predict the accuracy of Bookmakers.

First, it must be proved that bookmakers always favor one of the two teams, while a draw never has the lowest odds and thus the highest probability of occurrence. So, bookmakers never bet in draws. Then it is checked whether the bookmakers' predictions were right or wrong. To establish comparability between the bookmakers' model and our own model, the accuracy of the bookmakers is calculated again, i.e., how often the favorite was tipped correctly. Our accuracy is higher than those of the bookmakers with 33%, but bookmakers never bet on draws, so every D is automatically a wrong prediction. The bookmaker goal is not to predict the games correctly, but to provide their odds in such a way, that they make profit out of it. By comparing our overall accuracy (51 %) to the accuracy of the bookmaker predictions (33%), we still have comparably good results.

Now we want to test the economic efficiency of our outcome probabilities and compare it with the bookmaker odds. The betting strategy is as following: always bet on the favorite. We bet 1$ hypothetically on every predicted match outcome. If the prediction is right, we multiply the specific odd for the outcome with 1$ and add it to our "bank account". At the end we see, how much money we have won or lost.

We lost $13.83 over the season 2020/21, so the model is not able to generate profit for us on the betting market. As our model seems to predict home wins better than the other results, we will test another betting strategy, by only betting on home win predictions. By doing that we still end up losing $7.54. The reason for that is that the average home win odds are lower (2.2) than the draw odds (4.3) and away win odds (5.0), which means that a won bet on home wins results in less profit than on other results.

5 Discussion and Recommendations

The conclusion of our project is that we are not able to compete with the sports betting industry. It is also hard to compare our model to the odds, as they build their models not to predict the games right, but to generate profit with the odds. They also include the betting behavior of the users to build the optimal odds for them. Even though we lost money with that method, it wasn't much considered the fact that we put in a total of $306 in stacks and only lost $13.83/$7.54.

To improve our model, we should also exclude the option to predict draws, as our model totally failed to predict those. Using only two class classification models, we have more models to try and therefore maximize the chance of finding the optimal model for our problem. By balancing out the class imbalance with an up- or down sampling method, we should expect even better results.

Nevertheless, there is a need for further optimization in the future to increase the profit even more. For example, we could consider which players are in the starting eleven, the club's injury history, and which players are suspended due to red cards. It can be assumed that if a team has the best players ready and in the starting lineup, that the performance will be better. This is not reflected in our model because there is no information about the starting lineup. In addition, the strength of the players could be used with ratings, for example from the soccer manager game FIFA, to consider the strength of the team and generate an overall score for the team.

References

Football results, Statistics &amp; Soccer Betting Odds Data. (n.d.). Retrieved December 1, 2021, from http://www.football-data.co.uk/data.php.

Appendix

Table 1

Variable	**Team**	**Period**	**Matches**	**Description**
HT_Form_Points	HT	last 6 games	total	win = 3 points, draw = 1 p
HT_Form_Scored	HT	last 6 games	total	scored goals
HT_Form_Conceded	HT	last 6 games	total	conceded goals
HT_shots_form	HT	last 6 games	total	shots
HT_shots_on_target_form	HT	last 6 games	total	shots on target
HT_corners_form	HT	last 6 games	total	corners
HT_fouls_form	HT	last 6 games	total	fouls
HT_card_points_form	HT	last 6 games	total	yellow card = 10 points red card = 25 points
HT_Mean_Points	HT	entire past	total	win = 3 points, draw = 1 p
HT_Mean_Scored	HT	entire past	total	scored goals
HT_Mean_Conceded	HT	entire past	total	conceded goals
HT_Mean_shots	HT	entire past	total	shots
HT_Mean_shots_on_target	HT	entire past	total	shots on target
HT_Mean_corners	HT	entire past	total	corners
HT_Mean_fouls	HT	entire past	total	fouls
HT_Mean_card_points	HT	entire past	total	yellow card = 10 points red card = 25 points
AT_Form_Points	AT	last 6 games	total	win = 3 points, draw = 1 p
AT_Form_Scored	AT	last 6 games	total	scored goals
AT_Form_Conceded	AT	last 6 games	total	conceded goals
AT_shots_form	AT	last 6 games	total	shots
AT_shots_on_target_form	AT	last 6 games	total	shots on target
AT_corners_form	AT	last 6 games	total	corners
AT_fouls_form	AT	last 6 games	total	fouls
AT_card_points_form	AT	last 6 games	total	yellow card = 10 points red card = 25 points
AT_Mean_Points	AT	entire past	total	win = 3 points, draw = 1 p
AT_Mean_Scored	AT	entire past	total	scored goals
AT_Mean_Conceded	AT	entire past	total	conceded goals

