ERASMUS UNIVERSITEIT ROTTERDAM
Erasmus School of Economics

Bachelor Thesis Econometrie en Operationele Research

Forest Estimation for Forecasting Football Results

Abstract
Accurate predictions for a machine learning algorithm are a big decisive factor whether or not the technique sees any empirical use. Moreover, over the last decade an increased interest in football (soccer) has also cultivated to a similar increase in interest in the forecasting of the results. One of the prediction techniques that has gained ground is the use of random forests and variants based on it. In this research the variant introduced by Goller et al. (2018), the ordered random forest algorithm, is used with a database of variables based on past seasons, to simulate the outcomes of every match in the latest German football seasons. The given points are accumulated to make accurate predictions on the final ranking. Additionally, we introduce the neural ordered random forest technique as hybrid on the known algorithm ordered random forest and a neural network, by mapping every tree within the forest to a neural network structure. Although, it's performance is outclassed in some areas compared to the original technique, multiple adaptions are suggested for future improvement.

Name student: Ties Maasdam
Student ID number: 457787

Supervisor: N.F.S. Dijkstra
Second assessor: P.J.F. Groenen

Date final version: July 7, 2019

Table of Contents

1 Introduction

A popular delectation of football fans is to place bets on the results of impending matches. As football and betting on said matches has become more and popular over the last decades, there also has been increased interest in forecasting the results of tournaments by the statistical and machine learning field. Football is a highly unpredictable sport, the outcome of matches are dependent on many unknown variables where luck can be the decisive factor in the end. This makes for a perfect study ground for testing the performance of models in areas where not all explanatory variables are available and where it is hard to optimize the usage of the known data. Whereas, the performance can immediately be measured against other competitive models by computing the performance of the predictions against the bookmakers.

The statistical model that has seen a lot of adoption in the football match outcome prediction field is a Poisson regression model (Schauberger and Groll, 2018). The main advantage of the model, is the assumption that the amount of goals each team scores during a match is independent of the opposing team. This relaxation may not be the most appropriate but it eases the use of the model as there is no dependencies of one team in respect to the performance on the opposing team. Therefore an optimal performance can be achieved by identifying significant dependent variables and using those in the Poisson regression.

In the machine learning department there have been multiple algorithms that have been getting attention. An advantage of these techniques is that they do not make the vital assumption the Poisson model does, thus creating an environment where expected performance between teams can be measured. Joseph et al. (2006) have shown using Bayesian network that, while using a relatively small sample size, accurate forecasts can be made. Under the prerequisite that the model is built by someone adept in the field for the variable selection. Similarly, Loeffelholz et al. (2009) presented multiple versions of neural networks to predict match outcomes in the NBA. Their algorithm managed to predict the winning team on a higher average then a team of basketball experts. Another algorithm which has shown potential is random forests. Its usefulness is shown, for example, by Groll et al. (2018), who correctly predict the winning team in the 2014 FIFA world cup, even though another team was the evidently favored team. Furthermore, Schauberger and Groll (2018) presented that different variants of random forests algorithms outperform conventional regressions in terms of accuracy. They also have shown that random forests show to perform on par or even better as a multitude of well renowned bookmakers.

Since random forests reliably perform well, we will further look into random forests and possible alterations. That brings us to our main research question, namely: *"Should we adapt random forest estimation to get a good performance in respect to football match forecasting?"*. To answer this research question we theorize multiple sub-questions: *"How can we adapt random forest estimation in a feasible way, such that performance is improved?"* and *"How can random forest estimation techniques perform as well as our benchmark, the bookmakers?"*. A particular adaption which we will delve deeper into is proposed by Lechner and Okasa (2019). The results of a match will be ordered to a loss-draw-win basis. To answer our research question we will follow a similar course as Goller et al. (2018). We gather data of the German Fussball Bundesliga from over 10 years, to predict match outcomes of the latest seasons. The predictions are made on a game-by-game basis, for each match we estimate the probability of each outcome occurring. This way we simulate the whole competition multiple times and rank all the teams within a confidence interval.

Furthermore, using the framework given by Lechner and Okasa (2019), we will map the trees in the random forest to a neural network and further optimize the parameters to create a neural ordered random forest (Biau et al. (2018); Sethi (1990)). By fine tuning the network parameters a higher emphasis is placed on certain variables, which in turn lead to better predictions. The advantage of this approach, over using fully connected neural network, is that there are less parameters to optimize thus less data points are needed. This is favourably, since neural networks often require a big sample to make accurate predictions. Similarly, the framework allows use to preserve the robustness and to an extent the interpretability of random forests.

The rest of this research is structured as follows. Section 2 is dedicated to the origin of the data set and the adaptions made. We will discuss the methods used in our research in Section 3. In Section 4 we show the results of the empirical research we did. In Section 5 we give our conclusion on this research and in Section 6 we hold a discussion on topics for future researches and possibilities for enhancements.

2 Data Collection

Our research is based on the German football competition the Bundesliga. We collect data based on the 08/09 season is collected up until the current season, 18/19. This totals up to elevens full seasons, for a total of 3366 observations (matches). To make proper predictions, we collect a multitude of variables, which will be discussed below.

To make an estimation of how well a team can perform, we gather data on the structure of the team and the club, the extended list of variables can be found in the Appendix. Features that are used are, age and height structure of the team and market value of the players. As source we use www.transfermarkt.com, whose accuracy is discussed by Franck and Nüesch (2012) and Bryson et al. (2013). Due to time constraints we opted to only update these variables on a season by season basis, thus players leaving the team on a halfway point are recorded to stay at the club for the whole season and new players joining the club during the mid season are recorded as have been with club from the start. This may cause possible inconsistencies as players can play for two clubs at the same time, but we do not expect the impact of this to be significant. Furthermore, when computing variables based on the 11 best players of a team, we firstly base the ranking on the MV of the players, then if there are any, the first missing data point within the top 11 is replaced by the 12th' best player, the second missing data point by the 13th', etcetera. Then, if Q replacement of data took place, but we also need to compute somethings based on the 12th to 21th player for the same variable, we compute this ratio based on the $(12 + Q)$-th to $(21 + Q + F)$-th player, where F is the number of data points that were missing here to create this variable.

We also keep track of a variable based on where a match is played. Using data gathered from www.google.com/maps, we calculate the distance in time with a car from stadium to stadium for the outgoing team and capacity of the stadiums based on www.wikipedia.com. Similarly, we collect data on the social-economic circumstances, like GDP, based on where the clubs are situated, from www.regionalstatistik.de.Furthermore, the importance of the matches can be retrieved by using the television revenues accumulated from matches. We use the data found on www.fernsehgelder.de for the season from 12/13 to 18/19. The data points before season 12/13 are computed based on a regression of the revenue of the latter seasons.

Another set of variables that we use are based on the relation between Bundesliga matches and those of other competitions. From www.fifa.com we gather data based on when the world cup takes place to mark the corresponding seasons. Furthermore, by using the schedules from www.uefa.com and www.kicker.de, we get the data for which teams and when matches for the Champions League are played out. The schedule of the Bundesliga itself is taken from www.football-data.co.uk. The same source is used to get the statistics based on the performance of the teams based on their records in the previous season. If a team is new to the Bundesliga because of a promotion, we base their match statistics on the average of the worst 3 teams of the previous season. We also use the same source to measure how well our algorithms perform. By looking at the performance our algorithms achieve by the bookmakers we can measure how well the algorithm functions. We collect the promised returns from the following six bookmakers to measure our performance, Bet&Win (BW), Interwetten (IW), Pinnacle (PS), William Hill (WH) and VC BET (VC). The promised returns are based on the Friday afternoon odds for the weekend matches and the odds from Tuesday afternoon are used for the matches during the week.

3 Methodology

This section is used to review the current literature of random forest estimation and its deviation, the ordered random forest estimation. Furthermore, the theoretical groundworks will be laid for extensions on the existing literature. Finally, it will be described how these methods are used to replicate Goller et al. (2018), and expanded upon, using the framework given by Biau et al. (2018).

3.1 Random forest

Initially Breiman (2001) introduced random forests as an extension of the ideas proposed by Ho (1998). Since then, random forests have become a powerful tool for predictions in the machine learning field. The concept of random forests is to grow a multitude of single trees and randomly aggregate them. Over these aggregated trees the average is taken for regressions or a majority vote for classification. Here a regression tree is defined as a tree with as response value a probability distribution of the possible outcomes, whereas, a classification tree is a tree with a class as response value (Breiman, 1984).

