Auswirkungen der Parameterwahl bei Random Forest

Bachelorarbeit

vorgelegt von Katrin Racic-Rachinsky

Betreuerin: Prof. Dr. Anne-Laure Boulesteix

Institut für Statistik
Ludwig-Maximilians-Universität München
14. März 2018

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Random Forest Algorithmus wurde 2001 von Leo Breiman entwickelt. Dabei wird ein Ensemble aus Entscheidungsbäumen mit dem CART-Algorithmus erstellt, wobei für jeden Baum eine Bootstrap-Stichprobe aus den Trainingsdaten gezogen wird. Die Prädiktionen der einzelnen Bäume werden über das ganze Ensemble aggregiert. Die Random Forest Methode erfreut sich großer Beliebtheit, weil sie viele Vorteile bietet. Sie ist für Klassifikations- und Regressionsprobleme sowie für Überlebenszeitanalysen geeignet und zeichnet sich durch Schnelligkeit und eine hohe Vorhersagegenauigkeit aus. Zudem kann mit der Methode auch die Relevanz der einzelnen Kovariablen abgeschätzt werden (Variablenwichtigkeit). Außerdem ist sie für hochdimensionale Daten geeignet und kommt auch mit komplexen Interaktionen und untereinander hochkorrelierten Kovariablen zurecht. (vgl. Strobl et al., 2008) Ein weiterer Vorteil sind die sogenannten *out-of-bag*-Daten, also die Beobachtungen aus den Trainingsdaten, die jeweils nicht in die Bootstrap-Stichprobe gezogen wurden. Sie können unter anderem als eine Art eingebauter Testdatensatz direkt zur Evaluierung der Vorhersagegenauigkeit genutzt werden.

In der Literatur wird oft erwähnt, dass Random Forests verglichen mit anderen Methoden relativ unempfindlich auf Parameterveränderungen reagieren und meist mit den Default-Einstellungen gute Ergebnisse erreicht werden. (vgl. Cutler et al., 2011, 167; Strobl et al., 2009, 32) In dieser Arbeit soll nun an realen Datenbeispielen untersucht werden, wie sich die Einstellungen der wichtigsten Parameter bei Random Forest auf die Prädiktionsgüte und auf die Variablenwichtigkeit auswirken. In Kapitel 2 wird zunächst die Theorie des CART-Algorithmus vermittelt. Anschließend geht es in Kapitel 3 hauptsächlich um die Random Forest Methode, zu der einige wichtige Details geschildert werden. Zuvor wird aber ebenfalls in Kapitel 3 noch auf andere baumbasierte Ensemblemethoden eingegangen, die als Vorläufer des Random Forests angesehen werden können. Das Kapitel schließt mit einem kurzen Einblick in den Extra-Trees-Algorithmus ab, einer eng mit dem Random Forest verwandten Methode. In Kapitel 4 werden dann die Maßzahlen erläutert, mit denen in dieser Arbeit die Prädiktionsgüte beurteilt wird. Kapitel 5 beinhaltet zuerst eine Beschreibung des `R`-Pakets `ranger` und der Beispieldatensätze. Im Anschluss werden zuerst die Ergebnisse zum Einfluss unterschiedlicher Parametereinstellungen auf die Prädiktionsgüte dargestellt, dann wird geschildert, wie sich die Wahl der Parametereinstellungen auf die Variablenwichtigkeit auswirkt. Kapitel 6 bietet schließlich noch eine kurze Zusammenfassung der Ergebnisse.

2 CART - Classification and Regression Trees

Bei dem CART-Algorithmus, der 1984 von Breiman, Friedman, Olshen und Stone entwickelt wurde, werden Entscheidungsbäume gebildet, mit deren Hilfe die Werte einer Responsevariable prädiktiert werden können. Ein Entscheidungsbaum wird aus Trainingsdaten erzeugt, für die die Werte der Responsevariable bereits bekannt sind. Die Trainingsdaten $\mathcal{L}$ mit p Einflussvariablen bestehen dementsprechend aus N Beobachtungen (x_i, y_i), für $i = 1, 2, ..., N$, mit $x_i = (x_{i1}, ..., x_{ip})$. Bäume mit kategorialem Response nennt man Klassifikationsbäume, solche mit metrischem Response Regressionsbäume. (vgl. Hastie et al., 2016, 307) Breiman et al. (1984, 228) sprechen auch von *tree structured classifiers* und *tree structured predictors*.

Die Entscheidungsbäume entstehen durch rekursives Partitionieren des Merkmalsraums: man beginnt den Baum mit einem Wurzelknoten (*root node*), der alle N Beobachtungen enthält. Dieser Knoten wird in zwei disjunkte Teilmengen (Knoten) aufgeteilt, mit denen wiederum genauso verfahren wird. Dieses Verfahren wird wiederholt, bis ein Stoppkriterium erreicht wird, zum Beispiel eine minimale Knotengröße. Ziel des Teilen ist es, Knoten zu erhalten, die in sich möglichst homogen und untereinander möglichst heterogen in Bezug auf den Response sind. Knoten, die nicht weiter geteilt werden, heißen Endknoten (*terminal nodes*) oder Blätter (*leaves*). Knoten, die keine Endknoten sind, heißen innere Knoten (*internal/non-terminal nodes*). Die Endknoten bilden eine Partition des Merkmalsraums; für jeden Endknoten wird ein konstanter Response geschätzt. (vgl. Hastie et al., 2016, 305; Fahrmeir et al., 1996, 425)

Für jede Teilung eines Knoten wird eine der Einflussvariablen x_j und ein dazu passender Splitpunkt c (bei metrischen Variablen) oder eine Teilmenge S der Klassen (bei kategorialen Variablen) ausgewählt. Die Menge der Beobachtungen, die das Splitkriterium erfüllen, d.h. $\{X|X_j \leq c,\ c \in \mathbb{R}\}$ für metrische Variablen und $\{X|X_j \in S,\ S \subset \{1, .., L\}\}$ für kategoriale Variablen mit L Kategorien, bildet den linken Knoten, wohingegen diejenigen Beobachtungen, die das Kriterium nicht erfüllen, also $\{X|X_j > c,\ c \in \mathbb{R}\}$ bzw. $\{X|X_j \notin S,\ S \subset \{1, .., L\}\}$, in den rechten Knoten wandern. (Der Einfachheit halber wird im Folgenden unabhängig von der Art der Kovariable immer von ‚Splitpunkt' die Rede sein, wenn es um die Teilung eines Knotens geht, auch wenn es für kategoriale Kovariablen keinen Split*punkt* in dem Sinne gibt.) Damit ergeben sich für metrische Variablen jeweils maximal $N - 1$ und für kategoriale Variablen jeweils $2^{L-1} - 1$ potentielle Splitmöglichkeiten. (vgl. Breiman et al., 1998, 29-30)

Abbildung 2.1 zeigt links ein Beispiel für einen Entscheidungsbaum mit zwei

metrischen Kovariablen. Der Wurzelknoten K_0 wird so aufgeteilt, dass zwei Knoten K_1 und K_2 mit $\{X_1 \leq c_1\}$ beziehungsweise $\{X_1 > c_1\}$ entstehen. K_1 wird wiederum an Hand der Variable X_1 geteilt, aber mit Splitpunkt c_2. Es entstehen die Knoten K_3 mit $\{X_1 \leq c_1\} \cap \{X_1 \leq c_2\} = \{X_1 \leq c_2\}$ (für $c_2 \leq c_1$) und R_1 mit $\{X_1 \leq c_1\} \cap \{X_1 > 5.1 c_2\} = \{c_2 < X_1 \leq c_1\}$. R_1 wird nicht weiter geteilt und ist damit ein Endknoten. K_2 wird an Hand der Variable X_2 mit Splitpunkt c_3 geteilt, wodurch sich die Endknoten R_2 mit $\{X_1 > c_1\} \cap \{X_2 \leq c_3\}$ und R_3 mit $\{X_1 > c_1\} \cap \{X_2 > c_3\}$ ergeben. K_3 wird auch noch geteilt und liefert die Endknoten R_4 und R_5, mit $\{X_1 \leq c_2\} \cap \{X_2 \leq c_4\}$ beziehungsweise $\{X_1 \leq c_2\} \cap \{X_2 > c_4\}$. Die fünf Endknoten bilden eine Partition des Merkmalsraums, das heißt, sie sind untereinander disjunkt und ihre Vereinigung ergibt den gesamten Merkmalsraum.

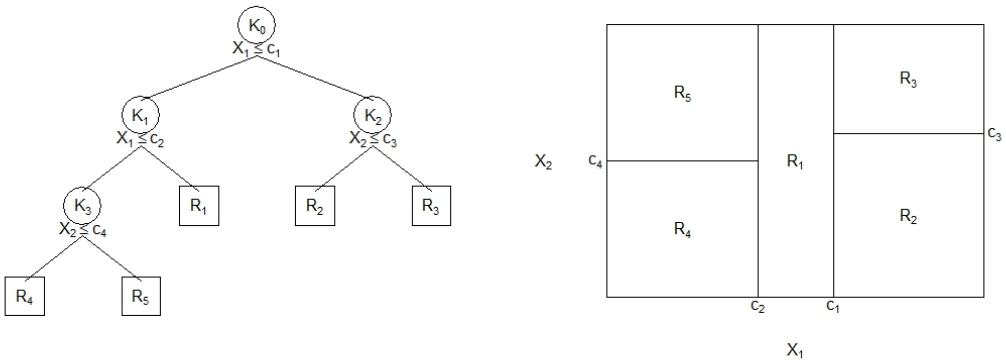


Abbildung 2.1: Partition eines zweidimensionalen Raums, links als Baum dargestellt (innere Knoten werden als Kreise dargestellt, Endknoten als Quadrate), rechts als Aufteilung in Rechtecke. (vgl. Hastie et al., 2016, 306)

Die Aufteilung in Endknoten durch rekursives binäres Partitionieren entspricht dem sukzessiven Aufteilen des Merkmalraums in Hyperrechtecke (Regionen). Im zweidimensionalen Raum lässt sich das gut veranschaulichen. Abbildung 2.1 rechts zeigt die selbe Partitionierung wie der Baum links daneben. Das gesamte Rechteck stellt, wie der Wurzelknoten des Baumes, den Merkmalsraum dar. Es wird zuerst bei c_1 geteilt und die Flächen links und rechts der senkrechten Gerade bei c_1 entsprechen den Knoten K_2 und K_3, die dann wiederum entsprechend geteilt werden, bis die fünf Regionen $R_1, ..., R_5$ entstanden sind. Diese Art der Darstellung verdeutlicht sehr gut, dass die Regionen eine Partition des Merkmalsraums darstellen. (vgl. Hastie et al., 2016, 305-306)

Zur Erstellung eines Baumes stellen sich nun drei Fragen:

1. Nach welchen Kriterien werden Splitvariable und Splitpunkt ausgewählt?

2. Wie wird entschieden, ob ein Knoten weiter geteilt oder zu einem Endknoten erklärt wird?

3. Wie werden den Endknoten Klassen oder Werte zugewiesen?

(vgl. Breiman et al., 1998, 22)

Am leichtesten lässt sich die dritte Frage beantworten. Für jeden Endknoten/jede Region R_m kann aus den Daten ein konstanter Responsewert geschätzt werden, der sich folgendermaßen darstellen lässt:

$$f(x) = \sum_{m=1}^{M} \beta_m \mathbb{I}(x \in R_m), \tag{2.1}$$

wobei M die Anzahl der Endknoten und $\mathbb{I}$ die Indikatorfunktion bezeichnet. Für einen kategoriellen Response mit L Klassen wird jedem Endknoten einfach die Klasse l zugewiesen, die am häufigsten in ihm vertreten ist, das heißt, es gilt

$$\hat{\beta}_m = \arg\max_{l} \sum_{x_i \in R_m} \mathbb{I}(y_i = l). \tag{2.2}$$

Nimmt man sich für eine metrischen Response zum Ziel, die Quadratsumme $\sum(y_i - f(x_i))^2$ zu minimieren, so ist offensichtlich der Mittelwert der Beobachtungen im Endknoten die beste Wahl, es gilt also

$$\hat{\beta}_m = ave(y_i | x_i \in R_m) = \frac{1}{N_m} \sum_{x_i \in R_m} y_i, \tag{2.3}$$

wobei N_m die Anzahl der Beobachtungen in Endknoten R_m bezeichnet. (vgl. Hastie et al., 2016, 307-9)

Das Ziel bei der Teilung der Knoten ist immer eine Verringerung der ‚Unreinheit'(*impurity*). Ein Knoten ist umso unreiner, je höher die Variabilität der Responsewerte innerhalb des Knotens ist. Für Klassifikationsbäume bedeutet das, die Unreinheit ist maximal, wenn alle Klassen gleich stark in einem Knoten vertreten sind und minimal, wenn alle Beobachtungen die gleiche Klasse haben.