AT_Mean_shots	AT	entire past	total	shots
AT_Mean_shots_on_target	AT	entire past	total	shots on target
AT_Mean_corners	AT	entire past	total	corners
AT_Mean_fouls	AT	entire past	total	fouls
AT_Mean_card_points	AT	entire past	total	yellow card = 10 points red card = 25 points

HT_Form_Points_opp	HT	last 6 games	opponent	win = 3 points, draw = 1 p
HT_Form_Scored_opp	HT	last 6 games	opponent	scored goals
HT_Form_Conceded_opp	HT	last 6 games	opponent	conceded goals
HT_shots_form_opp	HT	last 6 games	opponent	shots
HT_shots_on_target_form_opp	HT	last 6 games	opponent	shots on target
HT_corners_form_opp	HT	last 6 games	opponent	corners
HT_fouls_form_opp	HT	last 6 games	opponent	fouls
HT_card_points_form_opp	HT	last 6 games	opponent	yellow card = 10 points red card = 25 points

HT_Mean_Points_opp	HT	entire past	opponent	win = 3 points, draw = 1 p
HT_Mean_Scored_opp	HT	entire past	opponent	scored goals
HT_Mean_Conceded_opp	HT	entire past	opponent	conceded goals
HT_Mean_shots_opp	HT	entire past	opponent	shots
HT_Mean_shots_on_target_opp	HT	entire past	opponent	shots on target
HT_Mean_corners_opp	HT	entire past	opponent	corners
HT_Mean_fouls_opp	HT	entire past	opponent	fouls
HT_Mean_card_points_opp	HT	entire past	opponent	yellow card = 10 points red card = 25 points

AT_Form_Points_opp	AT	last 6 games	opponent	win = 3 points, draw = 1 p
AT_Form_Scored_opp	AT	last 6 games	opponent	scored goals
AT_Form_Conceded_opp	AT	last 6 games	opponent	conceded goals
AT_shots_form_opp	AT	last 6 games	opponent	shots
AT_shots_on_target_form_opp	AT	last 6 games	opponent	shots on target
AT_corners_form_opp	AT	last 6 games	opponent	corners
AT_fouls_form_opp	AT	last 6 games	opponent	fouls
AT_card_points_form_opp	AT	last 6 games	opponent	yellow card = 10 points red card = 25 points

AT_Mean_Points_opp	AT	entire past	opponent	win = 3 points, draw = 1 p
AT_Mean_Scored_opp	AT	entire past	opponent	scored goals
AT_Mean_Conceded_opp	AT	entire past	opponent	conceded goals
AT_Mean_shots_opp	AT	entire past	opponent	shots
AT_Mean_shots_on_target_opp	AT	entire past	opponent	shots on target
AT_Mean_corners_opp	AT	entire past	opponent	corners
AT_Mean_fouls_opp	AT	entire past	opponent	fouls
AT_Mean_card_points_opp	AT	entire past	opponent	yellow card = 10 points red card = 25 points