To reduce the possibility of overfitting to the data, we make use of bootstrap aggregation (bagging) for each tree. Bagging is it randomly picking of data points from our data with replacement, such that a new data set is created to build the tree upon. The advantage of bagging is that variance between the trees gets reduced.

We create each tree independently by repeatedly make a binary split on one of the end-nodes (leaves). We pick the leaf with the lowest Gini coefficient as the potential leaf to split. The Gini coefficient is calculated by

$$Gini = 1 - \sum_{t=0}^{t=k} P_t^2,$$

where k are the possible classes and P_k is the percentage of observations in that class. To decrease the variance of outcomes between the trees further, (Friedman et al., 2001) proposed to only use subset of the possible covariates at each split. As such, we randomly draw c out of C possible covariates without replacement. We take the recommended value $c = \sqrt{C}$ for our research. We then iterate over all of the c covariates. The variable with corresponding value which can split the leaf in such a way, that the weighted Gini coefficient of the suggested splitted data is the lowest, is considered as the next splitting variable. This is under the condition that the potential split on the leaf does indeed reduce the Gini coefficient compared to the coefficient of the leaf itself. The splitting of the leaves is repeated until a minimum amount of $L(x)$ leaves is attained or the predictor space cannot be split to reduce the gini index given the covariates, where $L(x)$ is the leaf which accomadates x. The goal of the splits is to partition the predictor space in such a way that, the regression values within a partition are similar, but the values in different splits differentiate by a substantial amount.

Thus, the algorithm takes a sample b of size N via bootstrapping, with the b data points a regression tree $T_b(x)$ is created, where x is the the evaluation point. The tree is grown by, at each split, choosing c out of the C possible covariates, to base a split on. Until the minimum amount of $L(x)$ total leaves is attained. Thus, the accumulation of B regressions trees $T_b(x)$ gives the random forest estimate $RF^B(x)$ as

$$RF^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \quad \text{with} \quad T_b(x) = \frac{1}{|\{i : X_i \in L(x)\}} \sum_{\{i:X_i \in L(x)\}} Y_i.$$

Here we define the covariates and the outcomes as X_i and Y_i, respectively.

3.2 Ordered random forest estimation

To make full use of all given data, we have to take into account the structure of football match results. In other words, the outcome of football matches can be classified as a problem with an ordered outcome of lose, draw or win, whereby taking full effect of this categorization the accuracy can be increased. The usage of a model for predicting an ordered outcome in football is not often used in the literature, with a few exceptions like Hothorn et al. (2006) and Hornung (2017). We will follow the proposed method by Lechner and Okasa (2019) to include the ordered outcomes accurately in our model, as is fully explained in Goller et al. (2018). The model provides us with outcomes probabilities and marginal effects for each category. All while handling the higher dimensional variable spaces by taking full advantage of the random forest framework.

A more technical explanation of the ordered random forest estimator will now be discussed. We define an ordered outcome variable which has m ordered classifications possible as $Y_i \in \{1, ..., M\}$. Furthermore, a sample size N, with $i = 1, ..., N$, is considered. The classification probabilities of the different outcomes evaluated at x, in other terms $P[Y_i = m|X_i = x]$, are estimated given a set of binary indicators $Y_{m,i} = \underline{1}(Y_1 \leq m)$ for $m = 1, ..., M-1$ by computing the cumulative probabilities. In order to perform this estimation, a random forest has to be created for each of the $M-1$ binary indicators, thus giving $\hat{Y}_{m,i} = \hat{P}[Y_{m,i} = 1|X_i = x]$. Creating these $M-1$ random forest estimations can be a computational demanding task, but since we only make use of three classes in our framework the size of the problem is diminished.

Because the summation of the cumulative probabilities has to be one, the M-th class' probability does not have to be calculated, since it must hold that $\hat{Y}_{M,i} = 1$. Afterwards, for every m categories and for all i the probabilities are computed via the cumulative probabilities. The outcome is directly taken from the ordered random forest estimation for the first category, where we define $\hat{P}^{tot}_{1,i} = \hat{Y}_{1,i}$. All subsequent classes have their individual probabilities estimated by making use of the preceding categories. In other words, by taking the probability estimated for the m-th class minus the probability for the preceding class, the probability of the secluded class can be calculated as $\hat{P}^{tot}_{m,i} = \hat{Y}_{m,i} - \hat{Y}_{m-1,i}$. For continuity we define that, if a probability becomes negative because of this computation we set its value to zero. To put it another way, if $\hat{P}^{tot}_{m,i} < 0$ then $\hat{P}^{tot}_{m,i} = 0$. To make sure the summation of the final probabilities equals to one, we rescale all the probabilities of the categories $m = 1, ..., M$ as

$$\hat{P}_{m,i} = \frac{\hat{P}^{tot}_{m,i}}{\sum_{m=1}^{M} \hat{P}^{tot}_{m,i}},$$

here the probabilities $\hat{P}_{m,i}$ are defined as the conditional ordered outcome probabilities $\hat{P}_{m,i} = \hat{P}[Y_i = m|X_i = x]$. Therefore, we now have probabilities for all the different categories.

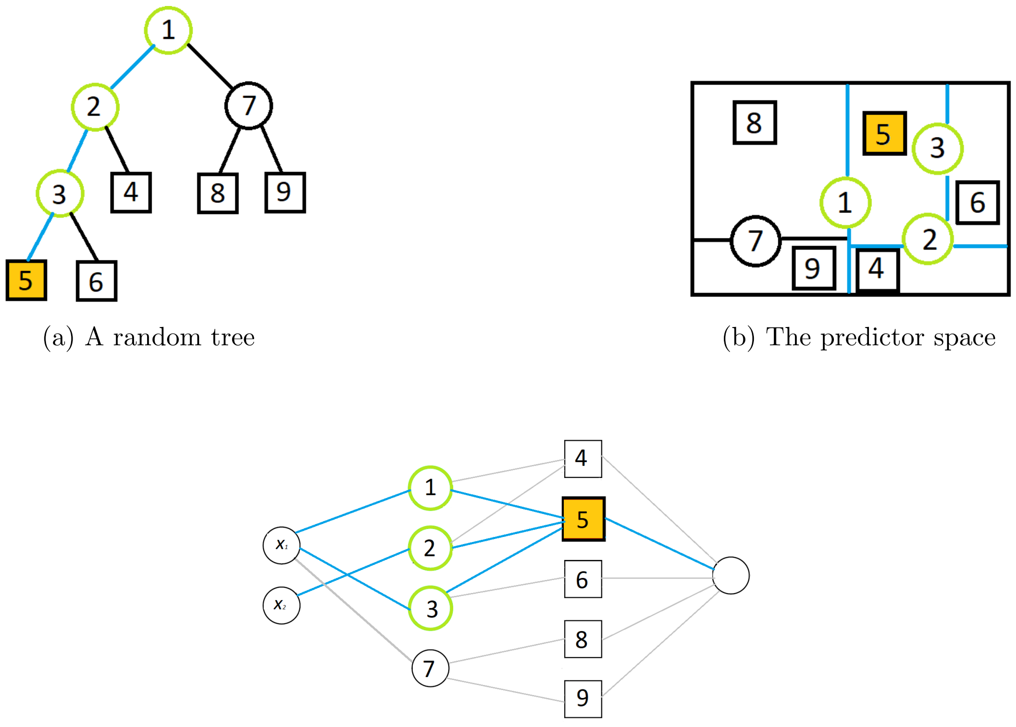
3.3 Neural network adaption of the ordered random forest

As extension on the current literature we suggest the usage of a neural ordered random forest. This extends on both the theories described in Section 3.1 and 3.2 by making use of the methods explained by Biau et al. (2018). The main idea is to make an one-on-one mapping between a tree and a neural network (Welbl (2014); Richmond et al. (2015)) and combine this with the use of ordered random forests (Lechner and Okasa, 2019) to make optimal predictions. We will first discuss how a tree is mapped to a neural network. Secondly, how we need to make small changes to the neural network to make backpropagation possible.

3.3.1 Mapping a tree to a neural network

For every tree in the random forest a mapping is made between the nodes in the tree to a neural network. The neural network is given three hidden layers, where, for every of the inner tree nodes, a neuron is connected in the first hidden layer. In the second hidden layer a neuron is placed for every leaf node. The connections (perceptrons) between the first two hidden layers are placed such that every node which is passed in the tree between the root and a certain leaf, which all have a node in the first hidden layer, are connected to a leaf node in the second hidden layer.

A visualization of this can be seen in Figure 1. The main idea is to minimize an empirical mean squared error from the predictions of this network, over an bagged sample. If the extra training and fine tuning of the network will lead to better predictions than ordered random forest shall be seen in Section 4.