Für Klassifikationsbäume kann man die Anteile

$$\hat{p}_{kl} = \frac{1}{N_k} \sum_{x_i \in R_k} \mathbb{I}(y_i = l) \tag{2.4}$$

der Responseklassen $1, ..., L$ in einem Knoten R_k bestimmen, wobei N_k die Anzahl der Beobachtungen im Knoten R_k bezeichnet. Um die Unreinheit eines Knoten auszudrücken, muss ein Unreinheitsmaß $i(k)$ als nichtnegative Funktion ϕ der $\hat{p}_{k1}, ..., \hat{p}_{kL}$

definiert werden, so dass ϕ für $\hat{p}_{k1} = p_{k2} = ... = \hat{p}_{kL}$ maximal wird und so dass gilt $\phi = 0$, wenn genau ein $\hat{p}_{kl}$ den Wert 1 annimmt. (vgl. Breiman et al., 1998, 23-5)

Mögliche Unreinheitsmaße sind unter anderem die Fehlklassifikationsrate, der Gini-Index oder die Entropie:

$$\text{Fehlklassifikationsrate: } \frac{1}{N_k} \sum_{x_i \in R_k} \mathbb{I}(y_i \neq l) = 1 - \hat{p}_{kl} \tag{2.5}$$

$$\text{Gini-Index: } \sum_{l \neq l'} \hat{p}_{kl}\ \hat{p}_{kl'} = \sum_{l=1}^{L} \hat{p}_{kl}(1 - \hat{p}_{kl}) = 1 - \sum_{l=1}^{L} \hat{p}_{kl}^2 \tag{2.6}$$

$$\text{Entropie: } -\sum_{l=1}^{L} \hat{p}_{kl} \log\ \hat{p}_{kl} \tag{2.7}$$

Allerdings reagieren der Gini-Index und die Entropie besser als die Fehlklassifikationsrate auf Änderungen in den Knotenwahrscheinlichkeiten. Hat man zum Beispiel für $L = 2$ einen Knoten mit 400 Beobachtungen in jeder Klasse (bezeichnet als (400,400)) und einen Split $(j, c)_1$, der Knoten (300,100) und (100,300) erzeugt, sowie einen Split $(j, c)_2$, der Knoten (200,400) und (200,0), so sollte $(j, c)_2$ vorgezogen werden, da er einen ganz reinen Knoten erzeugt. Gini-Index und Entropie nehmen auch für den zweiten, bevorzugten Split einen niedrigeren Wert an, wohingegen die Fehlklassifikationsrate in beiden Fällen 0.25 ergibt. Gini-Index und Entropie sind damit im Allgemeinen der Fehlklassifikationsrate als Unreinheitsmaß vorzuziehen. (vgl. Hastie et al., 2016, 309-10)

Als Kriterium dafür, wie gut ein Split (j, c) mit Variable j und Splitpunkt c die Unreinheit eines Knoten verringert, kann die Abnahme der Unreinheit für Klassifikationsbäume folgendermaßen definiert werden:

$$\triangle i[(j, c), k] = i(k) - [p_{yes} i(k_{yes}) + p_{no} i(k_{no})], \tag{2.8}$$

wobei p_{yes} und p_{no} jeweils der relative Anteil der Beobachtungen aus Knoten R_k ist, die durch den Split (j, c) in den linken Knoten $R_{k_{yes}}$ und in den rechten Knoten $R_{k_{no}}$ wandern. Indem man von allen möglichen Splits denjenigen mit minimaler Unreinheit $p_{yes} i(k_{yes}) + p_{no} i(k_{no})$ aussucht, maximiert man die Abnahme der Unreinheit. (vgl. Tutz et al 2012, Breiman)

Bei Regressionsbäumen verwendet man als Unreinheitsmaß $i(k)$ eines Knoten R_k die Summe der quadratischen Abstände $\sum_{x_i \in R_k} (y_i - \beta_k)^2$; die Abnahme der Unreinheit kann definiert werden als

$$\triangle i[(j,c),k] = i(k) - [i(k_{yes}) + i(k_{no})]. \tag{2.9}$$

Anders als bei Klassifikationsbäumen hat $i(k)$ bei Regressionsbäumen kein Maximum, wird aber auch minimal mit 0 für perfekte Reinheit, d.h. wenn der Knoten gar keine Variabilität aufweist. Bei der Abnahme der Unreinheit werden $i(k_{yes})$ und $i(k_{no})$ nicht gewichtet. Setzt man für β_k wieder den Mittelwert $\bar{y}(k) = \frac{1}{N_k} \sum_{x_i \in R_k} y_i$ ein, so muss für die Teilung eines Knotens also der Split gefunden werden, der

$$\sum_{x_i \in R_{k_{yes}}} (y_i - \bar{y}(k_{yes}))^2 + \sum_{x_i \in R_{k_{no}}} (y_i - \bar{y}(k_{no}))^2$$

minimiert. (vgl. Hastie et al., 2016, 307)

Nachdem nun erläutert wurde, nach welchen Kriterien Variable und Splitpunkt ausgewählt werden, bleibt noch zu klären, wann ein Knoten nicht weiter geteilt werden soll. Lässt man den Baum zu groß wachsen, so dass die Endknoten nur noch sehr wenige Beobachtungen enthalten, droht Overfitting. Das heißt, der Baum ist optimal an die Trainingsdaten angepasst, prädiktiert aber nicht so gut für neue Daten. Zu kleine Bäume hingegen erfassen eventuell die Struktur der Daten nicht gut genug. Um die Baumgröße angemessen zu beschränken, kann man zum Beispiel einen Schwellenwert festlegen und Knoten nur teilen, wenn die Abnahme der Unreinheit dadurch diesen Schwellenwert übersteigt. Allerdings besteht die Möglichkeit, dass auch ein weniger guter Split später zu einem sehr guten Split führt, weswegen diese Methode nicht unbedingt empfehlenswert ist.

Ein anderer Ansatz ist die Kosten-Komplexitäts-Beschneidung (*cost-complexity pruning*). Dabei lässt man den Baum sehr groß wachsen, bis eine minimale Knotengröße (von zum Beispiel fünf Beobachtungen pro Knoten) erreicht ist, und schneidet ihn dann zurück. Das funktioniert folgendermaßen: Sei T_0 der unbeschnittene Baum und $T \subset T_0$ ein beliebiger Baum, der durch das Zurückschneiden von T_0 erzeugt wird. Zurückschneiden bedeutet, dass innere Knoten aufgelöst werden. Das Kosten-Komplexitäts-Kriterium wird definiert als

$$C_\alpha(T) = \sum_{m=1}^{M} i(m) + \alpha M, \tag{2.10}$$

wobei M die Anzahl der Endknoten von Baum T ist und $\alpha \geq 0$ ein Tuning-Parameter, der den Trade-off zwischen Baumgröße und Güte der Anpassung an die Daten bestimmt. Je größer α gewählt ist, desto kleiner ist der Baum T, für $\alpha = 0$ hingegen erhält man den vollen Baum T_0.

Für jedes α gibt es einen eindeutig bestimmbaren Unterbaum T_α, der $C_\alpha(T)$ minimiert. Um T_α zu finden, wird *weakest link pruning* angewendet. Dabei wird nacheinander jeweils der innere Knoten aufgelöst, der die geringste Erhöhung pro Knoten von $\sum_m i(m)$ aufweist, $m = 1, ..., M$, bis nur noch der Wurzelknoten als ganzer Baum übrigbleibt. Dadurch entsteht eine endliche Sequenz von Unterbäumen, die T_α enthält. α kann mit Kreuzvalidierung geschätzt werden. Der optimal beschnittene Baum ist $T_{\hat{\alpha}}$, wobei $\hat{\alpha}$ so gewählt wird, dass die kreuzvalidierte Quadratsumme minimal wird. (vgl. Hastie et al., 2016, 307-8)

3 Baumbasierte Ensemblemethoden

Entscheidungsbäume brauchen als nichtparametrische Methode keine Verteilungsannahmen, haben einen geringen Bias und sind dank ihrer Baumstruktur leicht interpretierbar. Ein großer Nachteil ist allerdings ihre große Instabilität. Eine kleine Änderung in den Daten kann erhebliche Auswirkungen auf den ganzen Baum haben, was vor allem durch die hierarchische Struktur der Bäume bedingt ist: kommt es wegen kleinen Änderungen in den Daten (wie zum Beispiel Weglassen oder Hinzufügen weniger Beobachtungen) dazu, dass die erste splitting-Variable oder auch nur der erste Splitpunkt anders gewählt werden, so wirkt sich das auf alle weiteren Knoten und damit auf die gesamte Baumstruktur aus. Wegen dieser Instabilität weisen einzelne Bäume eine sehr hohe Variabilität in ihren Prädiktionen auf.

Ensemblemethoden wie Bagging, die Random Subspace Methode oder Random Forest bieten eine Lösung für dieses Problem der Instabilität: anstelle eines einzelnen Baumes wird die Mittelung der Prädiktionen eines Ensembles von Bäumen zur Vorhersage genutzt, wodurch eine Verringerung der Varianz erreicht wird. (vgl. Strobl et al., 2009, 16)

3.1 Tree bagging

Die Methode des *baggings* (kurz für ‚***b**ootstrap **agg**regat**ing**‘) wurde 1996 von Leo Breiman eingeführt. Das Grundprinzip ist einfach: Bagging passt mit Hilfe von Bootstrap-Stichproben viele Modelle an und mittelt ihre Prädiktionen zu einer Gesamtprädiktion. Das Prinzip des baggings lässt sich auf die unterschiedlichsten Modelle anwenden. Im Folgenden geht es allerdings nur um bagging von Entscheidungsbäumen. Wie in Kapitel 2 wird wieder von Trainingsdaten $\mathcal{L} = \{(x_1, y_1),(x_2, y_2), ..., (x_N, y_N)\}$ ausgegangen. Aus den Trainingsdaten wird eine Bootstrap-Stichprobe gezogen, das heißt, aus ihnen wird N-mal mit Zurücklegen gezogen. Dieser Vorgang

wird B-mal wiederholt, so dass B Bootstrap-Stichproben $\mathcal{L}^{*b}$, $b = 1, 2, ..., B$, jeweils vom Umfang N vorliegen. Für jede der Bootstrap-Stichproben wird ein Entscheidungsbaum T_b ohne Zurückschneiden erzeugt mit Vorhersagefunktion $\hat{f}^{*b}(x)$. (vgl. Breiman 1996, 123; vgl. Hastie et al., 2016, 282)

Für Regressionsbäume entspricht die Bagging-Vorhersagefunktion $\hat{f}^{bag}(x)$ dem Mittelwert der Vorhersagefunktionen $\hat{f}^{*b}(x)$ der einzelnen Bäume T_b:

$$\hat{f}^{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{3.1}$$

Bei Klassifikationsbäumen wird die Klasse l, $l = 1, ..., L$, prädiktiert, die von der Mehrheit der einzelnen Bäume T_b vorhergesagt wird (*majority vote*):

$$\hat{f}^{bag}(x) = \arg \max_{l} \sum_{b=1}^{B} \mathbb{I}(\hat{f}^{*b}(x) = l). \tag{3.2}$$

(vgl. Hastie et al., 2016, 282-3; Breiman 1996, 123)

Möchte man bei einem einzelnen Klassifikationsbaum für einen Beobachtungsvektor x_i keine Klasse prädiktieren, sondern Wahrscheinlichkeiten $\hat{\pi}_i$ dafür schätzen, welche Klasse vorliegt, so kann man dies mit den Anteilen $\hat{p}_{ml}$ der Responseklassen $1, ..., L$ in dem Endknoten R_m ($m = 1, ..., M$, wobei M der Anzahl der Endknoten entspricht), in dem x_i liegt, abschätzen (vgl. Formel (2.4)). Die bagging-Schätzung ergibt sich über den Mittelwert der Anteile $\hat{p}^{*b}_{ml}$ der einzelnen Bäume T_b (vgl. Breiman 1996, 135):

$$\hat{\pi}_i(l) = \widehat{P(l|x_i)} = \hat{p}^{bag}_{ml} = \frac{1}{B} \sum_{b=1}^{B} \hat{p}^{*b}_{ml}. \tag{3.3}$$

Nicht geeignet als Schätzung für $P(l|x_i)$ ist der Anteil $a_l(x_i)$ der Bäume, die für x_i die Klasse l prädiktieren. Stellt man sich beispielsweise ein Klassifikationsproblem vor, bei dem die wahre Wahrscheinlichkeit für Klasse 1 für x_i bei 0.75 liegt, und alle Bäume des Ensembles ordnen x_i die Klasse 1 zu, so ist $a_1 = 1$ kein guter Schätzer für $P(1|x_i)$. Lässt man allerdings die Bäume soweit wachsen, dass alle Endknoten nur noch eine Beobachtung enthalten oder nur Beobachtungen der selben Klasse, so gilt $a_l(x_i) = \hat{p}^{bag}_{ml}$. (vgl. Hastie et al., 2016, 283)

Bagging liefert für instabile Methoden sehr gute Ergebnisse, für sehr stabile Methoden bringt es hingegen kaum Verbesserung oder kann sogar zu einer leichten Verschlechterung der Vorhersagen führen. Für Klassifikations- und Regressionsbäume, die eine sehr hohe Variabilität haben, aber im Allgemeinen unverzerrt sind (das

heißt, im Mittel prädiktieren sie richtig), eignet sich die Methode sehr gut und kann, wie Breiman gezeigt hat, gegenüber einzelnen Bäumen zu erheblichen Verringerungen in den Fehlerraten der Prädiktionen führen. (vgl. Breiman 1996, 124; Strobl et al., 2009, 15-6)

3.2 Die Random Subspace Methode

Ein anderer Ansatz, um mehrere Bäume zu aggregieren und so die Varianz der Prädiktionen zu senken, wurde 1998 von Tin Kam Ho entwickelt: Die Random Subspace Methode. Der Grundgedanke ist hierbei, dass mehrere Bäume aus den gleichen Trainingsdaten $\mathcal{L}$ erzeugt werden, aber jeweils nur eine zufällig ausgewählte Teilmenge der Variablen als mögliche Splitvariablen zugelassen werden. Ho verwendet für das Ensemble von Bäumen den Ausdruck *decision forest*. Anstelle des gesamten Merkmalsraums wird für jeden Baum nur ein zufällig ausgewählter Unterraum (*random subspace*) des Merkmalsraums verwendet.