R-Code

```
library(dplyr)

library(tidyverse)

library(ggplot2)

library(caret)

library(e1071)

library(psych)

library(corrplot)

library(corrgram)

library(pROC)

library(leaps)

library(readr)

library(nnet)

library(randomForest)

library(MLmetrics)



#Build the dataframe----

season0 <- read.csv("D1 (14).csv", stringsAsFactors = T)

season1 <- read.csv("D1 (12).csv", stringsAsFactors = T)

season2 <- read.csv("D1 (11).csv", stringsAsFactors = T)

season3 <- read.csv("D1 (10).csv", stringsAsFactors = T)

season4 <- read.csv("D1 (9).csv", stringsAsFactors = T)

season5 <- read.csv("D1 (8).csv", stringsAsFactors = T)

season6 <- read.csv("D1 (7).csv", stringsAsFactors = T)
```

```
cols_1 <- c("Date", "HomeTeam", "AwayTeam","FTHG",          "FTAG","FTR"   ,"HS"    ,"AS"
        ,"HST","AST","HF","AF","HC","AC","HY","AY","HR","AR","BbAvH","BbAvD","BbAvA")
cols_2 <- c("Date", "HomeTeam", "AwayTeam","FTHG",          "FTAG","FTR"   ,"HS"    ,"AS"
        ,"HST","AST","HF","AF","HC","AC","HY","AY","HR","AR","AvgH","AvgD","AvgA")


season0 <- season0[cols_1]
season1 <- season1[cols_1]
season2 <- season2[cols_1]
season3 <- season3[cols_1]
season4 <- season4[cols_1]
season5 <- season5[cols_2]
season6 <- season6[cols_2]


season0 <- rename(season0, H_Odds = BbAvH, D_Odds = BbAvD, A_Odds = BbAvA)
season1 <- rename(season1, H_Odds = BbAvH, D_Odds = BbAvD, A_Odds = BbAvA)
season2 <- rename(season2, H_Odds = BbAvH, D_Odds = BbAvD, A_Odds = BbAvA)
season3 <- rename(season3, H_Odds = BbAvH, D_Odds = BbAvD, A_Odds = BbAvA)
season4 <- rename(season4, H_Odds = BbAvH, D_Odds = BbAvD, A_Odds = BbAvA)
season5 <- rename(season5, H_Odds = AvgH, D_Odds = AvgD, A_Odds = AvgA)
season6 <- rename(season6, H_Odds = AvgH, D_Odds = AvgD, A_Odds = AvgA)


df <- rbind(season0, season1, season2, season3, season4, season5, season6)
rm(season0, season1, season2, season3, season4, season5, season6)


sum(is.na(df))
```

```
df$Date <- as.Date(df$Date, "%d/%m/%y")

df$goals <- df$FTHG + df$FTAG


#Build functions to compute new variables-------

#create a function that takes a teams last 6/total games and sums up the stats

stat_function <- function(df, team, date) {

 last_6 <- tail(df[(df$HomeTeam == team | df$AwayTeam == team) & (df$Date < date),])

 past <- df[(df$HomeTeam == team | df$AwayTeam == team) & (df$Date < date),]

 point_count <- 0

 scored_count <- 0

 conceded_count <- 0

 shots_count <- 0

 shots_on_target_count <- 0

 corners_count <- 0

 fouls_count <- 0

 yellow_card_count <- 0

 red_card_count <- 0

 point_count_6 <- 0

 scored_count_6 <- 0

 conceded_count_6 <- 0

 shots_count_6 <- 0

 shots_on_target_count_6 <- 0

 corners_count_6 <- 0

 fouls_count_6 <- 0

 yellow_card_count_6 <- 0

 red_card_count_6 <- 0
```

```
if(nrow(last_6) < 3){


}

else {

  for (match in 1:nrow(last_6)){

    m <- last_6[match,]

    if (m$HomeTeam == team){

      scored_count_6 <- scored_count_6 + m$FTHG

      conceded_count_6 <- conceded_count_6 + m$FTAG

      shots_count_6 <- shots_count_6 + m$HS

      shots_on_target_count_6 <- shots_on_target_count_6 + m$HST

      corners_count_6 <- corners_count_6 + m$HC

      fouls_count_6 <- fouls_count_6 + m$HF

      yellow_card_count_6 <- yellow_card_count_6 + m$HY*10

      red_card_count_6 <- red_card_count_6 + m$HR*25

      if (m$FTR == "H"){

        point_count_6 <- point_count_6 + 3

      }

      else if (m$FTR == "D"){

        point_count_6 <- point_count_6 +1

      }

    }

    else {

      scored_count_6 <- scored_count_6 + m$FTAG

      conceded_count_6 <- conceded_count_6 + m$FTHG

      shots_count_6 <- shots_count_6 + m$AS
```

```
        shots_on_target_count_6 <- shots_on_target_count_6 + m$AST
        corners_count_6 <- corners_count_6 + m$AC
