Universiteit
Leiden
Wiskunde en
Natuurwetenschappen

Setting up a workflow for an explorative Random Forest Machine Learning Model

Felix Kapulla

Daily thesis advisor: Erik van Zwet. Associate Professor, medical statistics
University thesis advisor: Matthijs Tijink. Senior Data Consultant at CQM

Master Thesis
Statistics and Data Science
Leiden University

Contents

1 Introduction

2 Methods

In this section, an introduction to CART decision trees is given since these are the building blocks of the random forest algorithm. Furthermore, it will be discussed how standard errors of random forest predictions can be estimated by means of the jackknife-after-bootstrap procedure. This procedure will be manipulated to estimate the covariance between random forest predictions which is eventually needed to estimate the standard error of a simple defined main and low-order interaction effect. Also theoretical concepts about model-agnostic methods such as a permutation variable importance measure and partial dependence plots are elaborated upon since these will be output components of the final ML workflow.

2.1 CART Decision Tree

Decision trees are methods for supervised learning and were made popular by Breiman et al. [2] with CART. They can be used for both regression and classification problems. The main idea is to recursively partition the input space with binary splits into a set of non-overlapping rectangles and then fit a simple model (e.g. the average) in each region [3]. For a regression task the goal is to find rectangles $R_1, R_2, ..., R_J$ that minimize the loss function stated in equation 1:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{1}$$

This loss function is the sum of residual sums of squares (RSS) in each region, where $\hat{y}_{R_j}$ is the mean response of the training observations within the j-th box. Splitting the input space is done in a top-down, greedy and binary fashion. Top-down means that starting point is a single region containing all training observations and then successively split this region. The CART algorithm is greedy because at each step the predictor space is split based on a variable and a threshold value that leads to the greatest reduction in RSS, but only at that particular split. The algorithm does not look ahead and does not regard splits that might lead to lower reduction in RSS at future steps. Finally, binary splitting means that the predictor space is split based on one predictor X_j and one cutpoint s into two and only two new regions $R_1(j, s) = \{X|X_j < s\}$ and $R_2(j, s) = \{X|X_j > s\}$.

As shown in equation 2, predictor j and corresponding cutpoint s are chosen that minimize the minimized sums of RSS in the individual regions R_1 and R_2, where c_1 and c_2 are the average outcomes of all training observations in both regions respectively [3].

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \tag{2}$$
$$\text{with } c_1 = \text{ave}(y_i | x_i \in R_1(j, s)) \text{ and } c_2 = \text{ave}(y_i | x_i \in R_2(j, s))$$

Splitting the tree is stopped until a certain criterion is met, e.g. until all terminal leaves contain at least a specific amount of observations. After fitting the tree, a prediction for a new observation can be made by computing the mean response of the region in which the new observation falls. For a classification task a majority vote within a region is made to classify a new observation. An example of a fitted regression tree with two predictors and four splits is shown in figure 1. The right figure gives an idea about how the predictor space is partitioned into rectangular regions while the left figure shows the corresponding tree structure.

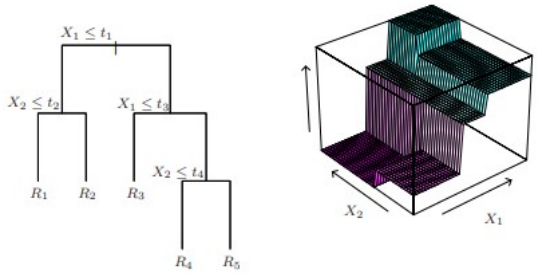

Figure 1: An example taken from [3]. A CART regression tree with four splits is fitted to a dataset with two predictors. At each step i, the variable and corresponding threshold t_i are selected that lead to the greatest reduction in RSS. The right figure shows how the predictor space is partitioned into rectangular regions while the left figure shows the corresponding tree.