Für einen Merkmalsraum mit p Dimensionen ergeben sich $2^p - 1$ mögliche Unterräume, auf deren Basis ein Entscheidungsbaum erzeugt werden kann; falls man nicht nur für jeden Baum, sondern sogar für jeden Split innerhalb des Baumes eine neue zufällige Variablenauswahl trifft, sind es noch erheblich mehr. Etwas weniger mögliche Unterräume gibt es, wenn nicht nur die Variablenauswahl innerhalb eines Baumes gleich bleibt, sondern auch eine feste Anzahl m_{try} möglicher Variablen für alle Bäume gewählt wird. In diesem Fall sind es $\binom{p}{m_{try}}$ mögliche Unterräume. Die Prädiktionen ergeben sich wie beim Bagging durch das Mittel (bei metrischem Response) oder durch einen *majority vote* (bei kategorialem Response) der Prädiktionen der einzelnen Bäume. (vgl. Ho 1998, 833-4)

Die Random Subspace Methode ist sehr gut für hochdimensionale Daten geeignet, da es durch eine hohe Anzahl an Variablen mehr als ausreichend mögliche Unterräume gibt, um eine Vielzahl an unterschiedlichen Bäumen aus denselben Trainingsdaten zu generieren. Für Daten mit sehr kleiner Variablenanzahl hingegen ist der Vorteil der Methode gegenüber einem einzelnen Baum nur gering. Bei den Analysen, die Ho durchgeführt hat, schnitt die Random Subspace Methode nicht nur im Vergleich zu einzelnen Klassifikationsbäumen, sondern auch im Vergleich zum Bagging besser ab. Als optimale Anzahl der zufällig ausgewählten Variablen empfiehlt Ho, $\frac{p}{2}$ zu nehmen. (vgl. Ho 1998, 833-840)

3.3 Random Forest

Random Forest kombiniert das Prinzip des Baggings mit der Random Subspace Methode. Die Prädiktion wird gemittelt über ein Ensemble von Bäumen, die aus Bootstrap-Stichproben erzeugt werden, wobei für jeden Split in jedem Baum nur eine zufällig ausgewählte Menge an Variablen verwendet wird. Der hauptsächliche Vorteil gegenüber des Baggings liegt darin, dass die einzelnen Bäume des Ensembles sich stärker voneinander unterscheiden und damit viel weniger stark miteinander korrelieren. Der ursprüngliche Random Forest Algorithmus wurde 2001 von Leo Breiman vorgestellt, und die Methode ist seitdem auf vielfältige Weise weiterentwickelt worden.

Durch die zufällige Variablenselektion werden für Splits öfter Variablen mit weniger Einfluss ausgewählt. Doch obwohl solche Splits erst einmal die Knotenunreinheit weniger verringern, als es Splits mit einer stärkeren Variable getan hätten, können sie insgesamt zu einer Verbesserung der Prädiktionsgüte beitragen. Der Grund dafür ist, dass es sich bei dem Algorithmus zur Erstellung eines einfachen Entscheidungsbaums um einen sogenannten Greedy-Algorithmus handelt, der immer nur den lokal besten Split auswählt, ohne die Auswirkungen auf nachfolgende Splits zu berücksichtigen. Alle Splits so zu wählen, dass sie zum global besten Baum führen, ist rechnerisch nicht möglich, deswegen können lokal weniger gute Splits mitunter global bessere Bäumen bewirken. (vgl. Strobl et al., 2009, 17-18)

3.3.1 Algorithmus und Parameter

Der Random Forest Algorithmus lässt sich folgendermaßen zusammenfassen (vgl. Algorithm 15.1 in Hastie et al., 2016, 388 und Algorithm 2 in Cutler et al., 2012, 8):

1. Für $b = 1, ..., B$ führe jeweils a) und b) durch:

a) Ziehe aus den Lerndaten eine Bootstrap-Stichprobe $\mathcal{L}^{*b}$ vom Umfang N

b) Lasse mit $\mathcal{L}^{*b}$ einen Entscheidungsbaum T_b ohne Zurückschneiden wachsen, indem mit allen Beobachtungen im Wurzelknoten begonnen wird und dann rekursiv für jeden Knoten die folgenden Schritte durchgeführt werden, bis eine minimale Knotengröße n_{min} erreicht ist.

i. Wähle zufällig m_{try} Variablen der p Variablen aus

ii. Wähle die beste Variablen/Splitpunkt-Kombination unter den m_{try} Variablen aus Schritt i. aus

iii. Teile den Knoten an Hand der Variablen/Splitpunkt-Kombination aus Schritt ii. in zwei Knoten auf

2. Man erhält ein Ensemble von Bäumen $\{T_b\}_1^B$

Für eine neue Beobachtung $x_0 = (x_{01}, x_{02}, ..., x_{0p})$ kann der Response folgendermaßen prädiktiert werden:

$$\text{Regression: } \hat{f}^{rf}(x_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x_0) \tag{3.4}$$

$$\text{Klassifikation: } \hat{f}^{rf}(x_0) = \arg\max_{l} \sum_{b=1}^{B} \mathbb{I}(\hat{f}^{*b}(x_0) = l) \tag{3.5}$$

wobei $\hat{f}^{*b}(x)$ jeweils die Vorhersagefunktion von Baum T_b ist. Die Prädiktionen werden also genauso wie beim bagging über den Mittelwert (für Regression) bzw. *majority vote* (für Klassifikation) gebildet (vgl. Formeln (3.1) und (3.2)). Genau so wie beim Bagging kann man bei Klassifikation für eine Beobachtung x_i nicht nur die Responseklasse selbst schätzen, sondern auch Wahrscheinlichkeiten $\hat{\pi}_i(l) = \hat{p}_{ml}^{rf}$ dafür, dass für x_i Klasse l vorliegt. (vgl. Formel (3.3))

Die einzelnen Bäume T_b sollten nicht zurückgeschnitten werden. Abhängig von den Daten kann es aber zu Overfitting kommen, wenn man die Bäume zu weit wachsen lässt, so dass eine geringere Tiefe der Bäume die Prädiktionsgenauigkeit verbessern kann. Die Tiefe der Bäume lässt sich über die minimale Knotengröße steuern, ab der nicht mehr weiter geteilt wird, oder auch über eine maximale Endknotenanzahl.

Im Allgemeinen gilt für die Anzahl B der Bäume, dass die Vorhersage umso zuverlässiger wird, umso mehr Bäume verwendet werden. Um die Stabilität der OOB-Prädiktionen zu gewährleisten, sollte B daher nicht zu klein gewählt werden. Anders als bei vielen anderen Ensemblemethoden ist auch mit steigender Baumanzahl kein Overfitting zu erwarten. (vgl. Cutler et al., 2012)

Für die Wahl des Parameters m_{try} wird im Allgemeinen $m_{try} = \lfloor\sqrt{p}\rfloor$ für Klassifikation und $m_{try} = \lfloor p/3\rfloor$ für Regression empfohlen. Bei manchen Daten erzielen andere Werte für m_{try} eine bessere Prädiktionsgüte, so dass m_{try} als Tuning-Parameter genutzt werden kann, das heißt, als Parameter, mit dessen Anpassung die Genauigkeit der Methode eventuell verbessert werden kann. (vgl. Hastie et al., 2016, 592)

Anstelle von Bootstrap-Stichproben können auch Stichproben mit kleinerem Umfang ohne Zurücklegen gezogen werden (*subsampling*). Die Standardeinstellung für

den Stichprobenumfang bei *subsampling* ist $0.632 * N$, was in etwa dem Anteil der Beobachtungen entspricht, die in eine Bootstrap-Stichprobe gezogen werden. Die Verwendung von *subsamples* anstatt von Bootstrap-Stichproben kann dazu beitragen, eine Verzerrung in der Variablenselektion zu verringern. (vgl. Strobl et al., 2009, 30)

3.3.2 Verringerung der Korrelationen

Beim Bagging sind die Vorhersagefunktionen $\hat{f}^{*b}(x)$ der einzelnen Bäume eines Ensembles identisch verteilt (*identically distributed*, i.d.), aber da sie alle aus sehr ähnlichen Daten erzeugt werden, sind sie normalerweise nicht unabhängig identisch verteilt (*independently identically distributed, i.i.d.*). Das heißt, die einzelnen Vorhersagefunktionen korrelieren miteinander, was Auswirkungen auf die Varianz der Vorhersagefunktion $\hat{f}^{bag}(x)$ hat: Bei B i.i.d. Zufallsvariablen, die jeweils die Varianz σ^2 haben, beträgt die Varianz ihres Mittelwerts $\frac{1}{B}\sigma^2$, wird also mit steigendem B immer kleiner. Sind die Zufallsvariablen hingegen nur i.d. mit paarweiser Korrelation ρ, so ist die Varianz des Mittelwerts

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2. \tag{3.6}$$

Der zweite Term wird mit steigendem B vernachlässigbar, doch der erste Term ist unabhängig von B und umso größer, je größer die Korrelation ρ ist. Die Varianz der bagging-Prädiktion ist also abhängig von den Korrelationen zwischen den einzelnen Bäumen des Ensembles.

Durch die Einschränkung der Kovariablenauswahl bei Random Forest unterscheiden sich die einzelnen Bäume des Ensembles stärker voneinander. Die Korrelationen zwischen den Bäumen werden verringert, wodurch sich auch die Varianz des Mittelwerts der Vorhersagefunktionen verringert und die Prädiktionen stabiler werden.

3.3.3 Out of bag Daten

Bei jeder Bootstrap-Stichprobe gibt es einige Beobachtungen aus dem ursprünglichen Datensatz, die nicht in die Bootstrap-Stichprobe gezogen wurden. Diese Daten werden bei Random Forest als ‚out of bag' (OOB) Daten bezeichnet. Sie werden innerhalb des Random Forest Algorithmus als Testdaten genutzt, um die Prädiktionsgüte und die Variablenwichtigkeit abzuschätzen.

Für jede Beobachtung $z_i = (x_i, y_i)$ wird der Response mit dem Mittelwert bzw. mit *majority vote* der Bäume prädiktiert, die aus Bootstrap-Stichproben erstellt

wurden, in denen z_i **nicht** enthalten ist. Bezeichnet man mit $B_i = \{b : (z_i) \notin \mathcal{L}^* b\}$ die Menge der Indizes b, für die z_i nicht in der entsprechenden Bootstrap-Stichprobe $\mathcal{L}^{*b}$ ist, so ergeben die OOB-Prädiktionen bei metrischem Response

$$\hat{f}_{oob}(x_i) = \frac{1}{|B_i|} \sum_{b \in B_i} \hat{f}^{*b}(x_i), \tag{3.7}$$

wobei $|B_i|$ die Kardinalität von B_i bezeichnet. Die OOB-Prädiktionen für kategorialen Response werden wieder durch *majority vote* ermittelt:

$$\hat{f}_{oob}(x_i) = \arg\max_{l} \sum_{b \in B_i} \mathbb{I}(\hat{f}^{*b}(x_i) = l). \tag{3.8}$$

Für einen metrischen Response wird die Güte des Modells typischerweise mit dem mittleren quadratischen Fehler (MSE) der *out-of-bag*-Prädiktionen beurteilt:

$$MSE_{oob} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}_{oob}(x_i))^2. \tag{3.9}$$

Um die Prädiktionsgüte für einen kategorialen Response zu beurteilen, kann man die Fehlklassifikationsrate für die OOB-Daten bilden:

$$E_{oob} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(y_i \neq \hat{f}_{oob}(x_i)). \tag{3.10}$$

Die Fehlerrate wird also über die gesamten OOB-Daten berechnet, und nicht wie manchmal fälschlicherweise angenommen über den Mittelwert der OOB-Fehlklassifikationsraten der einzelnen Bäume. Dadurch lässt sich auch für jede Klasse einzeln die Fehlerrate berechnen, und mit der Kreuztabelle aus y_i und $\hat{f}_{oob}(x_i)$ erhält man eine OOB-Konfusionsmatrix. (vgl. Cutler et al., 2012, 9)