        fouls_count_6 <- fouls_count_6 + m$AF
        yellow_card_count_6 <- yellow_card_count_6 + m$AY*10
        red_card_count_6 <- red_card_count_6 + m$AR*25
        if (m$FTR == "A"){
          point_count_6 <- point_count_6 + 3
        }
        else if (m$FTR == "D"){
          point_count_6 <- point_count_6 +1
        }
      }
    }
  }

  if(nrow(past) < 3){


  }
  else {
    for (match in 1:nrow(past)){
      m <- past[match,]
      if (m$HomeTeam == team){
        scored_count <- scored_count + m$FTHG
        conceded_count <- conceded_count + m$FTAG
        shots_count <- shots_count + m$HS
        shots_on_target_count <- shots_on_target_count + m$HST
        corners_count <- corners_count + m$HC
        fouls_count <- fouls_count + m$HF
        yellow_card_count <- yellow_card_count + m$HY*10
        red_card_count <- red_card_count + m$HR*25
        if (m$FTR == "H"){
          point_count <- point_count + 3
        }
        else if (m$FTR == "D"){
          point_count <- point_count +1
        }
      }
      else {
        scored_count <- scored_count + m$FTAG
        conceded_count <- conceded_count + m$FTHG
        shots_count <- shots_count + m$AS
        shots_on_target_count <- shots_on_target_count + m$AST
        corners_count <- corners_count + m$AC
        fouls_count <- fouls_count + m$AF
        yellow_card_count <- yellow_card_count + m$AY*10
        red_card_count <- red_card_count + m$AR*25
        if (m$FTR == "A"){
          point_count <- point_count + 3
        }
        else if (m$FTR == "D"){
          point_count <- point_count +1
        }
      }
    }
}
games_count <- nrow(past)
points <- point_count/games_count
mean_scored <- scored_count/games_count
mean_conceded <- conceded_count/games_count
mean_shots <- shots_count/games_count
mean_shots_on_target <- shots_on_target_count/games_count
mean_corners <- corners_count/games_count
mean_fouls <- fouls_count/games_count
card_points <- (yellow_card_count + red_card_count)/games_count
card_points_6 <- yellow_card_count_6 + red_card_count_6

result <<- data.frame("point_form" = point_count_6,
                      "scored_form" = scored_count_6,
                      "conceded_form" = conceded_count_6,
                      "shots_form" = shots_count_6,
                      "shots_on_target_form" = shots_on_target_count_6,
                      "corners_form" = corners_count_6,
                      "fouls_form" = fouls_count_6,
                      "card_points_form" = card_points_6,
                      "mean_points" <- points,
                      "mean_scored" = mean_scored,
                      "mean_conceded" = mean_conceded,
                      "mean_shots" = mean_shots,
              "mean_shots_on_target" = mean_shots_on_target,
              "mean_corners" = mean_corners,
              "mean_fouls" = mean_fouls,
              "card_points" = card_points
              )

   result
}



#create a function that takes a teams last 6/total games vs Opponent and sums up the stats
stat_function_opp <- function(df, HT, AT, date){
 last_6 <- tail(df[((df$HomeTeam == HT & df$AwayTeam == AT) | (df$HomeTeam == AT &
df$AwayTeam == HT)) & (df$Date < date),])
 past <-df[((df$HomeTeam == HT & df$AwayTeam == AT) | (df$HomeTeam == AT & df$AwayTeam ==
HT)) & (df$Date < date),]
 point_count <- 0
 scored_count <- 0
 conceded_count <- 0
 shots_count <- 0
 shots_on_target_count <- 0
 corners_count <- 0
 fouls_count <- 0
 yellow_card_count <- 0
 red_card_count <- 0
 point_count_6 <- 0
```

```
scored_count_6 <- 0
conceded_count_6 <- 0
shots_count_6 <- 0
shots_on_target_count_6 <- 0
corners_count_6 <- 0
fouls_count_6 <- 0
yellow_card_count_6 <- 0
red_card_count_6 <- 0
if(nrow(last_6) == 0){


}
else{
  for (match in 1:nrow(last_6)){