In general, the tree growing procedure may have a good fit on the training data since it is able to capture complex interactions in the data. However, a single tree method is likely to overfit on the training set since it is a very flexible method, characterized by low bias but high variance. Small changes in the data may lead to a completely different fitted tree. Figure 2 illustrates this bias variance trade-off for three different data sets that are characterized by different levels of non-linearity [4]. Generally, a more flexible specification of a model leads to an increase in variance (orange line) and a reduction in bias (blue line). However, the optimal model flexibility that leads to the lowest test MSE varies across the the data sets since the rates at which variance and bias change differ.

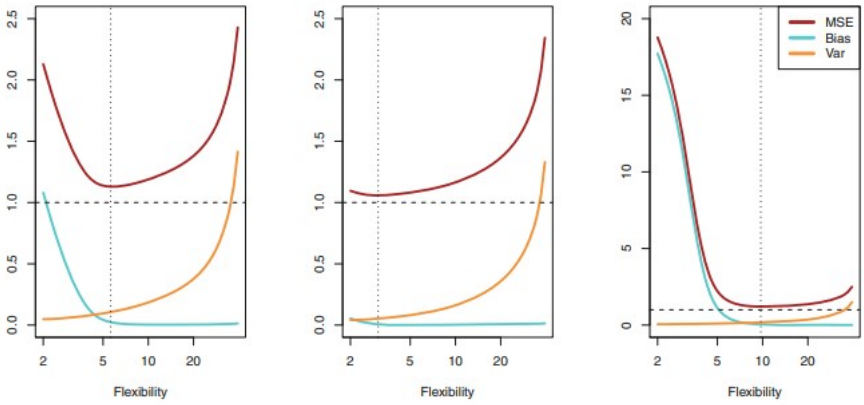

Figure 2: Illustration of Bias-Variance Trade-off for three different data sets taken from [4]. For the first data set there is almost a linear relationship between predictor and outcome variable while for the first data set the relationship is more non-linear and for the third even more. In all three cases, the variance (orange line) increases and the bias (blue line) decreases as the method's flexibility increases. However, the optimal flexibility level that leads to the lowest test MSE differs across the data sets. This is because the squared bias and variance change at different rates.

2.2 Random Forest

A random forest is an ensemble learning method that builds a large collection of de-correlated decision trees. Firstly, multiple bootstrap replicates of the original training set are formed by

sampling with replacement which serve as new training sets. On each bootstrapped training set a decision tree is fitted. To obtain a final prediction for an observation in a regression setting, all individual tree predictions are averaged. This approach is also called bootstrap aggregation (bagging). The essential idea is to average many noisy but approximately unbiased models and hence reduce variance [1]. Decision trees are a suitable building block for bagging since they are, if grown sufficiently, approximately unbiased but are characterized by relatively large variance. Furthermore, decision trees are identically distributed in bagging. Hence, the expectation of the average of such B trees is the same as the expectation of any individual tree. However, this is not the case for the variance and in terms of prediction performance the hope is to reduce it by averaging the trees. [1] Formally, a bagged tree predictor can be defined as in equation 3:

$$\hat{\theta}(x) = \frac{1}{B}\sum_{b=1}^{B} t_b^*(x, Z_{b1}^*, ..., Z_{bn}^*) \tag{3}$$

Here, t_b^* is a fitted tree on the b-th bootstrap sample $\left(Z_{b1}^*, ..., Z_{bn}^*\right)$ containing n training observations, where observations are drawn with replacement. To obtain a final prediction for observation x, all B individual tree predictions for x are averaged.

In addition to a mere bagging approach, the random forest algorithm also selects a random sample of m features from all possible P features at each step. The tree predictor must then split on one of those selected features. If $m = P$ the random forest becomes a bagged tree. This added randomness allows correlated predictors to be selected as well for the splitting and in general decorrelates the decision trees which should improve the overall prediction performance [3]. Reducing hyperparameter m will decrease the correlation between any pair of the trees and therefore reduce the variance of the average [3]. Formally, an auxiliary noise variable ξ_b is introduced as can be seen in equation 4.

$$\hat{\theta}(x) = \frac{1}{B}\sum_{b=1}^{B} t_b^*(x, \xi_{kb}, Z_{b1}^*, ..., Z_{bn}^*) \tag{4}$$