(a) A random tree

(b) The predictor space

(c) Neural network

Figure 1: Mapping of a tree from the predictor space to a neural network as in accordance with Biau et al. (2018).

First hidden layer

To create the same structure as a classical neural network we have to define certain characteristics. At every split in the generated tree, we divide the data in relation to the presented variable value. This splits the predictor space with the use of hyperplanes $\mathscr{H} = \{H_1, ..., H_{K-1}\}$, here we have $K-1$ hyperplanes as we define K as the total number of leaves in the tree. Here we define that the data points x^j which fall below the variable value α_j are given a $+1$ code and the others a -1 code, for a given dimension $j \in \{1, ..., d\}$, where d is the number of total variables. Then, each $H_k \in \mathscr{H}$ is specified as $\{x \in [0,1]^d : h_k(x) = 0\}$, where $[0,1]^d$ is one of the subsets attained after a split and $h_k(x) = x^{j_k} - \alpha_{j_k}$ for a $j_k \in \{1, ..., d\}$. Thus, the activation function of the neurons in the first hidden layer is defined as $\tau(h_k(x)) = \tau(x^{j_k} - \alpha_{j_k})$, with given threshold function $\tau(u) = 2\,\mathbb{1}_{u \geq 0} - 1$. Furthermore, the vector of weights in the first hidden layer is a binary indicator vector for a variable j_k.

Altogether, for every neuron in the first hidden layer there is a corresponding inner tree node, and consequently a split. Thus, for a given input x, there is a neuron which records the relative position to the corresponding split. Therefore the output of the first layer can be described as a vector ± 1 vectors $(\tau(h_1(x)), ..., \tau(h_{K-1}(x)))$. This vector construes all decisions made within the inner tree nodes, therefore, the value of $\tau(h_k(x))$ is $+1$ when x falls on one side of the hyperplane H_k or -1 when it is on the other side. Since, all neurons in the first layer are connected to only one input $x^{(j_k)}$ with a weight of 1, the leaf of x can be quite trivially determined in the second layer.

Second hidden layer

The leaf node of x is determined by using the output of the first layer, the $(K-1)$-vector of ± 1-bits. This is done by using adequate thresholds with a mix of the weighted bits. We define $\mathscr{L} = \{L_1, ..., L_K\}$ as the set of all the leaves in a tree. In the second hidden layer we use one neuron for each leaf in the tree, that brings us to having K neurons in the second hidden layer. A perceptron is created between neuron k from the first hidden layer to k' in the second hidden layer if and only if the hyperplane H_k is associated with the path taken during the sequence of splits when following a path from the root the leaf L_k. Thus, the output of the first hidden layer is seen as a vector $(u_1(x), ..., u_K(x))$, where $u_i(x)$ is a vector of ± 1-bits.

For a given neuron k' in the second hidden layer, the output $v'_k(x) \in \{-1, 1\}$ is $\tau(\sum_{k \to k'} b_{k,k'}$ $u_k(x) + b^0_{k'})$, here we define $k \to k'$ as a perceptron between k and k' and $b_{k,k'} = \pm 1$ is their respective weight. The offset $b_k^0 = -l(k') + \frac{1}{2}$, where the length of the path from the root to L_k' is $l(k')$. Hence, the value of the threshold functions is $\frac{1}{2}$ if $x \in L_{k'}$ and otherwise it is lower than $-\frac{1}{2}$. Consequently, output $v_{k'}(x) = 1$ if and only if for x the leaf node is $L_{k'}$. Altogether the output of the second hidden layer can be seen as the vector $(v_1(x), ..., v_K(x))$ of ± 1-bits, which are equal to -1 except for the bit with a value of +1 corresponding to the leaf $L(x)$.

Output layer

In the output layer we make use of the output of the second hidden layer, vector $(v_1(x), ..., v_K(x))$. The output layers computes the average $\bar{Y}_{k'}$ of the Y_i given the X_i which fall into $L_{k'}$. Another notation for this is $t_n(x) = \sum_{k'=1}^{K} w_{k'} v_{k'}(x) + b_{out}$, with $w_{k'} = \frac{1}{2}\bar{Y}_{k'}$ for all $k' \in \{1, ..., K\}$, and $b_{out} = \sum_{k'=1}^{K} \bar{Y}_{k'}$. Here we see that the baseline of the outcome is based on b_{out}, the summation of all possible outcomes dictated by all the leaves. Here all the leaves are weighted the same, regardless of how many data points landed in the leaf during the original tree creation.

3.3.2 Backpropagation

Every tree in the random forest $RF^B(x)$ is now mapped to a neural network with $K_n - 1$ neurons in the first hidden layer and K_n neurons in the second. We now consider the q-th tree $T_q(x)$, we know that the structure of the network is fixed, since after the tree building phase in the random forest algorithm the trees do not change anymore, thus, the weights and offsets of the layers will be fixed as well. To increase the performance of our forest, we will now try to let the weights vary by performing backpropagation training. The parameters are tuned by trying to minimize an empirical mean squared error over a sample which is randomly taken without replacement over our data set. This sample is different for each tree to avoid overfitting to the data.

A condition of backpropagation training is that the activation functions must be differentiable, thus, the original function $\tau(u) = 2\,\mathbb{1}_{u \geq 0} - 1$ gets replaced with a similar function which can be differentiated. This new function is the hyperbolic tangent activation function $\sigma(u) : \tanh(u) = \frac{e^{2u} - 1}{e^{2u} + 1}$. A different $\sigma(u)$ is used for the two hidden layers. In the first hidden layer $\sigma_1(u) = \sigma_1(\gamma_1 u)$ and in the second hidden layer $\sigma_2(u) = \sigma_2(\gamma_2 u)$ for all neurons. The hyperparameters γ_1 and γ_2 are strictly positive and decide on how quickly the transition from -1 to 1 goes. A larger γ means that the conversion goes at a faster rate and as γ approaches infinity, so will the activation function converge to the original threshold function.

Backpropagation is a gradient descent algorithm. Which comes down to updating the weights based on the $\mathscr{L}(X)$ empirical error of the neural network based on a data point x_i in a bagged data set. During the bagging of this data set, the data points that were already taken for the validation of the neural network are excluded. This is to avoid too much bias to the performance of a bagged set in relation to the validation set. First the error rate is calculated following $\mathscr{L}(X) = (Y - \hat{Y})^2$, here Y is the true outcome of X and $\hat{Y}$ is the probability the network estimated for the given outcome. Then, using the error rate each weight w_i in the vector of weights in the second hidden layer is updated with $w_i = -L(x_i) * (u_i(x)) * v_i''(x) * \delta$, where $v_i''(x)$ is the derivative of the $v_i'(x)$ and δ is the learning rate which usually take the scale of 10^{-5}. The derivative of $v_i''(x)$ is based on $\sigma(u)$ which has the derivative function $1 - \sigma(u)^2$ and can therefore be easily computed. This backprogagation is repeated for many bagged data sets based on the *epoch* hyperparameter. One *epoch* is to calculate the error rate and perform the backpropogation once. In the end the weights of the *epoch* are chosen that give lowest total error rate in respect to the validation data set. The weights of the first hidden layer are not updated, this is to keep the structure of the tree intact to allow for a certain extent of interpretability of the otherwise black box method of neural networks.

3.4 Estimating match outcomes with random forests

Now we will discuss how these methods are used in the empirical research. In order to look at the performance of the algorithms, we will use the gathered football data from the season 08/09 to 16/17 to create the ordered random forests and neural ordered random forests, respectively. As the neural forest requires standardized data, we use standardized data sets for both forest algorithms.

For both ordered random forests and neural ordered random forests, we create 30 trees with 100 leaves each per forest and the size of the bagged data needed for creating a tree is halve of our total training data set size. Only halve the data size is used to speed up the progress of the forest creation, as with some experimentation showed that the performance did not improve with a bigger data size. Furthermore, for ordered neural random forests, we perform 100 epochs a tree with $\frac{1}{9}$ of our training data as validation data and $\frac{1}{3}$ of the training data size for the bagged data for parameter optimization. Additionally, we use a learning rate of 10^{-4} and the values 100 and 1 are initialized for γ_1 and γ_2, respectively. These parameters are chosen in accordance with Biau et al. (2018), while keeping in mind total runtime and making sure that tree creation algorithm is cut short such that the leaves do have an uncertainty in them to a certain extent.