    m <- last_6[match,]
    if (m$HomeTeam == HT){
      scored_count_6 <- scored_count_6 + m$FTHG
      conceded_count_6 <- conceded_count_6 + m$FTAG
      shots_count_6 <- shots_count_6 + m$HS
      shots_on_target_count_6 <- shots_on_target_count_6 + m$HST
      corners_count_6 <- corners_count_6 + m$HC
      fouls_count_6 <- fouls_count_6 + m$HF
      yellow_card_count_6 <- yellow_card_count_6 + m$HY*10
      red_card_count_6 <- red_card_count_6 + m$HR*25
      if (m$FTR == "H"){
        point_count_6 <- point_count_6 + 3
      }
```

```
        else if (m$FTR == "D"){
          point_count_6 <- point_count_6 +1
        }
      }
      else {
        scored_count_6 <- scored_count_6 + m$FTAG
        conceded_count_6 <- conceded_count_6 + m$FTHG
        shots_count_6 <- shots_count_6 + m$AS
        shots_on_target_count_6 <- shots_on_target_count_6 + m$AST
        corners_count_6 <- corners_count_6 + m$AC
        fouls_count_6 <- fouls_count_6 + m$AF
        yellow_card_count_6 <- yellow_card_count_6 + m$AY*10
        red_card_count_6 <- red_card_count_6 + m$AR*25
        if (m$FTR == "A"){
          point_count_6 <- point_count_6 + 3
        }
        else if (m$FTR == "D"){
          point_count_6 <- point_count_6 +1
        }
      }
    }
  }
  if(nrow(past) == 0){


  }
  else{
```

```
  for (match in 1:nrow(past)){
    m <- past[match,]
    if (m$HomeTeam == HT){
      scored_count <- scored_count + m$FTHG
      conceded_count <- conceded_count + m$FTAG
      shots_count <- shots_count + m$HS
      shots_on_target_count <- shots_on_target_count + m$HST
      corners_count <- corners_count + m$HC
      fouls_count <- fouls_count + m$HF
      yellow_card_count <- yellow_card_count + m$HY*10
      red_card_count <- red_card_count + m$HR*25
      if (m$FTR == "H"){
        point_count <- point_count + 3
      }
      else if (m$FTR == "D"){
        point_count <- point_count +1
      }
    }
    else {
      scored_count <- scored_count + m$FTAG
      conceded_count <- conceded_count + m$FTHG
      shots_count <- shots_count + m$AS
      shots_on_target_count <- shots_on_target_count + m$AST
      corners_count <- corners_count + m$AC
      fouls_count <- fouls_count + m$AF
      yellow_card_count <- yellow_card_count + m$AY*10
      red_card_count <- red_card_count + m$AR*25
      if (m$FTR == "A"){
        point_count <- point_count + 3
      }
      else if (m$FTR == "D"){
        point_count <- point_count +1
      }
    }
  }
}
games_count <- nrow(past)
points <- point_count/games_count
mean_scored <- scored_count/games_count
mean_conceded <- conceded_count/games_count
mean_shots <- shots_count/games_count
mean_shots_on_target <- shots_on_target_count/games_count
mean_corners <- corners_count/games_count
mean_fouls <- fouls_count/games_count
card_points <- (yellow_card_count + red_card_count)/games_count
card_points_6 <- yellow_card_count_6 + red_card_count_6


result <<- data.frame("point_form_opp" = point_count_6,
                      "scored_form_opp" = scored_count_6,
                      "conceded_form_opp" = conceded_count_6,
                      "shots_form_opp" = shots_count_6,
                      "shots_on_target_form_opp" = shots_on_target_count_6,
                      "corners_form_opp" = corners_count_6,
                      "fouls_form_opp" = fouls_count_6,
                      "card_points_form_opp" = card_points_6,
                      "mean_poins_opp" <- points,
                      "mean_scored_opp" = mean_scored,
                      "mean_conceded_opp" = mean_conceded,
                      "mean_shots_opp" = mean_shots,
                      "mean_shots_on_target_opp" = mean_shots_on_target,
                      "mean_corners_opp" = mean_corners,