A great benefit of random forests is the direct estimation of a generalization error. This is possible because decision tree learners are fitted on identically distributed bootstrap samples of the training data in sequence. Hence, an out-of-bag error estimate ($\hat{OOB}$) can be computed according to equation 5 at every addition of a new tree.

$$\hat{OOB} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i^{OOB} - y_i\right) \tag{5}$$

Here, $\hat{y}_i^{OOB}$ is the average prediction for observation i from all fitted trees so far where i has not been part of. Doing this for each training observation and averaging all differences between the i-th OOB-estimate and its true outcome y_i leads to an OOB test error estimate along the fitting procedure.

2.2.1 Estimating Variable Main Effects And Corresponding Standard Errors

For any machine learning model $\hat{f}(\cdot)$, a simple effect size of predictor variable j could be defined and estimated by fixing some point in the middle of the predictor space and shift the coordinate of interest by some amount to form two new points A and B.

$$\hat{A} = (\hat{\mu}_1, \hat{\mu}_2, ..., \hat{\mu_j} - k \cdot \hat{\sigma}_{x_j}, ..., \hat{\mu}_P) \tag{6}$$

$$\hat{B} = (\hat{\mu}_1, \hat{\mu}_2, ..., \hat{\mu_j} + k \cdot \hat{\sigma}_{x_j}, ..., \hat{\mu}_P) \tag{7}$$

[1] Assume that B independently and identically distributed (i.i.d.) random variables are given, each with variance σ^2. The average of those random variables has variance $\frac{1}{B}\sigma^2$. If variables are only identically distributed with positive pairwise correlation ρ, the variance of the average is $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$

Here, the ML model $\hat{f}(\cdot)$ approximates the true functional relationship between P predictors and an outcome variable and $\hat{\mu}_j$, $\hat{\sigma}_j$ are estimates of the true mean and standard deviation of the j-th predictor variable respectively. Hence, $\hat{A}$ and $\hat{B}$ are estimated from data. An effect of x_j on the conditional mean $E(X \mid Y)$ can be defined as $\frac{f(B) - f(A)}{2 \cdot k \cdot \sigma_{x_j}}$ and estimated by $\frac{\hat{f}(\hat{B}) - \hat{f}(\hat{A})}{2 \cdot k \cdot \hat{\sigma}_{x_j}}$. This variable effect is corrected for possibly non-linear effects in the other variables and local at the mean of all other variables. To obtain a standard error estimate of such variable effect, the following expression needs to be estimated:

$$Var\left[\frac{\hat{f}(\hat{B}) - \hat{f}(\hat{A})}{2 \cdot k \cdot \hat{\sigma}_{x_j}}\right] = \frac{1}{(2 \cdot k \cdot \hat{\sigma}_{x_j})^2}\Big(Var[\hat{f}(\hat{B})] + Var[\hat{f}(\hat{A})] - 2 \cdot Cov(\hat{f}(\hat{A}), \hat{f}(\hat{B}))\Big) \tag{8}$$

Here, $\hat{\sigma}_{x_j}$ is considered to be constant and taken out of the variance expression since the variation of this estimate is negligible. The variance of this variable effect includes the variance of and covariance between random forest predictions. A natural estimator for this quantity is the expression stated in equation 8:

$$\hat{Var}\left[\frac{\hat{f}(\hat{B}) - \hat{f}(\hat{A})}{2 \cdot k \cdot \hat{\sigma}_{x_j}}\right] = \frac{1}{(2 \cdot k \cdot \hat{\sigma}_{x_j})^2}\Big(\hat{Var}[\hat{f}(\hat{B})] + \hat{Var}[\hat{f}(\hat{A})] - 2 \cdot \hat{Cov}(\hat{f}(\hat{A}), \hat{f}(\hat{B}))\Big) \tag{9}$$