The estimated forests are used to predict the outcomes from every match in both season 17/18 and 18/19. Then, based on the probabilities of each outcome occurring and a randomly drawn number between 0 and 1, a final result is chosen. This is to recreate some variance between the outcomes as seen in football. Given the final results predicted for every match, points are distributed for the teams, 0 points for a loss, 1 for a draw and 3 for a win. The seasons are both simulated a total of 15 times, this to create less bias to the randomness that can occur. For each team all the points are aggregated and divided by the total amount of simulations to get the final standings for each team with an estimated number of total points. We also calculate the standard deviation of these standings to have certain confidence interval.

Furthermore, we also use the predicted outcomes to compare our performance to the bookmakers. We calculate our performance for each bookmaker individually given the different betting strategies. For each of the strategies applies that, if a bet is placed, the total amount per bet for a match is standardized to a total of 1. The returns of the bets are aggregated over the whole season, and finally the average and standard deviation are computed of the accumulation of returns of all the simulation.

The first betting strategy, *Always bet proportionally* (ABP), is to bet on every match proportional to the odds predicted with the random forest. Thus, if we predict the odds of the outcomes to be 0.6, 0.2 and 0.2, for a home win, a draw and home lose, respectively, we also bet on the outcomes with 0.6, 0.2 and 0.2 per category, respectively. So if the true outcome is a home win and the return by the bookmaker for that is 4, we compute the return for that match to be $(0.6 * 4) - 1 = 1.4$.

A variant on the first strategy *Always bet proportionally net revenues* (ABPNR), tries to account for the margin of profit the bookmakers use on all the returns. We thus try to rescale the bookmaker odds to calculate the net earnings as it were, for a fairer comparison to the odds the bookmakers predict. Unfortunately, we do not know how the bookmakers distribute the margins. Therefore, for simplicity's sake we assume that they distribute the margins proportionally to the odds. For a discussion on this topic we refer to Levitt (2004), Paul and Weinbach (2008) and Paul and Weinbach (2012). To calculate the net earnings we invert the bookmaker's returns, to get the odds these returns imply, next we standardize these odds to sum up to 1 and finally we invert these odds again to receive the net returns.

The last strategy, *Bet only for expected positive yield* (BOFEPY), is based on the idea that we only will bet on matches where the predicted earnings of the match are lower than the risk taken. As we are betting a total of 1 each match, we at least want the return in of the categories multiplied with the predicted odd to exceed that. For example, we predict the odds to be 0.6, 0.2 and 0.2 with the bookmaker returns set at 1.04, 1.25 and 6, respectively. Then, as $0.2 * 6 = 1.2 > 1$, we will perform the bet. If none of the categories would have an expected yield higher than 1, no bet would be placed.

4 Results

In this section we will present the different results found in our research. In Section 4.1 we discuss the results found of predicting the seasons' outcomes using the ordered random forest algorithm. Similarly, in Section 4.2 the results of the predictions of the neural ordered random forest are presented. Lastly, Section 4.3 provides a better look into what variables have had a high importance during the tree building phase.

4.1 Ordered random forest

4.1.1 Point predictions

The predicted points per team and the actual points for the 17/18 season are reported in Table 1. Similarly, the results of the 18/19 season can be found in the Appendix. We observe that the majority of the points per club can be forecasted reasonably well. Although, The standard deviation of the points per team is rather high for each team. This can be explained by the uncertainty we tried to maintain by drawing a semi-random outcome proportionally to the predicted odds and the relatively small simulation count.

Furthermore, we notice that the upper echelon of teams have rather accurate predictions, but in the lower halve there are a few teams who performed a bit above expectations and vice versa. This falls within our expectations, because a certain set of variables only updates on a seasonal basis, so certain on-seasons developments are not taken into account for. Not to mention, the data used for teams that are just promoted is based on the previous season's worst teams, therefore the predictions can differ quite a bit from the realized points.

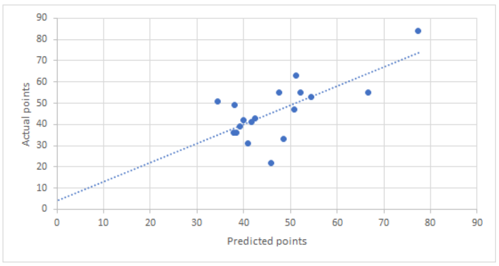
In Figure 2 a more clear visualization is presented of the difference between expected and the actual gained points per team for both predicted seasons. The figure shows that, although

Table 1: Ordered random forest prediction and realization of final standings in the Bundesliga 17/18 season.

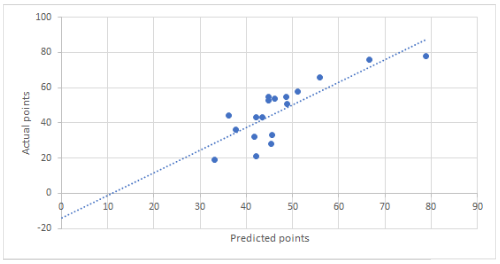
Team	Predicted points	Actual points
FC Bayern Munich	77.4 (7.0)	84
Borussia Dortmund	66.7 (6.5)	55
RB Leipzig	54.5 (4.9)	53
Bayer 04 Leverkusen	52.2 (6.4)	55
FC Schalke 04	51.2 (7.2)	63
Mönchengladbach	50.8 (7.7)	47
VfL Wolfsburg	48.6 (5.7)	33
TSG 1899 Hoffenheim	47.7 (7.2)	55
1. FC Köln	45.9 (7.2)	22
Hertha BSC	42.4 (4.9)	43
FC Augsburg	41.6 (7.5)	41
Hamburger SV	40.9 (8.4)	31
Werder Bremen	39.9 (8.6)	42
Hannover 96	39.2 (7.5)	39
SC Freiburg	38.4 (6.9)	36
Eintracht Frankfurt	38.1 (9.2)	49
1. FSV Mainz 05	37.8 (7.2)	36
VfB Stuttgart	34.3 (7.2)	51

The standard deviation is given within brackets.

there are some teams that deviate from the regression line, mostly the teams have an accurate prediction. Such that, the regression line has a slope close to 1 and cuts the x-axis close to 0 in both seasons. This means that on average the points gained per team are pretty accurately predicted, but there is some variance deviating from the trend on both sides of the spectrum, indicating that there is not a specific bias. Additionally, it is notable that the larger deviations from the regression line mostly occur in the 17/18 season.



(a) 17/18 Season



(b) 18/19 Season

Figure 2: Ordered random forest predictions against actual points of Bundesliga teams.

4.1.2 Betting strategies

The resulting returns from the different betting strategies for both predicted seasons are given in Table 2. Immediately it becomes clear that the ABP strategy, gives quite bad returns. An obvious cause is that the returns from the bookmakers are not profitable against the risk taken. So when we account for the net returns with ABPNR, we do see a positive return in the latter season. The likely cause is that the accuracy is higher in the latter season than the former. As the accuracies are 40.3% (0.02%) and 41.6% (0.02%), with the standard deviation given within brackets, respectively. Which could be argued as that the latter season is more predictable than the former, as there are also less big deviations from the predicted points to the actual points earned.

The last betting strategy considered, BOFEPY, also performs well according to the tables. With a positive average return by all the bookmakers for both seasons, with even more outstanding numbers in the latter season. Which in turn are also likely caused by the higher accuracy. Since the standard deviation is still relatively large even when we have positive returns, there will be a decent part of a confidence intervals that will result into negative returns. Thus, if this betting strategy is used there is still a modest probability for negative returns.

Table 2: Returns in percentage of betting strategies by different bookmakers over the 17/18 and 18/19 Bundesliga seasons of the ordered random forest.

Season	Strategy	B365	BW	IW	PS	WH	VC
17/18	ABP	-17.6 (3.6)	-18.2 (3.5)	-18.4 (3.6)	-10.5 (3.6)	-19.7 (3.5)	-15.4 (3.5)
	ABPNR	-2.7 (3.8)	-3.3 (3.7)	-2.3 (3.8)	-3.7 (3.7)	-2.2 (3.7)	-4.1 (3.6)
	BOFEPY	0.0 (1.6)	0.4 (1.1)	0.7 (1.2)	0.3 (1.8)	0.2 (1.1)	0.4 (1.4)
18/19	ABP	-12.3 (3.0)	-12.1 (2.9)	-12.9 (3.0)	-5.3 (3.0)	-13.7 (2.9)	-9.1 (3.0)
	ABPNR	3.3 (3.1)	2.9 (3.1)	2.7 (3.1)	4.5 (3.1)	2.5 (3.1)	2.6 (3.1)
	BOFEPY	2.6 (1.7)	2.1 (1.6)	1.9 (1.6)	1.9 (1.9)	0.4 (1.4)	1.8 (1.6)

The standard deviation is given within brackets.