                      "mean_fouls_opp" = mean_fouls,
                      "card_points_opp" = card_points
            )

            result
          }



          #Apply functions to dataframe----
          #Create cols for the new variables that will be computed by the functions
          #Distinguish between: HT and AT, Last 6 games and total past games, all games and against specific opponent

          df$HT_Form_Points <- NA
          df$HT_Form_Scored <- NA
          df$HT_Form_Conceded <- NA
```

```
df$HT_shots_form <- NA

df$HT_shots_on_target_form <- NA

df$HT_corners_form <- NA

df$HT_fouls_form <- NA

df$HT_card_points_form <- NA


df$HT_Mean_Points <- NA

df$HT_Mean_Scored <- NA

df$HT_Mean_Conceded <- NA

df$HT_Mean_shots <- NA

df$HT_Mean_shots_on_target <- NA

df$HT_Mean_corners <- NA

df$HT_Mean_fouls<- NA

df$HT_Mean_card_points <- NA


df$AT_Form_Points <- NA

df$AT_Form_Scored <- NA

df$AT_Form_Conceded <- NA

df$AT_shots_form <- NA

df$AT_shots_on_target_form <- NA

df$AT_corners_form <- NA

df$AT_fouls_form <- NA

df$AT_card_points_form <- NA


df$AT_Mean_Points <- NA

df$AT_Mean_Scored <- NA
```

```
df$AT_Mean_Conceded <- NA

df$AT_Mean_shots <- NA

df$AT_Mean_shots_on_target <- NA

df$AT_Mean_corners <- NA

df$AT_Mean_fouls <- NA

df$AT_Mean_card_points <- NA


df$HT_Form_Points_opp <- NA

df$HT_Form_Scored_opp <- NA

df$HT_Form_Conceded_opp <- NA

df$HT_shots_form_opp <- NA

df$HT_shots_on_target_form_opp <- NA

df$HT_corners_form_opp <- NA

df$HT_fouls_form_opp <- NA

df$HT_card_points_form_opp <- NA


df$HT_Mean_Points_opp <- NA

df$HT_Mean_Scored_opp <- NA

df$HT_Mean_Conceded_opp <- NA

df$HT_Mean_shots_opp <- NA

df$HT_Mean_shots_on_target_opp <- NA

df$HT_Mean_corners_opp <- NA

df$HT_Mean_fouls_opp <- NA

df$HT_Mean_card_points_opp <- NA


df$AT_Form_Points_opp <- NA
```

```
df$AT_Form_Scored_opp <- NA

df$AT_Form_Conceded_opp <- NA

df$AT_shots_form_opp <- NA

df$AT_shots_on_target_form_opp <- NA

df$AT_corners_form_opp <- NA

df$AT_fouls_form_opp <- NA

df$AT_card_points_form_opp <- NA


df$AT_Mean_Points_opp <- NA

df$AT_Mean_Scored_opp <- NA

df$AT_Mean_Conceded_opp <- NA

df$AT_Mean_shots_opp <- NA

df$AT_Mean_shots_on_target_opp <- NA

df$AT_Mean_corners_opp <- NA

df$AT_Mean_fouls_opp <- NA

df$AT_Mean_card_points_opp <- NA



#apply the functions on every match of the df

for (i in 1:nrow(df)){

  stat_home <- stat_function(df, df[i,]$HomeTeam, df[i,]$Date)

  df[i,]$HT_Form_Points <- stat_home[1]

  df[i,]$HT_Form_Scored <- stat_home[2]

  df[i,]$HT_Form_Conceded <- stat_home[3]

  df[i,]$HT_shots_form <- stat_home[4]

  df[i,]$HT_shots_on_target_form <- stat_home[5]
```

```
df[i,]$HT_corners_form <- stat_home[6]

df[i,]$HT_fouls_form <- stat_home[7]

df[i,]$HT_card_points_form <- stat_home[8]


df[i,]$HT_Mean_Points <- stat_home[9]

df[i,]$HT_Mean_Scored <- stat_home[10]

df[i,]$HT_Mean_Conceded <- stat_home[11]

df[i,]$HT_Mean_shots <- stat_home[12]

df[i,]$HT_Mean_shots_on_target <- stat_home[13]

df[i,]$HT_Mean_corners <- stat_home[14]

df[i,]$HT_Mean_fouls<- stat_home[15]

df[i,]$HT_Mean_card_points <- stat_home[16]


stat_away <- stat_function(df, df[i,]$AwayTeam, df[i,]$Date)

df[i,]$AT_Form_Points <- stat_away[1]

df[i,]$AT_Form_Scored <- stat_away[2]

df[i,]$AT_Form_Conceded <- stat_away[3]

df[i,]$AT_shots_form <- stat_away[4]

df[i,]$AT_shots_on_target_form <- stat_away[5]

df[i,]$AT_corners_form <- stat_away[6]

df[i,]$AT_fouls_form <- stat_away[7]

df[i,]$AT_card_points_form <- stat_away[8]


df[i,]$AT_Mean_Points <- stat_away[9]

df[i,]$AT_Mean_Scored <- stat_away[10]

df[i,]$AT_Mean_Conceded <- stat_away[11]
```

```
df[i,]$AT_Mean_shots <- stat_away[12]

df[i,]$AT_Mean_shots_on_target <- stat_away[13]