2.2.2 Standard Error of Random Forest Predictions

In general, when estimating the standard error of random forest predictions ($Var[\hat{f}(x)]$), one must account for two distinct sources of noise. Firstly, sampling noise and secondly Monte Carlo noise since only a finite number of bootstrap samples is used [6]. However, Wager et al. (2014) suggest two methods to estimate standard errors of random forest predictions. A bias-corrected version of the jackknife-after-bootstrap and the infinitesimal jackknife. For this project the former approach is used which applies the jackknife on top of the inherent Bootstrap samples that were used to fit the random forest. The notation for clarifying the concepts is based on [6]. In general, the jackknife-after-bootstrap estimate for a random forest prediction is defined in equation 10, where an infinite size of bootstrap samples (∞) is assumed.

$$\hat{V}_J^{\infty} = \frac{n-1}{n}\sum_{i=1}^{n}\left(\bar{t}^*_{(-i)}(x) - \bar{t}^*(x)\right)^2 \tag{10}$$

$\bar{t}^*_{(-i)}(x)$ is the expected random forest prediction for x, using only those trees where the i-th observation was not part of a bootstrap sample (out-of-bag), while $\bar{t}^*(x)$ is the expected random forest prediction for x or in other words the average over all tree predictions. However, in practice one can only work with a finite number of B bootstrap samples. The natural Monte Carlo approximation to the estimator above is defined in equation 11

$$\hat{V}_J^B = \frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{\theta}^B_{(-i)}(x) - \hat{\theta}^B(x)\right)^2, \tag{11}$$

$$\text{where } \hat{\theta}^B_{(-i)}(x) = \frac{\sum_{b:N^*_{bi}=0} t^*_b(x)}{|N^*_{bi} = 0|}$$

N^*_{bi} indicates the number of times the i-th observation appears in the b-th bootstrap sample. To obtain a variance estimate for a new observation x, for each observation i it is checked in which bootstrap samples that observation was not part of. Based on the corresponding fitted trees, predictions are made for x and averaged to obtain $\hat{\theta}^B_{(-i)}(x)$. This is equivalent to the computation of the i-th OOB-estimate in equation 5. This value is then subtracted by the random forest prediction

for the observation of interest ($\hat{\theta}^B(x)$) and the difference squared.

However, this finite-B estimate of variance is often badly biased upwards and often dominated by Monte Carlo error [6]. This is why Wager et al. (2014) derives a bias-corrected version of the jackknife estimator shown in equation 12:

$$\hat{V}^B_{J-U} = \hat{V}^B_J - (e-1)\frac{n}{B^2}\sum_{b-1}^{B}\left(t^*_b(x) - \bar{t^*}(x)\right)^2 \tag{12}$$

Next to the bias issue, the simple estimator in 11 requires $B = \Phi(n^{1.5})$ bootstrap samples to reduce Monte Carlo Noise down to the level of the inherent sampling noise, while the bias-corrected version only requires $B = \Phi(n)$ replicates [6].

These findings are summarized in figure 3, where the variance estimates of the jacknife-after bootstrap, the infinitesimal jackknife and its bias-corrected versions are compared for a bagged adaptive polynomial regression example on the cholesterol dataset of Efron and Feldman (1991) with 164 observations. In this figure the distribution of variance estimates are shown as a function of the number of trees B and the horizontal dashed line corresponds to the limiting variance estimate. It can be seen that the uncorrected versions are biased upwards for a small number of B and this bias goes away more slowly than the Monte Carlo variance. Furthermore, the bias-corrected versions seem to fix the bias issue without being unstable. However, another observation is that the bias-corrected infinitesimal estimate has about half the variance of the bias-corrected jackknife-after-bootstrap.

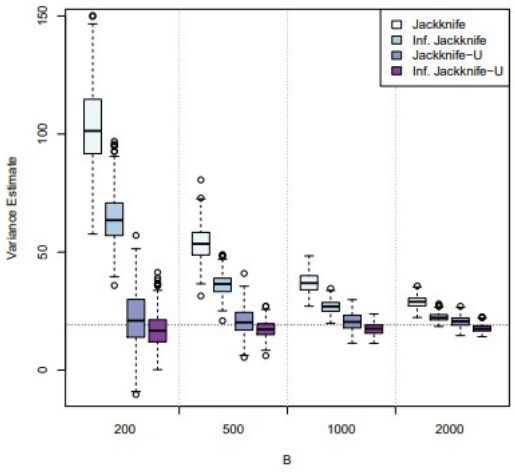

Figure 3: Figure taken from [6]. Shown is the performance of the jackknife-after-bootstrap, the infinitesimal jackknife and its bias-corrected version as a function of the number of trees B. For that a bagged adaptive polynomial regression model was used on the cholesterol dataset of Efron and Feldman (1991).