3.3.4 Variablenwichtigkeit

Neben Prädiktionen für neue Beobachtungen kann man mit Hilfe des Random Forest Algorithmus auch Aufschluss darüber erlangen, wie stark der Einfluss der einzelnen Kovariablen auf den Response ist, bzw. welche Kovariablen überhaupt für den Response relevant sind. Dazu wird für jede Kovariable die Variablenwichtigkeit (*variable importance*) gemessen. Eine Möglichkeit, die Variablenwichtigkeit zu messen, besteht darin, für jede Variable zu erfassen, um wie viel sie die Unreinheit in jedem Baum des Ensembles insgesamt verringert hat. Diese Abnahme der Unreinheit wird dann für jede Variable jeweils über alle Bäume gemittelt und ergibt

eine Maßzahl für die Relevanz der Variable. Für Klassifikation gibt es zum Beispiel die *Gini importance*, wenn als Unreinheitsmaß der Gini-Index (2.6) verwendet wird. Die *Gini importance* gibt die durchschnittliche Unreinheitsabnahme nach dem Gini-Kriterium an, die eine Variable an allen ihren Positionen im Random Forest erzielt. (vgl. Strobl et al., 2009, 22)

Mit Hilfe der OOB-Daten kann die Variablenwichtigkeit auch auf eine andere Art gemessen werden, mit der *permutation importance* . Für jeden Baum T_b wird für die jeweiligen OOB-Daten die Fehlklassifikationsrate bzw. der MSE erfasst. Dann werden die Werte der j-ten Variable in den OOB-Daten zufällig permutiert und der Fehler wird für diese modifizierten OOB-Daten erneut berechnet. Die Zunahme des Fehlers im Vergleich zu der nicht permutierten Variable wird über alle Bäume gemittelt und als Variablenwichtigkeitsmaß für die Variable j genutzt. Je niedriger die Zunahme des Fehlers, desto geringer ist die Relevanz der Variable. Ändert sich der Fehler gar nicht durch die Permutation der Werte, so hat die Variable offensichtlich keinen Einfluss auf den Response und ihre Variablenwichtigkeit beträgt Null. Auch wenn eine Variable in einem Baum gar nicht vorkommt, ist ihre Variablenwichtigkeit für diesen Baum per Definition Null. Nimmt der Fehler durch die Permutation nicht ab, sondern sogar zu, so ergibt sich ein negativer Wert für die Variablengewichtigkeit. Eine Abnahme des Fehlers ist so zu interpretieren, dass die Variable keinen Einfluss hat und die permutierten Fehler zufällig zu einer besseren Prädiktion führen. (vgl. Strobl et al., 2009, 20)

Formal lässt sich das Ganze so darstellen: Seien $\overline{\mathcal{L}}^{*b}$ die OOB-Daten für die Bootstrap-Stichprobe $\mathcal{L}^{*b}$ für einen Baum T_b und sei $|\overline{\mathcal{L}}^{*b}|$ die Kardinalität von $\overline{\mathcal{L}}^{*b}$. Dann ist die Wichtigkeit einer Variable j in Baum T_b

$$VI_b(j) = \frac{\sum\limits_{i \in \overline{\mathcal{L}}^{*b}} \mathbb{I}(y_i \neq \hat{f}^{*b}(x_i)) - \sum\limits_{i \in \overline{\mathcal{L}}^{*b}} \mathbb{I}(y_i \neq \hat{f}^{*b}(x_i^{perm}))}{|\overline{\mathcal{L}}^{*b}|}, \tag{3.11}$$

wobei $\hat{f}^{*b}(x_i^{perm})$ die Prädiktion für die i-te Beobachtung nach der Permutation der Werte der Variable j bezeichnet. Durch Mittelung über alle Bäume wird die Gesamtvariablenwichtigkeit der Variable j

$$VI(j) = \frac{1}{B} \sum_{b=1}^{B} VI_b(j) \tag{3.12}$$

berechnet. Ein großer Vorteil der *permutation importance* bei Random Forest ist, dass nicht nur Einfluss der Variable alleine, sondern auch ihr Einfluss in Interaktionen mit anderen Kovariablen berücksichtigt wird. (vgl. Strobl et al., 2009, 22)

3.3.5 Selektionsbias und alternative Splitkriterien

Wenn man, wie bisher beschrieben, zur Teilung eines Knotens aus allen möglichen Splits den besten in Bezug auf die Abnahme der Unreinheit auswählt, kann es zu einem Bias in der Variablenselektion kommen. Kovariablen, die viele mögliche Splitpunkte anbieten, haben eine höhere Wahrscheinlichkeit, als Splitvariablen ausgewählt zu werden als solche mit weniger potentiellen Splitpunkten. Das heißt, es gibt einen Selektionsbias zugunsten von metrischen Variablen und kategorialen Variablen mit vielen Klassen gegenüber kategorialen Variablen mit wenigen Klassen. Sie werden unabhängig von ihrer tatsächlichen Relevanz öfter ausgewählt. Auch Variablen mit vielen fehlenden Werten werden bevorzugt ausgewählt. (vgl. Strobl et al., 2007) Dieser Bias in der Variablenselektion hat einerseits Auswirkungen auf die Prädiktionen, andererseits auch auf die Messung der Variablenwichtigkeit. Besonders stark zeigt sich der Bias bei der *Gini importance*, doch auch die *permutation importance* kann in geringerem Ausmaß davon betroffen werden. (vgl. Strobl et al., 2009, 30)

Conditional inference forests. Einen Ansatz, um den Selektionsbias zu umgehen, bieten *conditional inference forests* (CIF), deren Bäume nicht nach dem CART-Algorithmus gebildet werden, sondern sogenannte *conditional inference trees* sind. Das Konzept wurde von Hothorn et al. (2006) entwickelt. Dabei wird die Selektion der Splitvariable von der Wahl des Splitpunkts getrennt.

Jeder Knoten wird durch einen Gewichtsvektor w repräsentiert, der für die Beobachtungen, die im Knoten enthalten sind, jeweils eine 1 und für die Beobachtungen, die nicht im Knoten enthalten sind jeweils eine 0 enthält. Die Teilung eines Knotens erfolgt in zwei Schritten. Im ersten Schritt wird für den Gewichtsvektor w die globale Nullhypothese auf Unabhängigkeit zwischen einer der m_{try} Kovariablen und dem Response getestet. Dafür wird ein Permutationstest mit linearen Teststatistiken verwendet. Falls die Nullhypothese zu einem vorher definierten Signifikanzniveau α nicht verworfen werden kann, wird der Knoten nicht weiter geteilt und ist somit ein Endknoten des Baumes. Wird die Nullhypothese abgelehnt, so wird die Kovariable mit dem kleinsten p-Wert ausgewählt. (vgl. Hothorn et al., 2006)

Im zweiten Schritt wird für die ausgewählte Kovariable der optimale Splitpunkt gesucht, indem für alle möglichen binären Partitionen ein Zwei-Stichproben-Test mit linearen Teststatistiken durchgeführt wird. Ausgewählt wird der Splitpunkt mit der größten Teststatistik. Da es für binäre Kovariablen immer nur einen möglichen Split gibt, entfällt hier natürlich der Test aus Schritt 2. An Hand der ausgewählten

Variable und des ausgewählten Splitpunkts wird der Knoten geteilt. Diese beiden Schritte werden rekursiv für alle Knoten wiederholt. (vgl. Hothorn et al., 2006; Wright et al., 2017, 1273)

Maximally selected rank statistics. Auch bei Random Forests, die *maximally selected rank statistics* verwenden (MSR-RF), wird die Suche nach der optimalen Variablen/Splitpunkt-Kombination in zwei Schritte aufgeteilt. Im ersten Schritt wird für jede Kovariable der beste Splitpunkt gefunden. Für jeden möglichen Splitpunkt c einer Variable werden lineare Rangstatistiken berechnet, und der Split mit der größten standardisierten Teststatistik wird als bester Split c_{opt} für diese Kovariable ausgewählt. (vgl. Wright et al., 2016, 1273-1274) Die Menge der möglichen Splitpunkte kann eingeschränkt werden, indem man nur Splitpunkte c für die Berechnung der Rangstatistiken zulässt, für die gilt $\varepsilon_1 \leq c \leq \varepsilon_2$, wobei $\varepsilon_1,\ \varepsilon_2$ Quantile der Verteilung der entsprechenden Kovariable sind und $\varepsilon_2 = 1 - \varepsilon_1$ gilt. Damit kann sichergestellt werden, dass die beiden durch die Teilung neu entstehenden Knoten jeweils eine ausreichend große Anzahl an Beobachtungen enthalten. (vgl. Wright et al., 2016, 1274)

Für binäre Kovariablen entfällt Schritt eins komplett, da es nur einen möglichen Split gibt, der damit automatisch für diese Variable ausgewählt wird. Im zweiten Schritt wird für jede Kovariable und ihren ausgewählten Splitpunkt c_{opt} getestet, ob die Aufteilung nach diesem Splitpunkt Einfluss auf die Verteilung des Responses hat. Die Nullhypothese, dass der Splitpunkt c_{opt} keinen Einfluss hat, ist damit $H_0 : P(Y \leq y | X \leq c_{opt}) = P(Y \leq y | X > c_{opt})$. Die Kovariable, die für diesen Test den kleinsten p-Wert hat, wird mit ihrem ausgewählten Splitpunkt c_{opt} für die Teilung ausgewählt. Da die genaue Verteilung der Teststatistik unter der Nullhypothese nicht bekannt ist, müssen die p-Werte approximiert werden. Die p-Werte werden mit der Benjamini-Hochberg-Prozedur adjustiert für das multiple Testen von m_{try} Kovariablen an einem Knoten. Ist keiner der adjustierten p-Werte kleiner als ein zuvor festgelegtes Signifikanzniveau α, so wird keine Teilung des Knotens durchgeführt. Andernfalls wird der Knoten mit der ausgewählten Variablen/Splitpunkt-Kombination geteilt. Schritt eins und Schritt zwei werden rekursiv für jeden Knoten wiederholt. (vgl. Wright et al., 2016, 1273-1275)

Ein Vorteil von MSR-RF gegenüber CIF ist, dass für Splitpunkt- und Variablenselektion das gleiche Verfahren angewendet wird, und nicht wie bei CIF zwei unterschiedliche Tests durchgeführt werden. Beide Verfahren haben einen sehr viel niedrigeren Selektionsbias als übliche Random Forests auf Basis des CART-Algorithmus. Sowohl MSR-RF als auch CIF schneiden insbesonders besser ab, was den Bias an-

geht, wenn anstelle von Bootstrap-Stichproben kleinere Stichproben ohne Zurücklegen verwendet wurden. Laut Strobl et al. (2009, 30) kann beispielsweise die *permutation importance* nur verlässlich interpretiert werden, wenn ein unverzerrtes Kriterium zur Knotenteilung in Verbindung mit *subsampling* verwendet wird.

3.4 Extremely randomized trees

‚Extremely randomized trees' ist eine baumbasierte Ensemble-Methode, die von Geurts et al. (2006) entwickelt wurde. Mit dem sogenannten Extra-Trees-Algorithmus wird genau wie bei Random Forests ein Ensemble aus unbeschnittenen Entscheidungsbäumen erstellt, über die die Vorhersagefunktion gemittelt wird. Die Hauptunterschiede zum klassischen Random Forest bestehen darin, dass Splitpunkte zufällig ausgewählt werden, und dass keine Bootstrap-Stichproben, sondern für jeden Baum des Ensembles die gesamten Trainingsdaten verwendet werden.

Für die Erstellung eines Baumes werden wie bei Random Forest für jeden Split m_{try} Variablen zufällig ausgewählt. Für jede dieser ausgewählten Variablen wird dann ein Splitpunkt zufällig ausgewählt. Unter diesen m_{try} Splitpunkten wird der optimale nach einem Unreinheitsabnahme-Kriterium ausgewählt. Geurts et al. (2006, 8) verwenden die Entropie für Klassifikation und die geschätzte Varianz für Regression. Ein Parameter n_{min} legt die Knotengröße fest, ab der ein Knoten nicht weiter gespalten wird. Wie bei Random Forest ergibt sich die Vorhersagefunktion des gesamten Ensembles bei Regression aus dem Mittelwert der Vorhersagefunktionen der einzelnen Bäume und wird bei Klassifikation durch *majority vote* bestimmt.

Für $m_{try} = 1$ sprechen Geurts et al. (2006, 13) von ‚totally randomized trees'. Dabei wird also erst zufällig eine Variable und dann zufällig ein Splitpunkt für diese Variable ausgesucht, der dann automatisch der Splitpunkt ist, mit dem der Knoten geteilt wird. Das heißt, für die Teilung eines Knotens spielt die Abnahme der Knotenunreinheit keine Rolle mehr.