df[i,]$AT_Mean_corners <- stat_away[14]

df[i,]$AT_Mean_fouls<- stat_away[15]

df[i,]$AT_Mean_card_points <- stat_away[16]


stat_home_opp <- stat_function_opp(df, df[i,]$HomeTeam, df[i,]$AwayTeam, df[i,]$Date)

df[i,]$HT_Form_Points_opp <- stat_home_opp[1]

df[i,]$HT_Form_Scored_opp <- stat_home_opp[2]

df[i,]$HT_Form_Conceded_opp <- stat_home_opp[3]

df[i,]$HT_shots_form_opp <- stat_home_opp[4]

df[i,]$HT_shots_on_target_form_opp <- stat_home_opp[5]

df[i,]$HT_corners_form_opp <- stat_home_opp[6]

df[i,]$HT_fouls_form_opp <- stat_home_opp[7]

df[i,]$HT_card_points_form_opp <- stat_home_opp[8]


df[i,]$HT_Mean_Points_opp <- stat_home_opp[9]

df[i,]$HT_Mean_Scored_opp <- stat_home_opp[10]

df[i,]$HT_Mean_Conceded_opp <- stat_home_opp[11]

df[i,]$HT_Mean_shots_opp <- stat_home_opp[12]

df[i,]$HT_Mean_shots_on_target_opp <- stat_home_opp[13]

df[i,]$HT_Mean_corners_opp <- stat_home_opp[14]

df[i,]$HT_Mean_fouls_opp<- stat_home_opp[15]

df[i,]$HT_Mean_card_points_opp <- stat_home_opp[16]


stat_away_opp <- stat_function_opp(df, df[i,]$AwayTeam, df[i,]$HomeTeam, df[i,]$Date)
```

```
  df[i,]$AT_Form_Points_opp <- stat_away_opp[1]

  df[i,]$AT_Form_Scored_opp <- stat_away_opp[2]

  df[i,]$AT_Form_Conceded_opp <- stat_away_opp[3]

  df[i,]$AT_shots_form_opp <- stat_away_opp[4]

  df[i,]$AT_shots_on_target_form_opp <- stat_away_opp[5]

  df[i,]$AT_corners_form_opp <- stat_away_opp[6]

  df[i,]$AT_fouls_form_opp <- stat_away_opp[7]

  df[i,]$AT_card_points_form_opp <- stat_away_opp[8]


  df[i,]$AT_Mean_Points_opp <- stat_away_opp[9]

  df[i,]$AT_Mean_Scored_opp <- stat_away_opp[10]

  df[i,]$AT_Mean_Conceded_opp <- stat_away_opp[11]

  df[i,]$AT_Mean_shots_opp <- stat_away_opp[12]

  df[i,]$AT_Mean_shots_on_target_opp <- stat_away_opp[13]

  df[i,]$AT_Mean_corners_opp <- stat_away_opp[14]

  df[i,]$AT_Mean_fouls_opp<- stat_away_opp[15]

  df[i,]$AT_Mean_card_points_opp <- stat_away_opp[16]

}


#Convert the dataypes of the columns from list to numeric

d <- df


d[,23:86] <- lapply(d[,23:86], unlist)

d[,23:86] <- lapply(d[,23:86], as.numeric)


#Remove NaN-rows in the data
```

```
sum(is.na(d))

d <- na.omit(d)

str(d)


#Round the values

d <- d %>% mutate_if(is.numeric, ~round(.,2))


#Remove unneeded df's

rm(result, stat_away, stat_away_opp, stat_home, stat_home_opp)


###Model Building-------

d_odds <- d


#Remove columns that we don't need for the model

drop <- c("Date", "HomeTeam", "AwayTeam", "FTHG", "FTAG", "HS", "AS", "HST", "AST", "HF", "AF",

"HC", "AC", "HY", "AY", "HR", "AR", "H_Odds", "D_Odds", "A_Odds", "goals", "Is_over_2.5")

d <- d[,!(names(d) %in% drop)]


#Feature Selection----


#identify correlated predictors

cor_mat <- cor(subset(d, select = -FTR))

highlyCorrelated <- findCorrelation(cor_mat, cutoff=0.75)


highlyCorrelated <- highlyCorrelated[!highlyCorrelated %in% 1]

names(df[,highlyCorrelated])
```

```
d <- d[,-highlyCorrelated]


#Recursive Feature Elimination

set.seed(101)

rfeC <- rfeControl(functions = rfFuncs, method="cv", number=10)


results <- rfe(d[,2:37], d[,1], sizes=(1:18), rfeControl=rfeC)


print(results)

predictors(results)

plot(results, type=c("g", "o"))


p <- head(results$optVariables, 11)


p <- c("FTR", p)


d <- d[,p]


#Descriptive statistics of our variables

names(d)

plot(d$FTR)

summary(d)



train <- head(d, n=1558)
```

```
test <- tail(d,n=306)



#Train models----


#Setup control function to make rolling forecasts for every matchday (9 games) based on the whole
data before the respective matchday


ctrl <- trainControl(method = "timeslice",
                 initialWindow = 306,
                 horizon = 9,
                 fixedWindow = FALSE,
                 skip = 9,
                 classProbs = TRUE,
                 savePredictions = TRUE,