2.2.3 Covariance between Random Forest Predictions

In order to derive a covariance estimate between random forest predictions, equation 11 is slightly manipulated. Note again, that equation 11 is the natural Monte Carlo approximation to the jackknife-after-bootstrap for a random forest prediction. Furthermore, the bias correction term in equation 12 is used to arrive at a bias-corrected version of a jackknife-after-bootstrap estimate for the covariance between two random forest predictions for points x and x'.

$$\widehat{Cov}_J^B = \frac{n-1}{n} \sum_{i=1}^{n} \left(\hat{\theta}_{(-i)}^B (x) - \hat{\theta}^B (x) \right) \cdot \left(\hat{\theta}_{(-i)}^B (x') - \hat{\theta}^B (x') \right) \tag{13}$$

$$\widehat{Cov}_{J-U}^B = \widehat{Cov}_J^B - (e-1) \frac{n}{B^2} \sum_{b-1}^{B} \left(t_b^*(x) - \bar{t}^*(x) \right) * \left(t_b^*(x') - \bar{t}^*(x') \right) \tag{14}$$

Again, in equation 13 for each training observation i it is checked in which bootstrap samples that observation was not part of. Based on the corresponding fitted trees, predictions are made for x and averaged to obtain $\hat{\theta}_{(-i)}^B(x)$. The same is also done for the second observation of interest x'. These jackknife averages are then subtracted by their corresponding random forest predictions and those quantities multiplied. Hence, this expression is a measure of joint variability of two random forest predictions. To obtain a (NxN) variance-covariance matrix between random forest predictions equation 14 needs to be applied to all pairs of observations.

2.2.4 Random Forest - Low-order Interaction Effect

Similarly to the main effect definition, a low-order interaction effect between two predictor variables can be defined by fixing some point in the middle of the predictor space and shifting the coordinate of interest by some amount. However, now the coordinate of the interaction variable of interest is shifted by some amount as well such that four points are formed in total. Figure 4 visualizes such situation, where the interest lies in the interaction between variables i and j. The blue line corresponds to two data points when predictor variable j is fixed at value $\mu_i - k \cdot \sigma_{x_i}$ and the red line to two data points when predictor variable j is fixed at value $\mu_i + k \cdot \sigma_{x_i}$. According to this illustration the effect of variable i is smaller for a relatively high level of variable j.

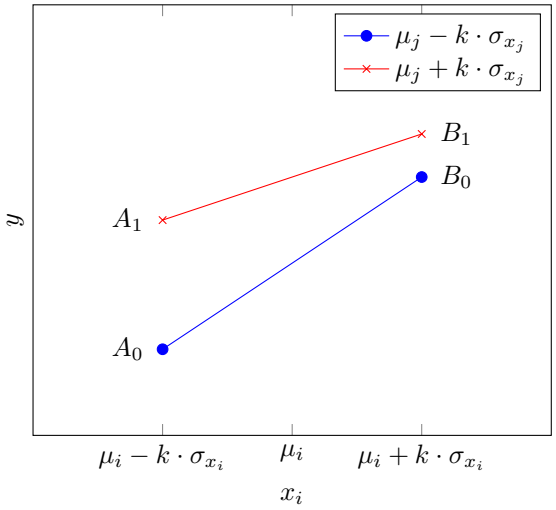

Figure 4: Illustration of Interaction Effect. All predictor variables are centered at their corresponding mean. Shift coordinates of interaction variables of interest (x_i and x_j) by some amount such that four new points A_0, A_1, B_0 and B_1 are formed.