4 Beurteilung der Prädiktionsgüte

Um die Prädiktionsgüte eines Modells zu beurteilen, gibt es verschiedene Maße. In dieser Arbeit werden für binären Response die Fehlklassifikationsrate und die AUC berechnet, für metrischen Response der MSE. Für alle verwendeten Datensätze wurde jeweils ein Teil der Beobachtungen ausgeschlossen, um als Testdatensatz zu dienen, für den die jeweiligen Gütemaße berechnet werden. Zusätzlich werden die Gütemaße jeweils für die OOB-Daten berechnet. Auf eine Kreuzvalidierung wird

verzichtet, da es nicht primär Ziel dieser Arbeit ist, die bestmögliche Parametereinstellungen für die untersuchten Datensätze zu finden. Vielmehr geht es darum, einen allgemeinen Eindruck zu erhalten, wie sehr sich Parameteränderungen auf die Prädiktionsgüte auswirken können.

4.1 Binärer Response

Bei binärem oder allgemein kategorialem Response kann die Güte der Prädiktionen an Hand der Fehlklassifikationsrate E gemessen werden, das heißt, an dem Anteil der Beobachtungen, die von dem Modell falsch klassifiziert werden. Die Fehlklassifikationsrate ist damit das Gegenteil der sogenannten *Accuracy*, die den Anteil der richtig klassifizierten Beobachtungen angibt; es gilt $E = 1 - Accuracy$. Die Fehlklassifikationsrate ist demnach definiert als

$$E = \frac{1}{N_*} \sum_{i=1}^{N_*} \mathbb{I}(y_i \neq \hat{y}_i). \tag{4.1}$$

Berechnet man die Fehlklassifikationsrate für die OOB-Daten, gilt $N_* = N$, mit $N =$ Anzahl der Beobachtungen im Trainingsdatensatz. $\hat{y}_i$ entspricht in diesem Fall der OOB-Vorhersagefunktion $\hat{f}_{oob}(x_i)$ (vgl. Kapitel 3.3.3, Formeln 3.8 und 3.10). Berechnet man den Fehler für den Testdatensatz, so gilt $N_* = N_{test}$, wobei N_{test} der Anzahl der Beobachtungen im Testdatensatz entspricht. $\hat{y}_i$ entspricht dann der Random-Forest-Vorhersagefunktion $\hat{f}^{rf}(x_i)$ (vgl. Formel 3.5). Die Fehlklassifikationsrate wird im Folgenden auch kurz als Fehlerrate oder einfach als Fehler bezeichnet werden.

Ein anderes Gütemaß für binäre Responsevariablen ist die AUC. AUC steht für *area under the curve*, wobei die ROC-Kurve (Receiver Operating Characteristic Kurve) gemeint ist. Die AUC wird auf Basis der geschätzten Wahrscheinlichkeiten $\hat{\pi}_i(1)$ berechnet (bei einer Klassenbezeichnung 0/1; vgl. Formel 3.3). Die AUC entspricht der Wahrscheinlichkeit, dass die Vorhersagefunktion eines Modells einer zufällig ausgewählten Beobachtung mit Response 1 eine höhere Wahrscheinlichkeit $\hat{\pi}_i(1)$ zuweist als einer zufällig ausgewählten Beobachtung mit Response 0. Umso größer die AUC ist, umso besser unterscheiden die Wahrscheinlichkeitsschätzungen des Modells zwischen den beiden Klassen. (vgl. Ferri et al., 2009, 30)

Formal kann man die AUC so definieren: Sei $N^{(1)}$ die Anzahl der Beobachtungen mit Response 1 und $N^{(0)}$ die Anzahl der Beobachtungen mit Response 0. Bezeichne außerdem $\hat{\pi}_1(1)^{(1)}, ..., \hat{\pi}_{N^{(1)}}(1)^{(1)}$ die vom Modell geschätzten Wahrscheinlichkeiten für Response 1 für die Beobachtungen, deren Response tatsächlich 1 ist. Analog,

bezeichne $\hat{\pi}_1(1)^{(0)}, ..., \hat{\pi}_{N^{(0)}}(1)^{(0)}$ die vom Modell geschätzten Wahrscheinlichkeiten für Response 1 für die Beobachtungen, deren Response in Wirklichkeit 0 ist. Mit einer Zuordnung S(.,.) mit

$$S(a,b) = \begin{cases} 0 & \text{für } a < b \\ 0.5 & \text{für } a = b \\ 1 & \text{für } a > b \end{cases}$$

lässt sich die AUC schätzen als

$$AUC = \frac{\sum_{v=1}^{N^{(1)}} \sum_{w=1}^{N^{(0)}} S(\hat{\pi}_v(1)^{(1)}, \hat{\pi}_w(1)^{(0)})}{N^{(1)} N^{(0)}}. \tag{4.2}$$

(vgl. Hanley und McNeil 1982, 31)

4.2 Metrischer Response

Die Prädiktionsgüte von Modellen mit metrischem Response kann mit dem MSE (*mean squared error*) beurteilt werden. Der MSE entspricht der über alle Beobachtungen gemittelten quadrierten Differenz zwischen dem wahren Wert und dem durch das Modell prädiktierten Wert:

$$MSE = \frac{1}{N_*} \sum_{i=1}^{N_*} (y_i - \hat{y}_i)^2. \tag{4.3}$$

Auch hier kann N_* die Anzahl der Beobachtungen der Trainingsdaten bedeuten, N, oder die Anzahl der Beobachtungen der Testdaten, N_{test}, je nachdem, ob der MSE für die OOB-Daten (MSE_{oob}, vgl. Formel 3.9) oder für die Testdaten berechnet wird. Ebenso entspricht $\hat{y}_i$ je nachdem der OOB-Vorhersagefunktion (Formel 3.7) oder der Random-Forest-Vorhersagefunktion (Formel 3.4) für Regressionsbäume.

Wie bei der Fehlklassifikationsrate ist auch für den MSE 0 der niedrigste Wert, der eine perfekte Prädiktion bedeuten würde. Anders als die Fehlklassifikationsrate, die maximal den Wert 1 annehmen kann (alle Beobachtungen wurden falsch klassifiziert), ist der MSE allerdings nicht nach oben beschränkt. Die Höhe des MSEs hängt einerseits von der Prädiktionsgüte, andererseits aber auch von der Varianz der Daten ab. Wird im Folgenden für den Datensatz mit metrischem Response von Fehler gesprochen, ist damit immer der MSE gemeint.

5 Analysen

5.1 Verwendete R-Pakete

5.1.1 Das Paket ranger

Alle Analysen wurden in R mit dem package `ranger` von Wright (2018) durchgeführt. `ranger` ist ein Implementierung des Random Forest Algorithmus in R. Es ist für hochdimensionale Daten geeignet und kann Ensembles aus Klassifikationsbäumen, Regressionsbäumen und survival-Bäumen erstellen. Da es in dieser Arbeit nur um Klassifikationsbäume und Regressionsbäume, wird im Weiteren nicht mehr auf Einstellungen für survival-Bäume eingegangen.

Die zwei wichtigsten Befehle des Pakets sind `ranger()`, zum Erstellen von Random Forests, und `predict()` um für neue Daten den Response zu prädiktieren. Für den Befehl `ranger()` werden die Trainingsdaten mit dem Parameter `data` übergeben, der Response kann entweder als Modell (Response∼Kovariablen) mit dem Parameter `formula` oder durch den Parameter `dependent.variable.name` übergeben werden. Es gibt eine Reihe weiterer Parameter für `ranger()`, im Folgenden wird allerdings nur auf diejenigen eingegangen, die für die Analysen in dieser Arbeit relevant sind.

Die Anzahl der Bäume kann mit dem Parameter `num.trees` gesetzt werden. Der Default ist 500. Der Parameter `mtry` bestimmt die Anzahl der für jeden Split zufällig ausgewählten Kovariablen. Default ist die abgerundete Wurzel aus der Anzahl p der Kovariablen. Für die Wahl der minimalen Knotengröße gibt es den Parameter `min.node.size`, mit Default-Werten 1 für Klassifikation und 5 für Regression. Zu beachten ist, dass `min.node.size` nicht die kleinstmögliche Anzahl der Beobachtungen in einem Endknoten angibt, sondern die kleinstmögliche Knotengröße, ab der ein Knoten nicht mehr weiter geteilt wird. Die Endknoten können dementsprechend weniger Beobachtungen enthalten als `min.node.size`.

Mit dem Parameter `sample.fraction` kann man die Größe der Stichproben bestimmen, die für die einzelnen Entscheidungsbäume aus den Trainingsdaten gezogen werden. Möglich sind Werte im Bereich $(0, 1]$ um den Anteil am Umfang der Trainingsdaten anzugeben. Zusätzlich kann der Parameter `replace` auf `TRUE` oder `FALSE` gesezt werden, um zu bestimmen, ob mit Zurücklegen oder ohne Zurücklegen gezogen werden soll. Per Default werden Bootstrap-Stichproben verwendet, das heißt, Stichproben, die aus den Trainingsdaten mit Zurücklegen gezogen werden und den selben Umfang haben wie die Trainingsdaten (`replace=TRUE` und `sample.fraction=1`). Wird `replace=FALSE` gesetzt, so ist der Default für `sample.fraction` 0.632.

Auch die Splitting-Regel lässt sich unterschiedlich einstellen, über den Parameter `splitrule`. Für Klassifikation ist der Default `"gini"`, wobei wie in Kapitel 2 beschrieben von allen möglichen Splits (für die m_{try} zufällig ausgewählten Variablen) derjenige verwendet wird, der die größte Abnahme der Unreinheit gemessen am Gini-Index verursacht. Für Regression ist die Abnahme der Unreinheit gemessen durch die geschätzte Varianz innerhalb eines Knotens Default-Kriterium (`splitrule="variance"`). Alternativ lässt sich der Parameter `splitrule` auf `"extratrees"` (für Klassifikation und Regression) oder `"maxstat"` (für Regression) setzen.

Die Splitting-Regel `"extratrees"` hält sich im Wesentlichen an den Extra-Trees-Algorithmus wie er in Kapitel 3.4 beschrieben wird: Für jede der m_{try} Kovariablen wird ein Splitpunkt zufällig ausgesucht. Die Variable, mit deren zufällig ausgewähltem Splitpunkt die größte Abnahme der Unreinheit erreicht wird (gemessen am Gini-Index bzw. an der der geschätzten Varianz), wird mit diesem Splitpunkt zur Teilung des Knotens genommen. Allerdings werden (soweit nicht anders eingestellt mit `replace` und `sample.fraction`) Bootstrap-Stichproben verwendet und nicht die Trainingsdaten selbst. Außerdem kann über den zusätzlichen Parameter `num.random.splits` eingestellt werden, wie viele Splitpunkte für jede Variable ausgewählt werden. Man kann also auch für jede der m_{try} Variablen mehrere Splitpunkte zufällig auswählen, unter denen dann der optimale Splitpunkt ausgesucht wird. Per Default ist `num.random.splits = 1` gesetzt, wie beim ursrpünglichen Extra-Trees-Algorithmus.

Bei der Splitting-Regel `"maxstat"` wird zum Festlegen eines Splits verfahren wie in Kapitel 3.3.5 beschrieben: Im ersten Schritt wird für jede der m_{try} Variablen ein Splitpunkt ausgewählt, im zweiten Schritt erfolgt die Auswahl einer der Variablen mit ihrem zuvor ausgewählten Splitpunkt. Als zusätzliche Parameter gibt es bei dieser Splitting-Regel `alpha`, womit das Signifikanzniveau für die adjustierten p-Werte festgelegt werden kann, sowie den Parameter `minprop`, mit dem das Quantil der Kovariablenverteilung ε_1 zur Einschränkung der Splitpunkt-Auswahl gesetzt werden kann (vgl. Kapitel 3.3.5). Der Parameter `alpha` beeinflusst die Baumgröße: je kleiner `alpha`, desto weniger tief wächst der Baum. (vgl. Wright et al., 2016, 1279)

Für kategorialen Response besteht die Möglichkeit, anstelle eines normalen Klassifikations-Forest einen *probability forest* zu bilden, indem der Parameter `probability` auf `TRUE` gesetzt wird (Default ist `FALSE`). Bei einem solchen *probability forest* werden Wahrscheinlichkeiten für die einzelnen Klassen anstatt die Klassenzugehörigkeit selbst prädiktiert. Diese Wahrscheinlichkeiten werden für jeden Baum einzeln gebildet und dann über alle Bäume gemittelt. (vgl. Formel 3.3 und Kapitel 3.3.1) Die

minimale Knotengröße ist hier per Default 10.

Ein weiterer Parameter, `respect.unordered.factors`, legt fest, wie mit kategorialen Kovariablen verfahren wird. Mit `"ignore"` werden alle kategorialen Variablen als ordinal skaliert angesehen, das heißt, Splitpunkte werden wie für metrische Variablen nur in der Form $x \leq c$ / $x > c$ gesetzt. Mit `"partition"` werden dagegen alle möglichen binären Partitionen für die Knotenteilung berücksichtigt. Eine dritte Möglichkeit ist die Einstellung `"order"`. Diese Einstellung wird nur für Regression und binäre Klassifikation empfohlen. Für Regression werden die Klassen der Kovariable nach ihrem mittleren Response geordnet, und dann wie ordinal skalierte Variablen behandelt, für binäre Klassifikation werden die Klassen der Kovariable nach dem jeweiligen Anteil der Beobachtungen geordnet, die in die zweite Responseklasse fallen. Diese Ordnung der Kovariablenklassen erfolgt allerdings nur einmal und nicht für jeden Split neu. Für `splitrule ="extratrees"` ist der Default `"partition"`, für alle anderen Splitting-Regeln `"ignore"`.