                 summaryFunction = multiClassSummary,
                 allowParallel = TRUE)



#K-nearest Neighbors
d_kknn <- train(FTR~., data=train, method="kknn", metric="ROC",
             trControl=ctrl, preProcess = c("center", "scale") )
print(d_kknn)


#Penalized Discriminant Analysis
d_pda <- train(FTR~., data=train, method="pda", metric="ROC",
             trControl=ctrl, preProcess = c("center", "scale") )
print(d_pda)


#Shrinkage Discriminant Analysis
d_sda <- train(FTR~., data=train, method="sda", metric="ROC",
             trControl=ctrl, preProcess = c("center", "scale") )
print(d_sda)



#High Dimensional Discriminant Analysis
d_hdda <- train(FTR~., data=train, method="hdda", metric="ROC",
              trControl=ctrl, preProcess = c("center", "scale") )
print(d_hdda)


#Single C5.0 Tree
d_C5 <- train(FTR~., data=train, method="C5.0Tree", metric="ROC",
            trControl=ctrl, preProcess = c("center", "scale") )
print(d_C5)



resample_results <- resamples(list(KKNN=d_kknn,PDA=d_pda,SDA=d_sda,
                              HDDA=d_hdda, C5TREE=d_C5))


summary(resample_results,metric = "Accuracy")


bwplot(resample_results , metric = "Accuracy")
bwplot(resample_results , metric = "Mean_Specificity")

bwplot(resample_results , metric = "Mean_Sensitivity")



#Best model is the Shrinkage Discriminant Analysis with a mean accuracy of 51%


conf_Matrix <- confusionMatrix(d_sda$pred$obs, d_sda$pred$pred)

conf_Matrix

#Good H prediction, Bad D predictions, Bad A predictions


preds <- predict(d_sda, test)


confusionMatrix(test$FTR, preds)


#Test Accuracy is now 51%



#Prediction accuracy of Bookmakers


sum(d_odds$D_Odds > d_odds$H_Odds & d_odds$D > d_odds$A_Odds)

#->Bookmakers never bet on draws


d_odds$pred <- as.factor(ifelse(d_odds$H_Odds < d_odds$A_Odds, "H", "A"))


d_odds$pred <- as.numeric(d_odds$pred)

d_odds$FTR <- as.numeric(d_odds$FTR)
```

```
d_odds$right_pred <- ifelse(d_odds$FTR == d_odds$pred, 1, 0)

bookmaker_acc <- sum(d_odds$right_pred)/nrow(d_odds)

bookmaker_acc


#Our accuracy is higher, but bookmakers never bet on draws, so every D is a wrong prediction

#The bookmakers goal is not to predict the games correctly, but to provide their

#odds in such a way, that they make profit out of it


#Now we want to test the economic efficiency of our outcome probabilities with the bookmakers odds

#Betting strategy: Always bet on the favorite

#We bet 1$ hypothetically on every predicted match outcome

#If the prediction is right, we multiply the specific odd for the outcome with 1$ and add it to our "bank account"

#At the end we see, how much money we lost/won


bet_test <- tail(d_odds, n=306)


bet <- cbind(bet_test[,c("FTR", "H_Odds", "D_Odds", "A_Odds")], preds)

bet$acc <- NA


bet$FTR <- as.factor(bet$FTR)

bet$FTR <- ifelse(bet$FTR == 1, "A", ifelse(bet$FTR == 2, "D", "H"))


for (i in 1:nrow(bet)){

  b <- bet[i,]
  if (b$FTR == b$preds){
    if(b$FTR == "H"){
      bet[i,]$acc <- b$H_Odds - 1
    }
    else if(b$FTR == "D"){
      bet[i,]$acc <- b$D_Odds - 1
    }
    else {
      bet[i,]$acc <- b$A_Odds - 1
    }
  }
  else {
    bet[i,]$acc <- -1
  }
}


bank_account <- sum(bet$acc)
bank_account
#By betting 1$ on the predicted result of every match we do lose 13.83$ over 1 season, which is not too bad to start with
#As we analyzed before, our predictions are way better when we look at home win predictions
#Now we want to find out what our profit, would be if we just place bets when the prediction is a home win


h_bet <- bet[bet$preds == "H",]
```

```
sum(h_bet$acc)

#We still lose 7.54$ over the season


mean(h_bet$H_Odds)

mean(h_bet$D_Odds)

mean(h_bet$A_Odds)


#Odds for home wins are way lower, so we don't earn that much with right predictions
```