4.2 Neural ordered random forest

4.2.1 Point and rank predictions

Like for the ordered random forest the results of the predicted points and the actual points for the 17/18 season are presented in Table 3, with the results for 18/19 found in the Appendix. Take note that the predicted points per team are all fairly close to each other. This is caused by the specification of b_{out}, which is biased to being equally distributed over the outcomes. A more in depth discussion on this topic is found in Section 6. Additionally, we take note of the standard deviation being decently high due to us taking a semi-random outcome proportionally based on the predicted odds, to preserve a part of the randomness found in typical football match outcomes and our use of only a limited number of simulations.

Table 3: Neural ordered random forest prediction and realization of final standings in the Bundesliga 17/18 season.

Team	Predicted points	Actual points
FC Bayern Munich	58.4 (5.4)	84
Borussia Dortmund	55.1 (6.3)	55
RB Leipzig	52.9 (10.3)	53
FC Schalke 04	52.5 (8.0)	63
Bayer 04 Leverkusen	51.9 (9.9)	55
Mönchengladbach	51.5 (7.0)	47
TSG 1899 Hoffenheim	50.7 (8.54)	33
Eintracht Frankfurt	50.4 (10.2)	55
FC Augsburg	49.9 (7.3)	22
Hamburger SV	47.3 (8.4)	43
1. FC Köln	46.7 (5.5)	41
VfL Wolfsburg	46.6 (8.9)	31
Hertha BSC	46.4 (5.6)	42
1. FSV Mainz 05	46.3 (9.6)	39
VfB Stuttgart	44.7 (9.7)	36
Werder Bremen	44.7 (5.6)	49
SC Freiburg	44.4 (9.4)	36
Hannover 96	43.4 (7.3)	51

The standard deviation is given within brackets.

As the point distribution is close to each other and, similarly to the ordered random forest, the standard deviation being decently high, point predictions may not be the most suitable use of the current implementation of the neural ordered random forest. This made us look at another performance measure of the algorithms, the mean squared error.

Table 4 shows for the 17/18 season, the predicted rank for both algorithms, the actual rank and the squared error between those. In the Appendix a similar table is shown for season 18/19. We notice that the neural ordered random forest has a lower mean squared error than it's ordered random forest counterpart. The mean squared error being 16.2 and 24.7, respectively. The lower error is mainly caused by reducing the error for the few rankings with a big error. Similarly, the 18/19 season also sees a lower mean squared error for the neural forest over the other algorithm. The mean squared error values for the neural algorithm is 10.4 and for the ordered forest it is 10.9. These lower mean square error could indicate a substantive way to make use of the current model. By, instead of making point prediction, making ranking predictions of the final tournament rankings.

Table 4: Predicted rankings and the squared error of the 17/18 Bundesliga season.

	Predicted rank			Squared error	
Team	Ordered	Neural	Actual rank	Ordered	Neural
FC Bayern Munich	1	1	1	0	0
Borussia Dortmund	2	2	4	4	4
RB Leipzig	3	3	6	9	9
Bayer 04 Leverkusen	4	5	5	1	0
FC Schalke 04	5	4	2	9	4
Mönchengladbach	6	6	9	9	9
VfL Wolfsburg	7	12	16	81	16
TSG 1899 Hoffenheim	8	7	3	25	16
1. FC Köln	9	11	18	81	49
Hertha BSC	10	13	10	0	9
FC Augsburg	11	9	12	1	9
Hamburger SV	12	10	17	25	49
Werder Bremen	13	16	11	4	25
Hannover 96	14	18	13	1	25
SC Freiburg	15	17	15	0	4
Eintracht Frankfurt	16	8	8	64	0
1. FSV Mainz 05	17	14	14	9	0
VfB Stuttgart	18	15	7	121	64

4.2.2 Betting strategies

We present the results of the betting strategies for the predicted seasons in Table 5. The main take away visible is that the results are proportionally the same as those of the ordered random forest, but worse across the board. For instance the ABP strategy sees a decrease of about 5% return over all the bookmakers, but also a decrease in the size of the standard deviation. We argue this is caused by the lower accuracy of the neural ordered random forest. The accuracy is 36.3% (0.03%) in the 17/18 season and 37.9% (0.04%) for the 18/19 season, where the standard deviation is given within brackets. A decrease in accuracy of about 4% for each season. Similar to ABP, the returns of ABPNR observe about the same trend, resulting into that there is only one positive average return, for bookmaker Pinnacle in the 18/19 season.

The BOFEPY strategy also sees a decrease in returns, with some of the returns going down 8%, while others see a decrease of about 3%. To top it off, the standard deviation sees an increase for all bookmakers. We argue this is caused by the specification of b_{out}, which causes a bias to equally distributed odds over the outcomes. The biased odds can randomly add additional matches to bet on. Resulting into a worse performance, as the bets are not that favourable, and more variety in total returns.

Table 5: Returns in percentage of betting strategies over the 17/18 and 18/19 Bundesliga seasons of the neural ordered random forest.

Season	Strategy	B365	BW	IW	PS	WH	VC
17/18	ABP	-21.5 (2.2)	-23.9 (2.4)	-23.9 (2.0)	-15.6 (2.5)	-24.2 (2.1)	-21.9 (2.5)
	ABPNR	-6.8 (2.3)	-9.2 (2.5)	-8.1 (2.2)	-8.9 (2.6)	-6.9 (2.2)	-10.9 (2.6)
	BOFEPY	-5.1 (2.8)	-4.8 (2.7)	-3.0 (2.3)	-5.0 (2.6)	-3.9 (3.3)	-5.9 (3.5)
18/19	ABP	-16.1 (1.8)	-16.7 (1.9)	-19.0 (1.8)	-8.6 (1.9)	-18.5 (1.9)	-14.1 (2.0)
	ABPNR	-0.7 (1.9)	-2.0 (1.9)	-3.7 (1.9)	0.3 (1.9)	-2.6 (2.0)	-2.5 (2.1)
	BOFEPY	-5.8 (3.3)	-6.4 (3.6)	-7.4 (2.6)	-4.1 (3.0)	-7.2 (4.1)	-5.0 (3.7)

The standard deviation is given within brackets.

4.3 Variable selection

An important factor of the performance of a random forest algorithm, is if there is an adequate pool of explanatory variables to pick from. Thus, by identifying which variables are often picked during the tree building phase and which variables often occur in the same branch, we can classify the importance of certain sets of variables. For this, all the variables that were used to perform a split within the tree were tracked, this data was used over all the simulations to calculate the average and the standard deviation. The resulting table can be found in the Appendix.

The average of the variables used showed that the most important set of variables are those which differentiate the two teams in terms of size of the club. As a bigger club in terms of revenue can buy more expensive players and can give better support to those players, they are also more likely to win a match. Therefore, it falls within expectations that the most picked variables refer mostly to the difference in, the amount of TV revenue earned per year, the total market value of the club and a mix of those. But we also see that variables like, the unemployment rate in the area and the Log GDP are important. These can indicate multiple things, but probably the most important factor is that there seems to be a correlation between bigger clubs and well-off areas.

Additionally, the variables that are arguably not connected to the club size specifically but the team itself, like share of left foot players and the amount of fouls, yellow cards and goals in a previous season etcetera, play only an average amount of play during the splitting phase. This is most likely caused by that these stats do not directly or indirectly imply the size of the club nor if a team is good or bad. On the contrary, boolean variables for traditional and yo-yo teams, or teams participating in the Champions League were barely picked at all. Although, these do indicate the size of the club to a certain extend, they do not allow for subtle difference between the clubs, so they would see the most use during splitting when there are only a few matches left to consider. As we cut off the splitting progress of the leaves early to allow for a distribution of odds within a leave, these variables are not favoured to be picked.

5 Conclusion

It is exceedingly important for machine algorithms to be able to make accurate predictions in highly unpredictable fields. We thus set out to develop a good variant on the random forest algorithm, which can make accurate predictions in the uncertain field of football predictions. Therefore, in this research we address the research question: *"Should we adapt random forest estimation to get a good performance in respect to football match forecasting?"*. To get a satisfactory answer we created the following sub-questions: *"How can we adapt random forest estimation in a feasible way, such that performance is improved?"* and *"How can random forest estimation techniques perform as well as our benchmark, the bookmakers?"*.

To answer the first sub-question, we dived deeper into two things. Firstly, improvement of the random forest algorithm through adaptions, and secondly, variable importance. If we first consider the former, we adapted the random forest algorithm to the ordered random forest and the neural forest algorithm. Then, through the results we came across that it is important how we define the performance. The ordered random forest managed to make rather accurate point predictions per team. On the other hand the neural ordered random forest had a lower value in terms of the mean squared error in ranking the teams over the predicted seasons.