Die Variablenwichtigkeit kann bei `ranger()` entweder durch die Unreinheitsabnahme bestimmt werden oder mit der *permutation importance* (vgl. Kapitel 3.3.4), indem der Parameter `importance` auf `"impurity"` oder `"permutation"` gesetzt wird. Das Unreinheitsmaß ist für Klassifikation der Gini-Index, für Regression die geschätzte Varianz. Mit der Default-Einstellung `"none"` wird gar keine Variablenwichtigkeit berechnet.

5.1.2 Weitere R-Pakete

Für die Berechnung von Fehlklassifikationsrate, AUC und MSE wurden die Befehle `Accuracy`, `AUC` und `MSE` aus dem Paket `MLmetrics` von Yan (2016) verwendet. Die Fehlklassifikationsrate wurde dabei mit 1-`Accuracy` berechnet. Außerdem wurde der Befehl `heatmap.2` aus dem Paket `gplots` von Warnes et al. (2016) zur Erstellung von *heatmaps* verwendet sowie der Befehl `minor.tick` aus dem Paket `Hmisc` von Harrell (2018).

5.2 Datenbeispiele

5.2.1 Brustkrebsdaten

Als Beispiel für Klassifikation werden Daten aus einer Studie von Hatzis et al. (2011) zur Überlebenszeit von Brustkrebspatientinnen nach einer Chemotherapie verwendet. Wie bei Boulesteix et al. (2017) wird als Response die (binär kodierte) RCB-Klasse (*rcb_class*) genommen, und als Kovariablen *age*, *nodal status*, *tumor stage*,

grade, *estrogen receptor* (*er*) und *progesterone receptor* (*pr*). Die RCB-Klasse (*residual cancer burden*) gibt Auskunft darüber, wieviel residuales Tumorgewebe noch vorhanden ist nach der Chemotherapie. Es gibt vier Stufen, RCB-0 bis RCB-III. In den Originaldaten von Hatzis et al. (2011) sind die ersten beiden Klassen schon zusammengefasst, in dieser Arbeit werden, wie bei Boulesteix et al. (2017) auch die Klassen II und III zusammengefasst, so dass die Variable binär kodiert ist mit den Ausprägungen RCB-0/I vs. RCB-II/III. Von den sechs oben genannten Kovariablen ist nur *age* metrisch, die restlichen Kovariablen sind kategorial: *tumor stage* hat fünf Kategorien ($T0, ..., T4$), *nodal status* vier ($N0, .., N3$), *grade* drei (1,2,3), und *er* sowie *pr* sind binär mit N vs. P.

Die Daten liegen in zwei Datensätzen vor, die als Trainings- und Testdatensatz genutzt werden. Die ursprünglichen Daten enthalten 310 (Trainingsdatensatz) bzw. 198 (Testdatensatz) Beobachtungen. 29 Beobachtungen aus dem Trainingsdatensatz und 90 Beobachtungen aus dem Testdatensatz wurden wegen fehlenden Werten ausgeschlossen. Aus dem Trainingsdatensatz wurden zusätzlich vier Beobachtungen ausgeschlossen, die entweder für *er* oder *pr* die Ausprägung 'I' hatten. Es bleiben damit 277 Beobachtungen im Trainings- sowie 108 Beobachtungen im Testdatensatz. Im Trainingsdatensatz fallen nur 28.88% (80 von 277) der Beobachtungen in die Klasse RCB-0/1, im Testdatensatz sind es mit 25.93% (28 von 108) noch weniger.

5.2.2 GlaucomaM

Da die Brustkrebsdaten einerseits nur sehr wenige Kovariablen haben und andererseits die beiden Klassen des Responses so ungleich verteilt sind, wird als weiteres Beispiel für Daten mit binärem Response der Datensatz *GlaucomaM* aus dem R-Paket TH.data von Hothorn (2016) verwendet. Die Responsevariable ist *Class* und gibt an ob eine Glaukomerkrankung (grüner Star) vorliegt (Ausprägung 'glaucoma') oder nicht (Ausprägung 'normal'). Das Verhältnis 'glaucoma':'normal' beträgt 1:1 im Datensatz. Die 62 Kovariablen erfassen unterschiedliche Messungen an der Papille des Sehnervkopfes und sind alle metrisch. Man kann davon ausgehen, dass die Variablen zum Teil stark miteinander korrelieren.

Der Datensatz enthält 196 Beobachtungen. Es liegen keine fehlenden Werte vor, daher wurden keine Beobachtungen ausgeschlossen. Die Daten wurden in einen Trainings- und einen Testdatensatz unterteilt. Dazu wurden ca. 30% der Daten (59 Beobachtungen) zufällig ausgewählt für den Testdatensatz. Der Trainingsdatensatz enthält demnach 137 Beobachtungen. Das Verhältnis 'glaucoma':'normal' beträgt in beiden Datensätzen ungefähr 1:1 (im Trainingsdatensatz 69:68 und im Testdatensatz

29:30).

5.2.3 NHANES-Daten

Daten aus dem 2007/2008-Zyklus des National Health and Nutrition Examination Surveys (NHANES) dienen als Datenbeispiel mit metrischem Response. Bei NHANES handelt es sich um eine Studie, die jährlich in den USA durchgeführt wird, um den Gesundheitsstatus und Ernährungszustand von Kindern und Erwachsenen in den USA zu untersuchen. In Interviews und körperlichen Untersuchungen werden demographische, sozioökonomische, gesundheits- und ernährungsbezogene Merkmale erfasst.

Für diese Arbeit wird wie bei Rospleszcz et al. (2016) eine Auswahl von 28 Kovariablen verwendet und als Response der CRP-Wert. CRP steht für C-reaktives Protein, ein Plasmaprotein, das vermehrt bei Entzündungszuständen vorkommt. (vgl. Rospleszcz 2016) Acht Kovariablen sind metrisch, zum Beispiel *age*, *bmi*, *cholesterol*. Neun Kovariablen sind binär und beziehen sich hauptsächlich darauf, ob Erkrankungen wie Asthma oder Herzinfarkt vorliegen bzw. aufgetreten sind. Die restlichen 11 Kovariablen sind kategorial mit mehr als zwei Kategorien. Dabei sind sie zum Teil nominal (z. B. *race*, *medicalPlaceToGo*) und zum Teil ordinal skaliert (z. B. *education*, *income*).

Der Datensatz enthält 1914 Beobachtungen. Es liegen keine fehlenden Werte vor, daher wurden keine Beobachtungen ausgeschlossen. Die Daten wurden in einen Trainings- und einen Testdatensatz unterteilt. Dazu wurden ca. 30% der Daten (574 Beobachtungen) zufällig ausgewählt für den Testdatensatz. Der Trainingsdatensatz enthält demnach 1340 Beobachtungen.

5.3 Prädiktionsgüte

In diesem Kapitel soll an Hand der drei Datenbeispiele untersucht werden, wie sich unterschiedliche Einstellungen der Parameter `mtry`, `splitrule`, `min.node.size`, `replace` und `sample.fraction` auf die Prädiktionsgüte auswirken.

5.3.1 Ergebnisse für die Brustkrebsdaten

Abbildung 5.1 zeigt den geschätzten OOB-Fehler für die Brustkrebsdaten in Abhängigkeit von der Anzahl der Bäume. Betrachtet werden Random Forests mit $B = 1, ..., 1500$ Bäumen. Die Fehlerraten der einzelnen Forests schwanken ziemlich stark, zu Beginn erkennt man aber im Mittel eine Abnahme des Fehlers mit steigender

Baumzahl, und auch die Schwankungen werden kleiner. Ab ca. 200 Bäumen ist aber im Mittel keine Verringerung des Fehlers mehr zu erkennen, und auch die Schwankung nimmt mit steigender Baumanzahl nur sehr geringfügig ab. Für die folgenden Analysen mit den Brustkrebsdaten werden deswegen die Gütemaße über hundert Forests mit je 500 Bäumen gemittelt.

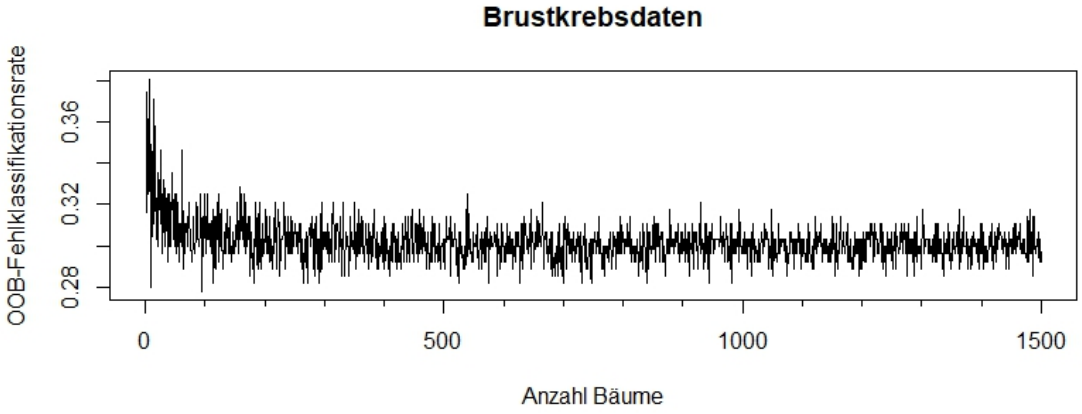


Abbildung 5.1: OOB-Fehlklassifikationsrate in Abhängigkeit von der Anzahl der Bäume (`num.trees`) für die Brustkrebsdaten

Da alle kategorialen Kovariablen der Brustkrebsdaten ordinal skaliert sind (mit Ausnahme der binären), wird bei den folgenden Analysen mit diesem Datensatz der Parameter `respect.unordered.factors` für beide Splitting-Regeln auf `"ignore"` gesetzt. Die Einstellung `"partition"`, die für `"extratrees"` Default ist, ist für ordinal skalierte Variablen nicht sinnvoll.

Der Parameter `mtry`. Zuerst soll der Einfluss des Parameter `mtry` auf die Prädiktionsgüte untersucht werden, also der Einfluss der Anzahl der zufällig ausgewählten Kovariablen für jeden Split. Die Default-Einstellung für `mtry` liegt für die Brustkrebsdaten bei 2 (für $p = 6$ Kovariablen gilt $\lfloor\sqrt{p}\rfloor = 2$). Abbildungen 5.2 und 5.3 zeigen die Fehlklassifikationsrate und die AUC für die OOB-Daten und den Testdatensatz für alle möglichen Einstellungen von `mtry`, links jeweils mit `splitrule = "gini"`, rechts jeweils mit `splitrule = "extratrees"`. Jeder Boxplot entspricht 100 Random Forests mit jeweils 500 Bäumen. Bei `"extratrees"` ist als Anzahl der random splits immer 1 gewählt. Erstens scheint die Anzahl der zufällig ausgewählten Splits kaum Einfluss auf die Prädiktionsgüte zu haben (vgl. Abbildung 8.1 im Anhang). Zweitens sind zwei der Kovariablen binär (*er* und *pr*); für sie gibt es nur einen möglichen Splitpunkt. Für `num.random.splits` > 1 würden diese beiden

Kovariablen benachteiligt werden.