Hence, a simple low-order interaction effect between two variables can be defined as in equation 15. The derivation of the denominator is shown in appendix A. Furthermore, a standard error of such interaction effect can be derived in a similar manner as it was done for the main effect. The derivation is shown in appendix B.

$$\hat{\gamma}_{i,j} = \frac{\left(\hat{f}(\hat{A}_1) - \hat{f}(\hat{A}_0) \right) - \left(\hat{f}(\hat{B}_1) - \hat{f}(\hat{B}_0) \right)}{4 \cdot k^2 \cdot \hat{\sigma}_{x_i} \cdot \hat{\sigma}_{x_j}} \tag{15}$$

2.3 Permutation Variable Importance

Another output element of the workflow is a measure for variable importance. Here, the model-agnostic permutation variable importance (PVI) algorithm based on Fisher, Rudin, and Dominici (2018) is used. The rationale is that permuting values of a feature x_j breaks the association between that feature and the outcome. Hence, if the model relied for its predictions on x_j, then permuting its values would result in an increase of the model's prediction error. Contrary, if the model's prediction error would not increase after permuting values of x_j, then this feature can be considered to be not important for the model [5]. Algorithm 2 shows how the computation of PVI for each variable works.

Algorithm 1 Permutation Variable Importance Algorithm

1: **Input:** Trained Model $\hat{f}(\boldsymbol{X})$, feature matrix $\boldsymbol{X}$, Outcome vector y, MSE Loss Function $\boldsymbol{L}\left(y, \hat{f}(\boldsymbol{X})\right)$
2: Estimate original loss for $\hat{f}(\boldsymbol{X})$: $MSE_0 = \boldsymbol{L}\left(y, \hat{f}(\boldsymbol{X})\right)$
3: **for** Feature $j \in 1, ..., P$ **do**
4: Generate random permutation vector: $\overrightarrow{r}$
5: Generate feature matrix $\boldsymbol{X}_{\overrightarrow{r}}$ by permuting j-th feature according to $\overrightarrow{r}$
6: Estimate new prediction error for $\hat{f}\left(\boldsymbol{X}_{\overrightarrow{r}}\right)$: $MSE_{\overrightarrow{r}} = \boldsymbol{L}\left(y, \hat{f}\left(\boldsymbol{X}_{\overrightarrow{r}}\right)\right)$
7: Compute Permutation Variable Importance as $PVI_j = \frac{MSE_{\overrightarrow{r}}}{MSE_0}$

Firstly, after fitting a ML model $\hat{f}(\boldsymbol{X})$ the benchmark prediction error $\boldsymbol{L}\left(y, \hat{f}(\cdot)\right)$ is computed. For a regression task this is simply the mean squared error. Next, for a given variable j a permutation vector is generated according to which the variable is permuted such that a new input matrix $\boldsymbol{X}_{\overrightarrow{r}}$ is generated. Afterwards, the prediction error is re-estimated for this new input matrix based on the initially fitted model such that PVI_j can be computed as in line 7 of algorithm 2. It is important to mention that such measure does not reflect the intrinsic predictive value of a feature by itself but how important this feature is for a particular model. Hence, to obtain meaningful insights from such measure it is crucial to train a model that generalizes well on unseen data [5].

2.4 Partial Dependence Plots

In order to visualize possibly complex, non-linear relationships between predictors and the predicted outcome variable, partial dependence plots (PDP) can be used. A PDP is a type of global model-agnostic method which shows the marginal effect of usually one or two features on the predicted outcome of any machine learning model [5]. Given a regression task, the partial dependence function is defined according to equation 16.

$$\hat{f}_S(X_S = x_S) = \mathbb{E}_{X_c}\left[\hat{f}(X_S, X_C)\right] = \int \hat{f}(X_S, X_C)\, d\mathbb{P}\left(X_C\right) \tag{16}$$

Here, X_S is the set of variables for which the PDP is to be plotted, while X_C is the complementary set of all other variables. To compute values of the partial dependence function predictions are averaged over the marginal distributions of the complementary set X_C which is done by integration. Hence, the partial dependence function reduces a possibly complex prediction function $\hat{f}$ to a function that is only dependent on usually at most two variables.