When we consider variable importance to improve forest estimation, we noticed in the results that certain sets of variables are the most prevailing during data splitting. The variables that saw the most use were those, which could capture the difference in size of the clubs the best. Furthermore, as a result of cutting off the tree growing process early, binary variables that showed the difference between only certain sets of teams, saw almost no use. Therefore, we proclaim that more variables that indicate the size of the clubs but also team features could lead to better predictions.

The latter sub-question concerned, we use the two random forest adaptations and different betting strategies to identify the performance of the adaptations. In the results we saw that the ordered random forest performed decent in comparison to the bookmakers, when always a bet was placed proportional to the odds predicted while trying to account for the profit the bookmakers make on each bet. Additionally, when a different betting strategy was used, which only places a bet when the expected returns on one of the outcomes is positive, we saw positive average returns over all the bookmakers. The neural ordered random forest performed overall worse for all the betting strategies, netting up to only seeing a positive average return on one of the bookmakers when accounting for their profit margin.

To tackle the main question, we combine the results of the sub-questions. We measure the performance of two different alterations of the random forest. In the results we find that the two implementations excel in different things. The neural ordered random forest performs better when trying to predict the ranking of the teams in the tournament, while the ordered random forest outperformed the other algorithm in terms of accuracy, point predictions and performance compared to the bookmaker benchmark. Concluding from the results, we find that different alterations of the random forest algorithm have different effects on the given performance measures. Thus, we suggest to delve deeper into the particular algorithms given in this research or other adaptions of the random forest algorithm to find an optimal performance given the desired target performance measure.

6 Discussion

During this research, a multitude of limitations manifested itself. As such, we will now discuss these limitations in our research. Furthermore, we give pointers on to how these challenges can be avoided or solved in future exploration of this research field.

No data based on on-season developments: A major disadvantage of the performance of the algorithms compared to the bookmakers is that the random forests do not make use of the latest season developments. An improvement for future research is to update all the covariates based on all the events that happen, e.g. players cannot participate during upcoming matches due to injuries or suspensions. This allows for more accurate predictions and improvement of the results. However, this would not work if the given data is yet unavailable, when one would want to predict a tournament that is yet to begin. Not to mention, a different way of basing the variables of newly promoted teams, can also lead to more accurate predictions of these teams

Specification of b_{out} in neural forests: A major flaw for our neural forest is likely the specification of b_{out}. The main concern is that it uses the average of the odds of all the leaves, without any notice to the importance of said leaf. As many leaves held more data points during the training set, that information can be used to put an emphasis on certain leaves when computing b_{out}. As of now the neural networks in the neural forests predict a probability based on the average of the leaves, and not particularly the leaf where the data point would end up on. This leads to the odds per outcomes being biased to being equally distributed over the outcomes. Thus, creating more randomness in the outcomes. A better specification of b_{out} can lead to major improvements in the predictions and we put an emphasis on this as the main topic for future research.

Backpropagation algorithm in neural forests: The backpropagation algorithm used to tune the weight parameters in our research is rather bare bones. Similar to not using the latest tree building techniques, this is due to time constraints during the research. An example of an improved algorithm is Adam proposed by Kingma and Ba (2014). By using multiple data points to update the weights once and having a momentum parameter, the empirical error would converge to a minimum faster. Under the assumption such a minimum exists.

Optimization of weights in the first hidden layer: Another potential improvement for the neural ordered random forest is to let the weights of the first hidden layer update during the backpropagation phase. This would give the opportunity to the neural network to find more hidden correlations between certain features. The drawback is that the ease of interpretability of the tree gets lost, but this would also allow for the adoption of a fully-connected neural network with two hidden layers. Where the tree structure gets used to initialize the starting weights, while also allowing for new connections to be made during the backpropagation phase.

Missing latest developments in forest building techniques: Due to time constraints during the research, it was out of our scope to also implement all of the latest tree building techniques. An example of such a technique is the honest splitting rule introduced by Wager and Athey (2018) and Athey et al. (2019). The honest splitting rule entails that during the phase where the best split is sought, different data is used for measuring what leaf should be split compared to what variable should be used to split the data.

References

S. Athey, J. Tibshirani, S. Wager, et al. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, 2019.

G. Biau, E. Scornet, and J. Welbl. Neural random forests. *Sankhya A*, pages 1–40, 2018.

L. Breiman. *Classification and regression trees*. Routledge, 1984.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

A. Bryson, B. Frick, and R. Simmons. The returns to scarce talent: footedness and player remuneration in european soccer. *Journal of Sports Economics*, 14(6):606–628, 2013.

E. Franck and S. Nüesch. Talent and/or popularity: what does it take to be a superstar? *Economic Inquiry*, 50(1):202–216, 2012.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

D. Goller, M. C. Knaus, M. Lechner, G. Okasa, et al. Predicting match outcomes in football by an ordered forest estimator. Technical report, University of St. Gallen, School of Economics and Political Science, 2018.

A. Groll, T. Kneib, A. Mayr, and G. Schauberger. On the dependency of soccer scores–a sparse bivariate poisson model for the uefa european football championship 2016. *Journal of Quantitative Analysis in Sports*, 14(2):65–79, 2018.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):832–844, 1998.

R. Hornung. Ordinal forests. *Journal of Classification*, pages 1–14, 2017.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674, 2006.

A. Joseph, N. E. Fenton, and M. Neil. Predicting football results using bayesian nets and other machine learning techniques. *Knowledge-Based Systems*, 19(7):544–553, 2006.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

M. Lechner and G. Okasa. Random forest estimation of the econometric ordered choise model. *Unpublished Manuscript*, 2019.

S. D. Levitt. Why are gambling markets organised so differently from financial markets? *The Economic Journal*, 114(495):223–246, 2004.

B. Loeffelholz, E. Bednar, and K. W. Bauer. Predicting nba games using neural networks. *Journal of Quantitative Analysis in Sports*, 5(1), 2009.

R. J. Paul and A. P. Weinbach. Price setting in the nba gambling market: Tests of the levitt model of sportsbook behavior. *International Journal of Sport Finance*, 3(3):137, 2008.

R. J. Paul and A. P. Weinbach. Does sportsbook. com set pointspreads to maximize profits? *The Journal of Prediction Markets*, 1(3):209–218, 2012.

D. L. Richmond, D. Kainmueller, M. Y. Yang, E. W. Myers, and C. Rother. Relating cascaded random forests to deep convolutional neural networks for semantic segmentation. *arXiv preprint arXiv:1507.07583*, 2015.

G. Schauberger and A. Groll. Predicting matches in international football tournaments with random forests. *Statistical Modelling*, 18(5-6):460–482, 2018.

I. K. Sethi. Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78 (10):1605–1613, 1990.

S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

J. Welbl. Casting random forests as artificial neural networks (and profiting from it). In *German Conference on Pattern Recognition*, pages 765–771. Springer, 2014.

A Data appendix

Table 6: All variables used during the research.

Variable	Reference to	Unit	Update
Season ID	-	category	yearly
TV revenue	home	euro	yearly
TV revenue	away	euro	yearly
TV revenue difference	home-away	euro	yearly
Club market value	home	euro	yearly
Club market value	away	euro	yearly
Club market value difference	home-away	euro	yearly
Club market value / TV revenue	home	euro	yearly
Club market value / TV revenue	away	euro	yearly
Club market value / TV revenue difference	home-away	euro	yearly
Club market value - TV revenue	home	euro	yearly
Club market value - TV revenue	away	euro	yearly
Club market value - TV revenue difference	home-away	euro	yearly
Market value share	home	euro	yearly
Standardized market value	home	euro	yearly
Standardized market value	away	-	yearly
Standardized market value difference	home-away	-	yearly
Unemployment rate	home	percentage	yearly
Unemployment rate	away	percentage	yearly
Unemployment rate difference	home-away	percentage	yearly
Log GDP	home	euro	yearly
Log GDP	away	euro	yearly
log GDP difference	home-away	euro	yearly
Previous game earned points	home	numerical	match
Previous game earned points	away	numerical	match
Previous game earned points share of total	home	percentage	match
Previous game earned points share of total	away	percentage	match
Impending Champions League match (2 weeks or less)	home	boolean	match
Impending Champions League match (2 weeks or less)	away	boolean	match
Just had Champions League match (2 weeks or less)	home	boolean	match
Just had Champions League match (2 weeks or less)	away	boolean	match
Champions League this season	home	boolean	yearly
Champions League this season	away	boolean	yearly
Current season is before world cup	-	boolean	yearly
Current season is after world cup	-	boolean	yearly
Stadium capacity	-	numerical	once
Distance away team travelled	-	numerical	once