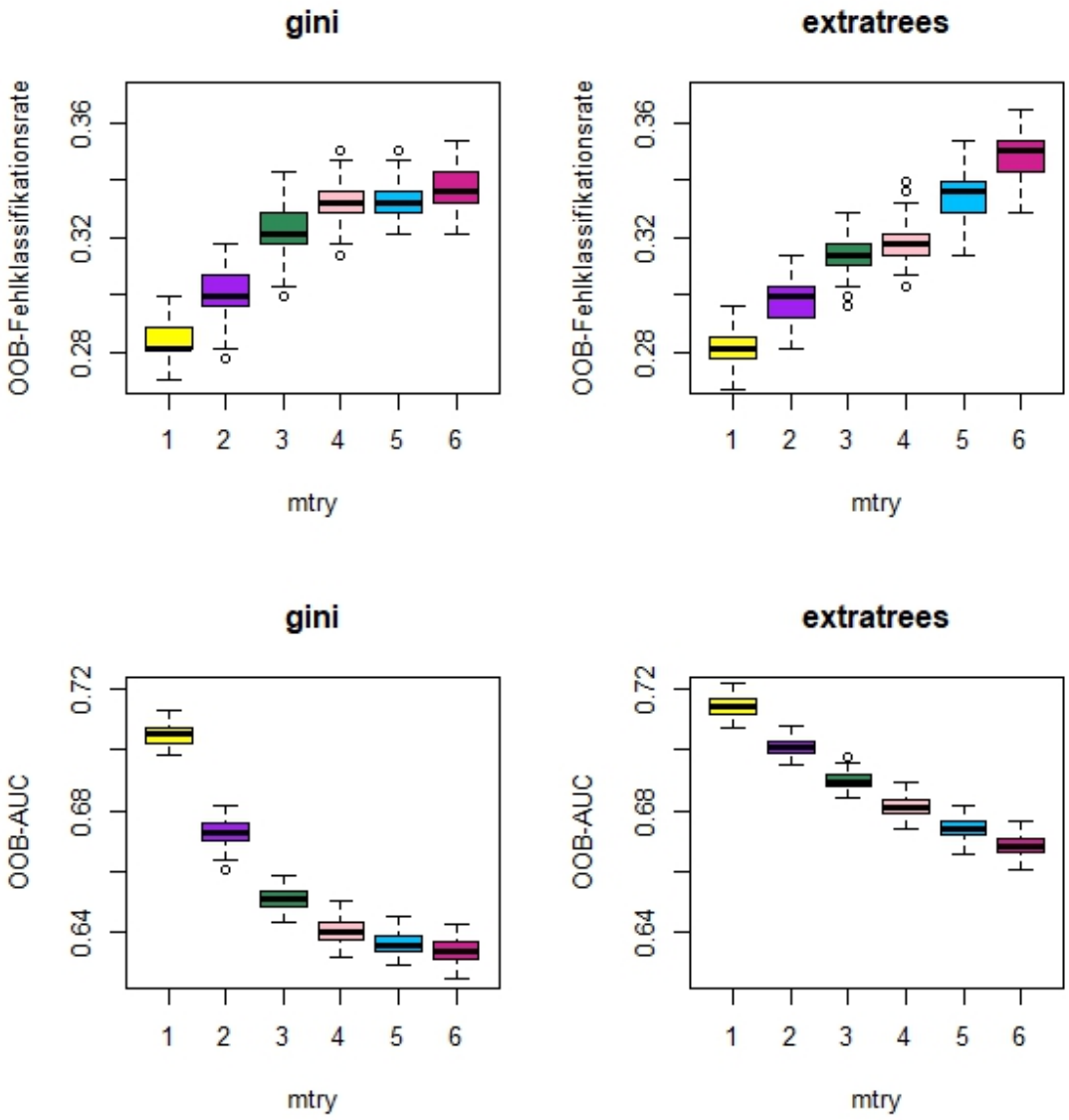


Abbildung 5.2: OOB-Fehlklassifikationsrate (oben) und OOB-AUC (unten) in Abhängigkeit von `mtry`für die Brustkrebsdaten. Links jeweils für `splitrule = "gini"`, rechts jeweils für `splitrule = "extratrees"` (mit `num.random.splits = 1`).

Wie Abbildungen 5.2 und 5.3 zeigen, liefert bei `splitrule = "gini"` jeweils `mtry=1` die niedrigste Fehlklassifikationsrate und den höchsten AUC-Wert, sowohl für die OOB-Daten als auch für den Testdatensatz. Mit `"extratrees"` schneiden für die OOB-Daten ebenfalls sowohl bei der Fehlklassifikationsrate als auch bei der AUC Random Forests mit `mtry=1` am besten ab, bei den Testdaten erzielt die niedrigste Fehlerrate `mtry=2`, die Default-Einstellung, den höchsten AUC-Wert hingegen

erreicht wieder `mtry=1`. Für `mtry>2` nimmt die Prädiktionsgüte mit steigendem `mtry` sowohl für die OOB- als auch für die Testdaten immer weiter ab, unabhängig davon, welche Splitting-Regel verwendet wurde und ob Fehlklassifikationsrate oder AUC betrachtet werden.

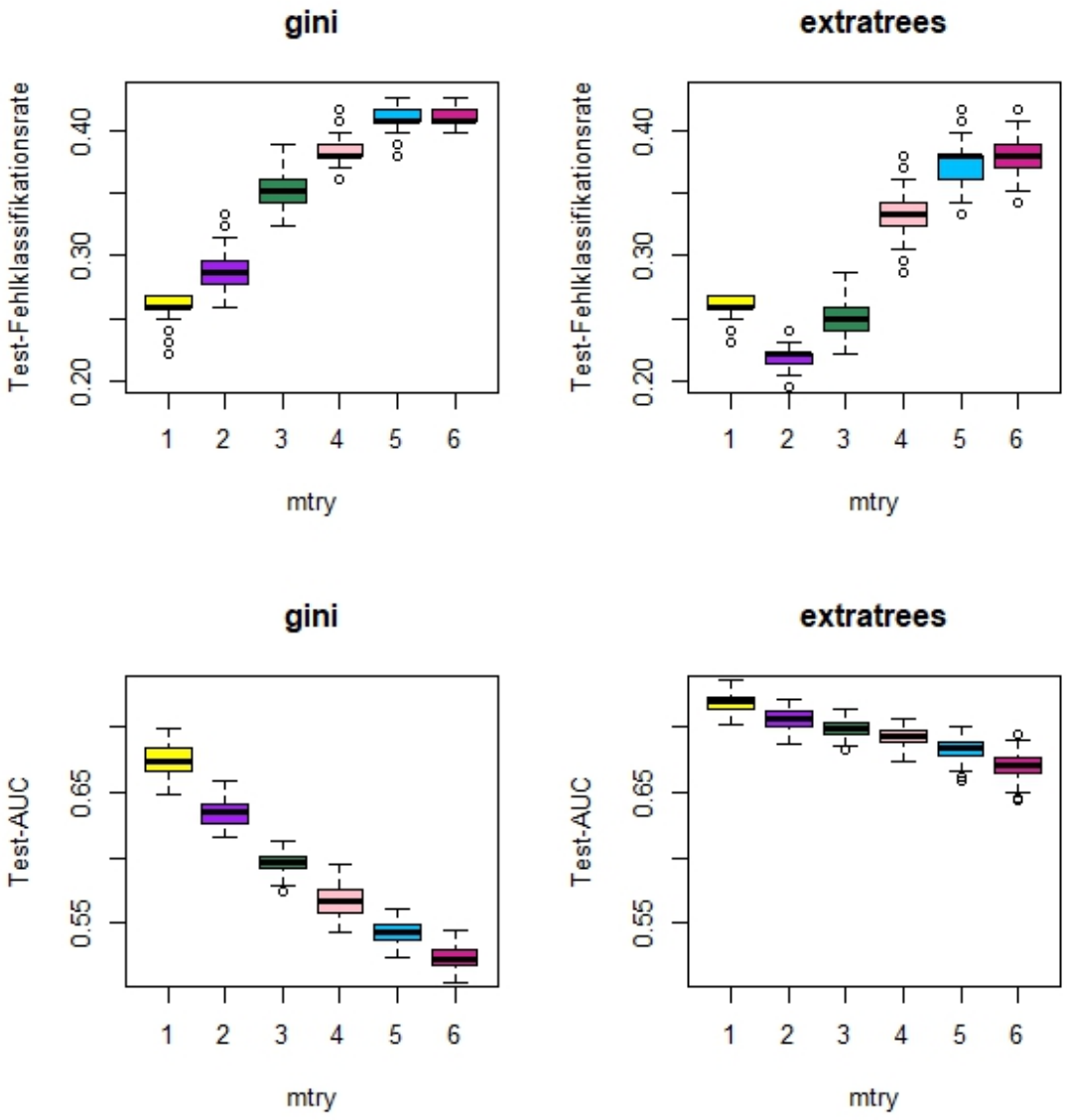


Abbildung 5.3: Test-Fehlklassifikationsrate (oben) und Test-AUC (unten) in Abhängigkeit von `mtry` für die Brustkrebsdaten. Links jeweils für `splitrule = "gini"`, rechts jeweils für `splitrule = "extratrees"` (mit `num.random.splits = 1`).

Tabelle 5.1 zeigt entsprechend zu den Abbildungen 5.2 und 5.3 die Mittelwerte der Fehlklassifikationsraten und der AUCs für die OOB-Daten und die Testdaten, jeweils für jede der beiden Splitting-Regeln. Möchte man die Prädiktionsgüte

in Hinblick auf die Splitting-Regeln vergleichen, so erreicht man fast immer mit `"extratrees"` ein besseres Ergebnis. Lediglich beim OOB-Fehler für `mtry` gleich 5 oder 6 sowie beim Testfehler für `mtry=1` schneidet `"gini"` etwas besser ab. Den im Mittel niedrigsten Fehler für die OOB-Daten (0.282) erzielt man mit der Kombination aus `mtry=1` und der Splitting-Regel `"extratrees"`. Für `"gini"` und `mtry=1` ist der Fehler allerdings kaum höher. Für `mtry>1` liegt die OOB-Fehlklassifikationsrate unabhängig von der Splitting-Regel bei ca. 0.3 oder mehr. Damit ist sie höher als der Anteil der Beobachtungen mit Ausprägung RCB-0/I (0.289), das heißt, für `mtry>1` werden mit der Vorhersagefunktion des Random Forests mehr Beobachtungen falsch klassifiziert, als wenn man einfach alle Beobachtungen als RCB-II/III klassifizieren würde.

`mtry`	1	2	3	4	5	6
gini OOB-Fehlklassifikationsrate	0.283	0.300	0.322	0.331	0.333	0.336
ET OOB-Fehlklassifikationsrate	0.282	0.297	0.313	0.318	0.334	0.348
gini Test-Fehlklassifikationsrate	0.259	0.289	0.351	0.385	0.410	0.410
ET Test-Fehlklassifikationsrate	0.262	0.221	0.251	0.331	0.373	0.381
gini OOB-AUC	0.705	0.673	0.651	0.640	0.636	0.634
ET OOB-AUC	0.714	0.701	0.690	0.681	0.674	0.668
gini Test-AUC	0.675	0.636	0.596	0.567	0.543	0.523
ET Test-AUC	0.719	0.706	0.699	0.693	0.683	0.670

Tabelle 5.1: Gerundete Mittelwerte der Fehlklassifikationsraten und AUCs für die Brustkrebsdaten in Abhängigkeit von `mtry` (vgl. Abbildungen 5.2 und 5.3). ET steht für `splitrule = "extratrees"`, gini für `splitrule = "gini"`. Der niedrigste Wert (bzw. der höchste für die AUC) in jeder Zeile ist rot markiert.

Ähnlich sieht es bei den Prädiktionen für die Testdaten aus. Hier erreicht man fast durchgehend mit `"extratrees"` bessere Werte als mit `"gini"`. Nur für `mtry`=1 ist der Fehler für `"gini"` etwas geringer als für `"extratrees"`. Die Fehlklassifikationsrate von `"gini"` und `mtry`=1 entspricht mit 0.259 genau dem Anteil der Beobachtungen mit Ausprägung RCB-0/I, also der Fehlklassifikationsrate, die man erhielte, wenn alle Beobachtungen als RCB-II/III klassifiziert werden würden. Einen niedrigeren Wert (0.221) erhält man im Mittel nur für `mtry`=2 in Kombination mit der Splitting-Regel `"extratrees"`.

Minimale Knotengröße. Ein weiterer Parameter, der Einfluss auf die Prädiktionsgüte haben kann, ist die minimale Knotengröße. Abbildung 5.4 zeigt die Fehlklassifikationsrate für die OOB-Daten (oben) und die Testdaten (unten) in Abhängigkeit von der minimalen Knotengröße, dunkelblau und mit durchgezogener Linie

für `splitrule="gini"` und `mtry`=2, rot und gepunktet für `splitrule="extratrees"` und `mtry`=2, blau und mit Strich-Punkt-Linie für `splitrule="gini"` und `mtry`=1, pink und gestrichelt für `splitrule="extratrees"` und `mtry`=1. Dabei sind wieder jeweils die Mittelwerte von 100 Random Forests mit je 500 Bäumen genommen worden. Die Werte für `min.node.size` laufen von 1 bis 125, da der Fehler für alle dargestellten Einstellungen ab ca. 120 konstant bleibt bei 0.289, was dem Anteil der Beobachtungen mit RCB-0/I am Gesamtdatensatz entspricht. Anscheinend werden für Forests mit einer sehr großen minimalen Knotengröße, die also nur aus sehr kleinen Bäumen bestehen, alle Beobachtungen der Klasse RCB-II/III zugeordnet. Der Default-Wert für `min.node.size` liegt für Klassifikation bei 1.

Wie man in Abbildung 5.4 sieht, hängt der Einfluss der minimalen Knotengröße für sehr davon ab, welche Splitting-Regel und welcher Wert für `mtry` gewählt wird. Für `mtry`=2 schneiden Forests mit dem Default 1 für `min.node.size` auf den OOB-Daten für beide Splitting Regeln schlecht ab. Für `"gini"` erzielt man mit Knotengrößen von ca. 30-55 die besten Ergebnisse, für `"extratrees"` mit Knotengrößen von ca. 20-30. Die Werte sind für beide Splitting-Regeln vergleichbar groß. Ist hingegen `mtry` gleich 1 gesetzt, liefern für beide Splitting-Regeln kleinere Knotengrößen von ca. 5-25 (für `"gini"`) bzw. 5-15 (für `"extratrees"`) im Mittel niedrigere Fehlerraten. Die Unterschiede, die man mit verschiedenen Knotengrößen erreicht, sind aber für `mtry` gleich 1 eher gering.