Estimating the partial dependence function can be done via Monte Carlo as described in algorithm 2.

Algorithm 2 Estimation of Partial Dependence Function

Input: Trained Prediction Model $\hat{f}(\boldsymbol{X})$, feature matrix $\boldsymbol{X}$, Range of grid values for X_S: x_s^G, N Training Observations
for $x_S \in x_S^G$ **do**
    for i in $1:N$ **do**
        $x_S \rightarrow x_S^{(i)}$
        $\hat{y}_i^{PDP} = \hat{f}\left(x_S, X_C^{(i)}\right)$
    $\hat{f}\left(x_S\right) = \frac{1}{n}\sum_{i=1}^{n} \hat{y}_i^{PDP}$

Firstly, a range of grid values for the set of variables of interest is defined. Then, for each grid value (x_s) and each training observation i its value for x_s is replaced by the grid value while all other features of that instance are hold constant ($x_C^{(i)}$). For this modified observation a prediction is made according to the fitted model $\hat{f}$. The partial dependence value for x_S is estimated by averaging over all predictions.

3 Simulation

3.1 Setup

3.2 Results

4 Implementation of a Workflow for an explorative Random Forest Model

5 Discussion

6 Conclusion

References

[1] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[2] Leo Breiman et al. *Classification and regression trees*. Routledge, 2017.

[3] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[4] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[5] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[6] Stefan Wager, Trevor Hastie, and Bradley Efron. "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1625–1651.

A Recovering Value of Interaction Effect between two variables

Given are variables with some mean mu_j and some variance σ_j^2. To measure an low-order interaction effect between two variables, firstly all predictor variables are centered at their respective means. Only interaction variables of interest are shifted by some amount such that 4 new points are formed. Hence, the true new points A_0, A_1, B_0 and B_1 are formed. A simple low-order interaction effect can be defined as follows:

$$\gamma_{i,j} = \frac{\left(\hat{f}(\hat{A}_1) - \hat{f}(\hat{A}_0)\right) - \left(\hat{f}(\hat{B}_1) - \hat{f}(\hat{B}_0)\right)}{4 \cdot k^2 \cdot \hat{\sigma}_{x_i} \cdot \hat{\sigma}_{x_j}} \tag{17}$$

In order to recover the value $\gamma_{i,j}$, the denominator needs to be taken into account and in the following the derivation of this quantity is given. For this, the simple functional relationship $y_i = \alpha x_1 + \beta x_2 + \gamma(x_1 x_2)$ is assumed. To measure an interaction effect between the two variables, the mean coordinate of x_1 and x_2 are shifted by some multiple of its respective standard deviation such that four new points are formed. Hence, the mean of x_1 is shifted by $k \cdot \sigma_1$ and the mean of x_2 by $k \cdot \sigma_2$.

1. $A_0 = \alpha\left(\mu_1 - k\sigma_1\right) + \beta\left(\mu_2 - k\sigma_2\right) + \gamma\left(\mu_1 - k\sigma_1\right)\left(\mu_2 - k\sigma_2\right)$
2. $A_1 = \alpha\left(\mu_1 - k\sigma_1\right) + \beta\left(\mu_2 + k\sigma_2\right) + \gamma\left(\mu_1 - k\sigma_1\right)\left(\mu_2 + k\sigma_2\right)$
3. $B_0 = \alpha\left(\mu_1 + k\sigma_1\right) + \beta\left(\mu_2 - k\sigma_2\right) + \gamma\left(\mu_1 + k\sigma_1\right)\left(\mu_2 - k\sigma_2\right)$
4. $B_1 = \alpha\left(\mu_1 + k\sigma_1\right) + \beta\left(\mu_2 + k\sigma_2\right) + \gamma\left(\mu_1 + k\sigma_1\right)\left(\mu_2 + k\sigma_2\right)$