Table 6 continued from previous page

Variable	Reference to	Unit	Update
Time travelled away team	-	minutes	once
Previous season goals	home	numerical	yearly
Previous season points	home	numerical	yearly
Previous season goals at half time	home	numerical	yearly
Previous season shots	home	numerical	yearly
Previous season target shots	home	numerical	yearly
Previous season fouls	home	numerical	yearly
Previous season corners	home	numerical	yearly
Previous season yellow cards	home	numerical	yearly
Previous season red cards	home	numerical	yearly
Previous season goals	home	numerical	yearly
Previous season points	home	numerical	yearly
Previous season goals at half time	away	numerical	yearly
Previous season shots	away	numerical	yearly
Previous season target shots	away	numerical	yearly
Previous season fouls	away	numerical	yearly
Previous season corners	away	numerical	yearly
Previous season yellow cards	away	numerical	yearly
Previous season red cards	away	numerical	yearly
Previous season points home - points away	home-away	numerical	yearly
Previous season goals home - goals away	home-away	numerical	yearly
Market value average difference	home-away	euro	yearly
Market value standard deviation difference	home-away	euro	yearly
Ratio top 3 to top 12-14 market value	home	ratio	yearly
Ratio top 3 to top 12-14 market value difference	home-away	ratio	yearly
Ratio top 11 to top 12-21 market value	home	ratio	yearly
Ratio top 11 to top 12-21 market value difference	home-away	ratio	yearly
Age mean difference	home-away	numerical	yearly
Age standard deviation difference	home-away	standard deviation	yearly
Age top 11 difference	home-away	numerical	yearly
Ratio top 11 to top 12-21 age difference	home-away	ratio	yearly
Number of players older than 20 years difference	home-away	numerical	yearly
Minimum aged player difference	home-away	numerical	yearly
Maximum aged player difference	home-away	numerical	yearly
Share left foot difference	home-away	percentage	yearly
Share two feet difference	home-away	percentage	yearly
Share left foot in top 11 difference	home-away	percentage	yearly
Share two feet in top 11 difference	home-away	percentage	yearly
Height mean difference	home-away	numerical	yearly
Height standard deviation difference	home-away	standard deviation	yearly
Height mean top 11 difference	home-away	numerical	yearly

Table 6 continued from previous page

Variable	Reference to	Unit	Update
Height standard deviation top 11 difference	home-away	standard deviation	yearly
Yo-yo club	home	boolean	once
Yo-yo club	away	boolean	once
Other club	home	boolean	once
Other club	away	boolean	once
Traditional club	home	boolean	once
Traditional club	away	boolean	once

B Results 18/19 season appendix

Table 7: Ordered random forest prediction and realization of final standings in the Bundesliga 18/19 season.

Team	Predicted points	Actual points
FC Bayern Munich	78.9 (6.3)	78
Borussia Dortmund	66.7 (7.9)	76
RB Leipzig	56.0 (7.1)	66
Bayer 04 Leverkusen	51.1 (6.0)	58
TSG 1899 Hoffenheim	48.9 (6.8)	51
Mönchengladbach	48.7 (8.0)	55
Eintracht Frankfurt	46.2 (7.7)	54
FC Schalke 04	45.5 (8.8)	33
VfB Stuttgart	45.5 (7.6)	28
VfL Wolfsburg	44.9 (7.9)	55
Werder Bremen	44.7 (6.7)	53
Hertha BSC	43.5 (8.8)	43
1. FSV Mainz 05	42.2 (7.5)	43
Hannover 96	42.1 (6.2)	21
FC Augsburg	41.8 (8.4)	32
SC Freiburg	37.7 (7.2)	36
Fortuna Düsseldorf	36.3 (5.8)	44
1. FC Nuremberg	33.2 (6.2)	19

The standard deviation is given within brackets

Table 8: Neural ordered random forest prediction and realization of final standings in the Bundesliga 18/19 season.

Team	Predicted points	Actual points
FC Bayern Munich	57.3 (9.5)	78
Borussia Dortmund	53.3 (9.7)	76
RB Leipzig	52.7 (7.7)	66
TSG 1899 Hoffenheim	51.9 (8.4)	51
Bayer 04 Leverkusen	51.5 (7.3)	58
1. FSV Mainz 05	49.5 (10.6)	43
Eintracht Frankfurt	49.3 (6.9)	54
Mönchengladbach	49.3 (7.3)	55
Werder Bremen	49.2 (7.4)	53
FC Schalke 04	48.8 (6.0)	33
Fortuna Düsseldorf	48.3 (7.9)	44
Hertha BSC	48.0 (5.9)	43
FC Augsburg	47.7 (11.0)	32
VfL Wolfsburg	47.7 (8.4)	55
VfB Stuttgart	47.6 (8.0)	28
1. FC Nuremberg	44.9 (6.3)	19
Hannover 96	44.3 (6.3)	21
SC Freiburg	43.6 (10.0)	36

The standard deviation is given within brackets

Table 9: Predicted rankings and the squared error of the 18/19 Bundesliga season

	Predicted rank			Squared error	
Team	Ordered	Neural	Actual rank	Ordered	Neural
FC Bayern Munich	1	1	1	0	0
Borussia Dortmund	2	2	2	0	0
RB Leipzig	3	3	3	0	0
Bayer 04 Leverkusen	4	5	4	0	1
TSG 1899 Hoffenheim	5	4	9	16	25
Mönchengladbach	6	8	5	1	9
Eintracht Frankfurt	7	7	7	0	0
FC Schalke 04	8	10	14	36	16
VfB Stuttgart	9	15	16	49	1
VfL Wolfsburg	10	14	6	16	64
Werder Bremen	11	9	8	9	1
Hertha BSC	12	12	11	1	1
1. FSV Mainz 05	13	6	12	1	36
Hannover 96	14	17	17	9	0
FC Augsburg	15	13	15	0	4
SC Freiburg	16	18	13	9	25
Fortuna Düsseldorf	17	11	10	49	1
1. FC Nuremberg	18	16	18	0	4

C Variable selection appendix

Table 10: Accumulation of variable selection during the tree building phase over the simulation. Given for the ordered random forest (ORF) and neural ordered random forest (NORF), for both forest 1, which encapsulates the forest that predicts the home team win versus draw or loss, and forest 2, which predicts home win or draw versus loss

	forest 1 mean (st. dev.)		forest 2 mean (st. dev.)	
Variable (full names found in Table 6)	ORF	NORF	ORF	NORF
Season ID	27 (5.7)	35 (6.9)	34 (7.6)	29 (5)
TV revenue	50 (3.9)	45 (11.1)	46 (17.7)	53 (6.2)
TV revenue	33 (17.9)	42 (5.6)	53 (5.1)	66 (15.2)
TV revenue diff.	73 (7.5)	75 (11.3)	100 (10.4)	81 (12.7)
Club MV	46 (5.8)	48 (9.9)	58 (8.3)	50 (7.3)
Club MV	41 (7.7)	48 (10.6)	46 (8.2)	56 (8.6)
Club MV diff.	74 (10)	68 (13.5)	70 (13.5)	78 (7.7)

Table 10 continued from previous page.

	forest 1 mean (st. dev.)		forest 2 mean (st. dev.)	
Variable (full names found in Table 6)	ORF	NORF	ORF	NORF
Club MV / TV revenue	53 (10.4)	48 (7.9)	40 (15.3)	45 (8.6)
Club MV / TV revenue	57 (9.8)	38 (9)	49 (8.4)	53 (8.9)
Club MV / TV revenue diff.	67 (7.3)	66 (7.1)	54 (13.3)	69 (8.2)
Club MV - TV revenue	37 (7.7)	39 (8.4)	46 (4.9)	59 (13.6)
Club MV - TV revenue	42 (5.8)	49 (5.5)	40 (6.2	32 (13.1)
Club MV - TV revenue diff.	70 (14)	75 (10.1)	67 (9.9)	50 (10)
MV share	35 (8.9)	41 (15.1)	50 (8.9)	57 (16.5)
Standardized MV	44 (10.5)	36 (13.3)	35 (10.8)	37 (9.8)
Standardized MV	46 (6)	39 (8)	53 (15.1)	42 (6.2)
Standardized MV diff.	66 (15.3)	55 (7.5)	75 (20.8)	64 (11)
Unemployment rate	33 (9.4)	50 (8.4)	53 (9.4)	49 (3.7)
Unemployment rate	41 (10.2)	42 (18.1)	39 (6.3)	58 (15.5)
Unemployment rate diff.	57 (8.9)	48 (11)	57 (10.1)	53 (7)
Log GDP	44 (5.3)	37 (4.7)	51 (7.1)	45 (9.5)
Log GDP	58 (12.9)	56 (8)	31 (12.6)	44 (7.5)
log GDP diff.	59 (9)	75 (7)	47 (11.7)	56 (4.3)
Pg. earned points	10 (5.3)	19 (4.2)	15 (3.2)	9 (7.7)
Pg. earned points	17 (5.4)	19 (4.4)	15 (4.5)	10 (6.6)
Pg. earned points share of total	68 (7.4)	72 (11.4)	70 (7.9)	72 (8.5)
Pg. earned points share of total	62 (8.1)	64 (6.8)	67 (13.4)	49 (9.8)
Im. Champions League match	2 (1.1)	2 (1.2)	2 (1.2)	6 (3.9)
Im. Champions League match	1 (1.6)	3 (2.6)	2 (1.2)	4 (2.4)
Past Champions League match	6 (4.1)	2 (0.9)	2 (1.4)	3 (2.3)
Past Champions League match	3 (1.7)	3 (1.2)	2 (2.7)	0 (3)
Champions League this season	5 (2.9)	3 (2.2)	0 (2.2)	4 (2.6)
Champions League this season	2 (1)	1 (2)	4 (2.6)	4 (2.2)
Current season is before world cup	4 (1.8)	3 (2.1)	4 (2)	2 (1.9)
Current season is after world cup	2 (3.6)	0 (3.9)	6 (2.9)	3 (2.6)
Stadium capacity	22 (8.2)	15 (7.3)	35 (7.7)	36 (7.6)
Distance away team travelled	54 (7.8)	56 (11)	55 (7.2)	60 (8.1)
Time travelled away team	48 (6.3)	36 (16.4)	50 (5.5)	58 (10.8)
Ps. goals	30 (5.6)	23 (6.1)	25 (6.6)	44 (13.5)

Table 10 continued from previous page.

	forest 1 mean (st. dev.)		forest 2 mean (st. dev.)	
Variable (full names found in Table 6)	ORF	NORF	ORF	NORF
Ps. points	33 (6)	34 (7.2)	40 (8.3)	37 (5.1)
Ps. goals at half time	27 (6.3)	22 (5.3)	36 (7.6)	24 (9)
Ps. shots	36 (8.7)	37 (8.8)	32 (7.8)	38 (6.1)
Ps. target shots	31 (6.5)	39 (7.7)	38 (8)	38 (7.4)
Ps. fouls	37 (5.2)	28 (10)	25 (11.6)	39 (7.1)
Ps. corners	35 (7.3)	44 (7.8)	26 (10.3)	34 (6.9)
Ps. yellow cards	38 (6.4)	40 (12.6)	32 (5.5)	24 (10.7)
Ps. red cards	20 (3.4)	18 (6.3)	25 (6.4)	21 (3.6)
Ps. goals	28 (5.3)	30 (6.8)	27 (5.4)	33 (7.5)
Ps. points	28 (7.5)	41 (5.4)	33 (6.5)	29 (3.5)
Ps. goals at half time	26 (4.7)	21 (12)	19 (7.9)	24 (6.3)
Ps. shots	36 (5.3)	50 (11.4)	34 (5.7)	36 (6.9)
Ps. target shots	40 (7.1)	34 (12.5)	31 (5.8)	26 (9.4)
Ps. fouls	36 (11.4)	37 (8.1)	35 (7.9)	24 (10.2)
Ps. corners	23 (14.2)	29 (8.5)	25 (6.6)	36 (7.3)
Ps. yellow cards	35 (5.5)	38 (8.9)	27 (5.2)	28 (5.3)
Ps. red cards	15 (5.1)	16 (4.8)	17 (4)	17 (4.2)
Ps. points home - points away	41 (8.5)	47 (6.2)	43 (6)	33 (13.1)
Ps. goals home - goals away	44 (7.8)	33 (7.6)	41 (6.1)	34 (10.5)
MV average diff.	55 (7.6)	49 (8)	44 (10.9)	47 (7.6)
MV st. dev. diff.	45 (10)	58 (7.9)	42 (9.5)	48 (4)
Ratio top 3 to top 12-14 MV	33 (2.3)	35 (6.1)	30 (6.9)	28 (4.9)
Ratio top 3 to top 12-14 MV diff.	45 (8.5)	50 (6.3)	42 (8.4)	47 (6.2)
Ratio top 11 to top 12-21 MV	27 (14.1)	38 (19)	36 (4)	37 (7.3)
Ratio top 11 to top 12-21 MV diff.	53 (9.2)	56 (7.9)	44 (8)	44 (7.8)
Age mean diff.	56 (8)	60 (12.7)	46 (4.9)	41 (8.1)
Age st. dev. diff.	56 (8)	54 (9.3)	53 (10)	41 (8.3)
Age top 11 diff.	48 (5.1)	35 (7.4)	47 (8.7)	35 (8.6)
Ratio top 11 to top 12-21 age diff.	49 (5.6)	53 (5.6)	54 (9.4)	54 (11.5)
N. of players older than 20 years diff.	44 (12)	27 (9.4)	32 (5.7)	28 (11.4)
Minimum aged player diff.	11 (5.4)	14 (4.3)	13 (3.7)	12 (5.1)
Maximum aged player diff.	9 (6.7)	13 (7.1)	11 (3.7)	17 (5.3)

Table 10 continued from previous page.

	forest 1 mean (st. dev.)		forest 2 mean (st. dev.)	
Variable (full names found in Table 6)	ORF	NORF	ORF	NORF
Share left foot diff.	47 (9.3)	41 (7.5)	45 (5.6)	60 (19.2)
Share two feet diff.	39 (11.3)	34 (6.3)	57 (10.8)	48 (8.3)
Share left foot in top 11 diff.	20 (6.6)	27 (9.3)	22 (6.3)	27 (8.5)
Share two feet in top 11 diff.	15 (3.6)	11 (7.6)	14 (4.6)	16 (3.5)
Height mean diff.	46 (3.9)	43 (10)	55 (8.6)	44 (8.4)
Height st. dev. diff.	56 (10.2)	58 (7.9)	53 (9.3)	38 (11.5)
Height mean top 11 diff.	42 (5)	38 (5.9)	38 (5.4)	37 (6.4)
Height st. dev. top 11 diff.	55 (10)	47 (10.7)	45 (4.2)	48 (7.9)
Yo-yo club	2 (1.3)	3 (1.4)	3 (2)	1 (2.7)
Yo-yo club	4 (1.9)	3 (2.2)	2 (0.9)	0 (2.3)
Other club	0 (1.8)	1 (2.2)	4 (2.6)	3 (1.2)
Other club	7 (3.8)	5 (3)	0 (2)	2 (1.5)
Traditional club	4 (1.9)	3 (2.1)	3 (1.5)	2 (2.2)
Traditional club	4 (1.7)	0 (2)	1 (2.5)	0 (2)

D Programming code appendix

In the attachment all the programs written are given. These programs are rather long, therefore, we opted to only include the description of all the files here.

D.1 Variable creation

scraper.java

Scrapes team data from www.transfermarkt.com using the JSoup package. This data is written to csv files which are read in makevariables.java.

makevariables.java

Uses data received from the csv files made in scraper.java to create the variables used in this research. These variables are then written to a new set of csv files, which are used in csv-maker.java.

csvmaker.java

Uses the variables made in makevariables.java, but as they were not in the same structure as the rest of the database yet, rewrites the order the variables are given to match the structure with the rest of the database. These structured variables are then written to a final csv file, which content can be directly added to the rest of the database.

D.2 Random forests

Tree.java

Class which creates a tree instance. It contains a number of functions to manage the structure of the tree, it also can set and retrieve data regarding the tree.

randomForest.java

This class performs basically all the computations regarding a random forest. During the tree building phase it computes where and with what data splits have to be made, this information is passed on to the tree instance in Tree.java. Additionally, it contains a set of functions to compute and return performance measures in regard to the forest created.

main.java

Reads the database files and uses functions in randomForest.java to create random forests which are used as a ordered random forest. Furthermore, it retrieves the performance measures and returns it as the output.

NeuralNetwork.java

Class functions as pseudo-extension on the tree instance. The class also adds an additional set of functions similar to randomForest.java to perform backpropagation on the network and to compute performance measures based on neural network.

mainNN.java

Similar to main.java, but now uses the random forests created as neural ordered random forests.