According to equation 17 the following expression must be evaluated:

$$\begin{aligned}
\gamma_{i,j} = &\Big(\big(\alpha\left(\mu_1 - k\sigma_1\right) + \beta\left(\mu_2 - k\sigma_2\right) + \gamma\left(\mu_1 - k\sigma_1\right)\left(\mu_2 - k\sigma_2\right)\big) \\
&- \big(\alpha\left(\mu_1 - k\sigma_1\right) + \beta\left(\mu_2 + k\sigma_2\right) + \gamma\left(\mu_1 - k\sigma_1\right)\left(\mu_2 + k\sigma_2\right)\big)\Big) \\
&- \Big(\big(\alpha\left(\mu_1 + k\sigma_1\right) + \beta\left(\mu_2 + k\sigma_2\right) + \gamma\left(\mu_1 + k\sigma_1\right)\left(\mu_2 + k\sigma_2\right)\big) \\
&- \big(\alpha\left(\mu_1 + k\sigma_1\right) + \beta\left(\mu_2 - k\sigma_2\right) + \gamma\left(\mu_1 + k\sigma_1\right)\left(\mu_2 - k\sigma_2\right)\big)\Big) \\
= &\gamma\left(\mu_1 - k\sigma_1\right)\left(\mu_2 + k\sigma_2\right) - \gamma\left(\mu_1 - k\sigma_1\right)\left(\mu_2 - k\sigma_2\right) \\
&- \gamma\left(\mu_1 + k\sigma_1\right)\left(\mu_2 + k\sigma_2\right) + \gamma\left(\mu_1 + k\sigma_1\right)\left(\mu_2 - k\sigma_2\right) \\
= &\gamma\mu_1 k\sigma_2 - \gamma k\sigma_1\sigma_2 - \left(-\gamma\mu_1 k\sigma_2 + \gamma k\sigma_1\sigma_2\right) \\
&- \left(\gamma\mu_1 k\sigma_2 + \gamma k\sigma_1\sigma_2\right) - \gamma\mu_1 k\sigma_2 - \gamma k\sigma_1\sigma_2 \\
= &-4\gamma k^2\sigma_1\sigma_2
\end{aligned}$$

Hence, to recover γ the nominator in equation 17 must be divided by $-4\gamma k^2\sigma_1\sigma_2$.

B Obtaining a standard error estimate for a simple defined low-order interaction effect

To obtain a standard error estimate of a simple defined low-order interaction effect, the following expression needs to be estimated:

$$\hat{Var}\left(\hat{\gamma}_{i,j}\right) = \frac{1}{\left(4 \cdot k^2 \cdot \hat{\sigma}_{x_i} \cdot \hat{\sigma}_{x_j}\right)^2} \quad \hat{Var}\left[\hat{f}(\hat{A}_1) - \hat{f}(\hat{A}_0) - \hat{f}(\hat{B}_1) + \hat{f}(\hat{B}_0)\right] \tag{18}$$

$$= \frac{1}{\left(4 \cdot k^2 \cdot \hat{\sigma}_{x_i} \cdot \hat{\sigma}_{x_j}\right)^2} \quad \Big[\hat{Var}(\hat{f}(\hat{A}_1)) + \hat{Var}(\hat{f}(\hat{A}_0)) + \hat{Var}(\hat{f}(\hat{B}_1)) + \hat{Var}(\hat{f}(\hat{B}_0)) \tag{19}$$

$$-2\hat{Cov}(\hat{f}(\hat{A}_1), \hat{f}(\hat{A}_0)) - 2\hat{Cov}(\hat{f}(\hat{A}_1), \hat{f}(\hat{B}_1))$$

$$+2\hat{Cov}(\hat{f}(\hat{A}_1), \hat{f}(\hat{B}_0)) + 2\hat{Cov}(\hat{f}(\hat{A}_0), \hat{f}(\hat{B}_1))$$

$$-2\hat{Cov}(\hat{f}(\hat{A}_0), \hat{f}(\hat{B}_0)) - 2\hat{Cov}(\hat{f}(\hat{B}_1), \hat{f}(\hat{B}_0))\Big]$$

Variances of and covariances between random forest predictions are estimated by means of a bias-corrected versions of the jackknife-after-bootstrap method.