Auch die gemittelten Fehlklassifikationsraten für die Testdaten, dargestellt in Abbildung 5.4 unten, bleiben für alle gezeigten Parametereinstellungen ab einer minimalen Knotengröße von ca. 110 konstant bei 0.259, dem Wert, der dem Anteil der kleineren Klasse an den Gesamtdaten entspricht. Ansonsten unterscheiden sich die Verläufe der einzelnen Kurven aber sehr stark von den Ergebnissen für die OOB-Daten. Für `mtry`=1 hat die minimale Knotengröße wieder für beide Splitting-Regeln nur wenig Einfluss. Für `"gini"` erweisen sich dabei sehr kleine Knotengrößen (0-10) am besten. Für `"extratrees"` erreicht der Fehler schon ab einer Knotengröße von 30 kaum noch Änderung; für kleinere Knotengrößen ist er geringfügig höher. Ganz anders für `mtry`=2: Hier hat die Knotengröße für beide Splitting-Regeln erheblichen Einfluss. Setzt man `splitrule="gini"`, so ist der Fehler für sehr kleine Knotengrößen im Mittel am größten, während die Fehlerrate für Knotengrößen zwischen 90 und 100 deutlich niedriger ist und bei `min.node.size`=96 den niedrigsten Wert erreicht (0.233). Für `"extratrees"` hingegen ist in Kombination mit `mtry`=2 die gemittelte Fehlklassifikationsrate für Knotengrößen kleiner 80 durchgehend sehr niedrig, am niedrigsten im Bereich 10-30. Mit ca. 0.22 liefert die Kombination aus `"extratrees"`, `mtry`=2 und `min.node.size` < 80 den deutlich niedrigsten Fehler.

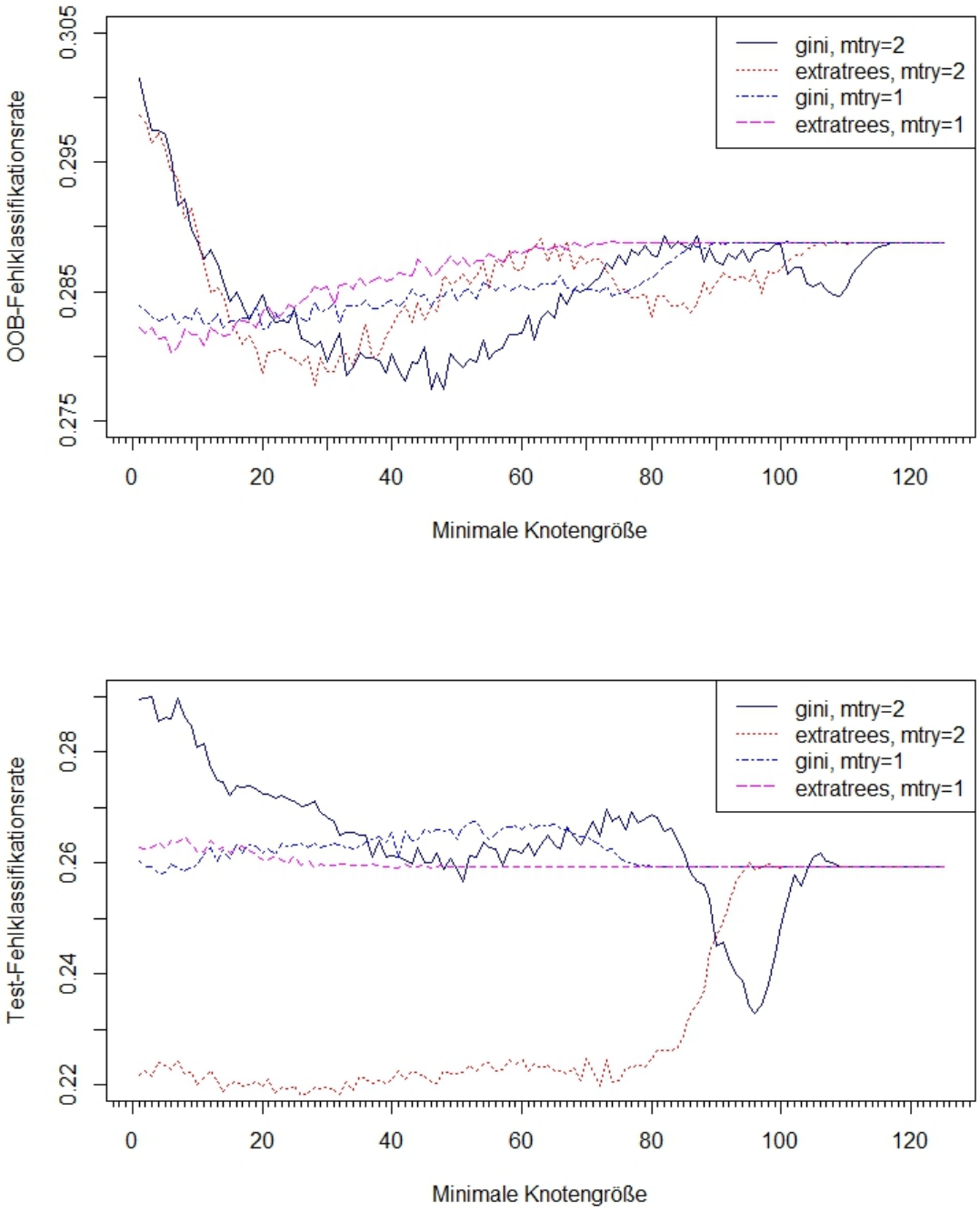


Abbildung 5.4: Fehlklassifikationsrate für die Brustkrebsdaten in Abhängigkeit von der minimalen Knotengröße. Oben: für die OOB-Daten, unten: für die Testdaten.

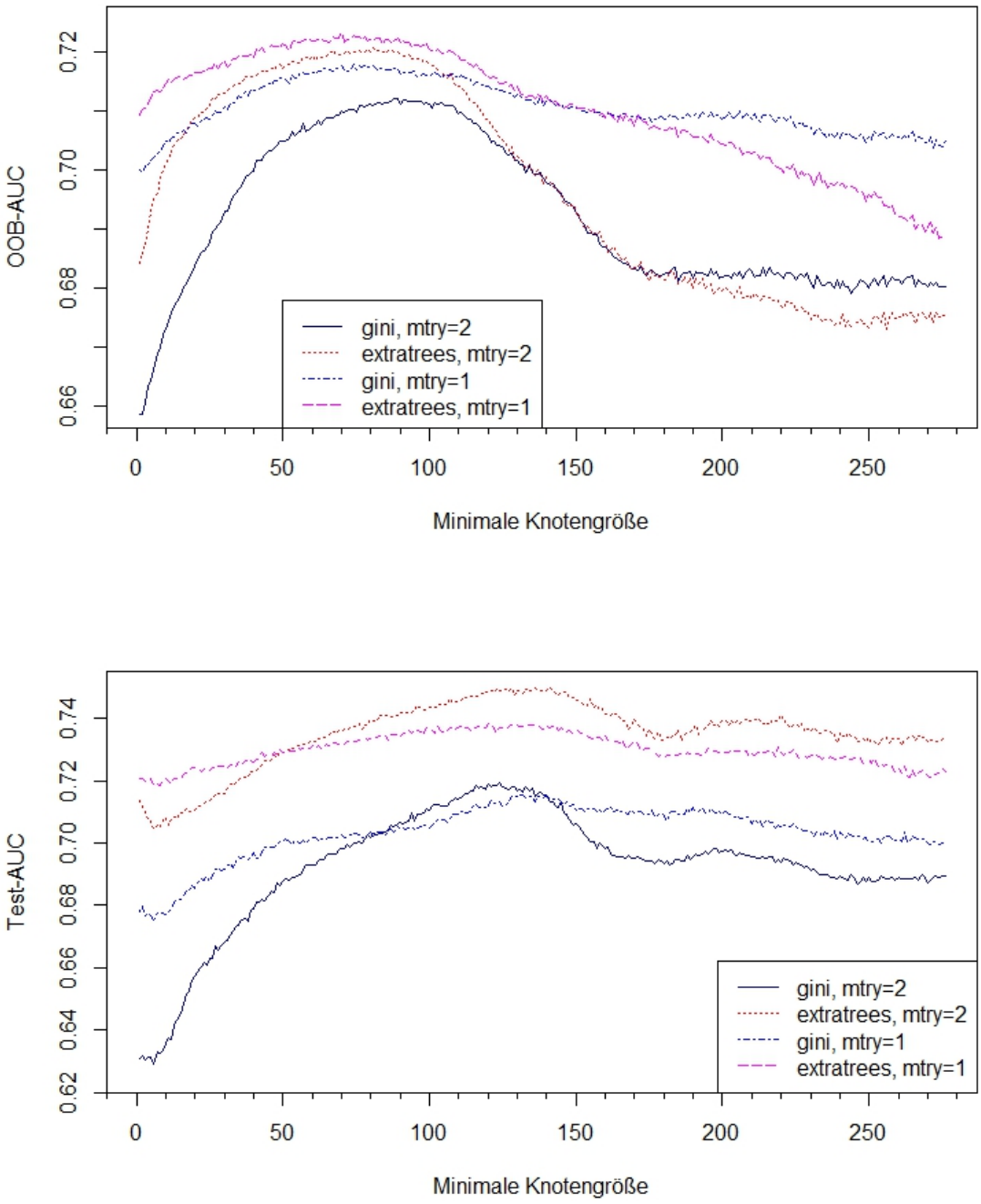


Abbildung 5.5: AUC für die Brustkrebsdaten in Abhängigkeit von der minimalen Knotengröße. Oben: für die OOB-Daten, unten: für die Testdaten.

Die gemittelten AUCs in Abhängigkeit von der minimalen Knotengröße, berechnet für OOB-Daten und Testdaten, sind in Abbildung 5.5 dargestellt. Da die gemittelten AUC-Werte nicht ab einer bestimmten Knotengröße konstant werden, laufen

die Werte für `min.node.size` von 1 bis 276. Für `min.node.size` = 277 bestünde jeder Baum nur aus dem Wurzelknoten. Da der Parameter nicht die kleinstmögliche Größe der Endknoten angibt, sondern die kleinstmögliche Größe, für die ein Knoten noch geteilt werden kann, ist aber für minimale Knotengrößen ≤ 276 eine Teilung des Wurzelknotens möglich.

Im Gegensatz zu den Fehlklassifikationsraten ähneln sich die einzelnen AUC-Kurven viel mehr in ihrem Verlauf. Die höchsten AUC-Werte erhält man bei allen gezeigten Einstellungen für eine große, aber nicht zu große Knotengröße, bei den OOB-Daten jeweils für eine minimale Knotengröße von ca. 50-120, bei den Testdaten ca. 100-150. Der negative Einfluss von sehr kleinen Knotengrößen ist bei den OOB-Daten generell für `mtry`=2 sowie für die Testdaten für `mtry`=2 in Verbindung mit `"gini"` sehr groß. Vergleicht man die beiden Splitting-Regeln miteinander, so lässt sich sagen, dass `"extratrees"` bei den Testdaten für alle Knotengrößen deutlich besser abschneidet als `"gini"`. Bei den OOB-Daten liefert `"extratrees"` für `mtry`=1 bis zu einer minimalen Knotengröße von ca. 150 im Mittel höhere AUC-Werte als `"gini"`, und für `mtry`=2 liefert `"extratrees"` bis zu einer Knotengröße von ca. 130 höhere gemittelte AUC-Werte.

Stichprobengröße und subsampling vs. Bootstrap. Abbildung 5.6 zeigt die OOB-Fehlklassifikationsraten bei Subsampling (Ziehen ohne Zurücklegen) mit verschiedenen Stichprobengrößen und für unterschiedliche Splitting-Regeln jeweils für `mtry`=1 und `mtry`=2. Die Stichprobengrößen sind dabei als prozentualer Anteil (sample fraction) des Trainingdatensatzumfangs $N = 277$ angezeigt. Die Anteile reichen von 5% bis 85%, in 10-Prozentpunkt-Schritten. Allerdings wurde 63.2% an Stelle von 65% genommen, da das die Default-Einstellung für Ziehen ohne Zurücklegen (`replace = FALSE`) ist. Auf Anteile über 85% ist in der Darstellung verzichtet worden, da bei den meisten Ergebnissen die Fehler ab einer gewissen Anteilsgröße mit steigender Größe nur zunehmen. Zum Vergleich wird auch der Fehler für Bootstrap-Stichproben (BT) mit abgebildet. Wie schon zuvor steht jeder Boxplot für 100 Random Forests mit je 500 Bäumen. In Tabelle 5.2 sind die entsprechenden Mittelwerte für alle in der Abbildung gezeigten Kombinationen aufgelistet.

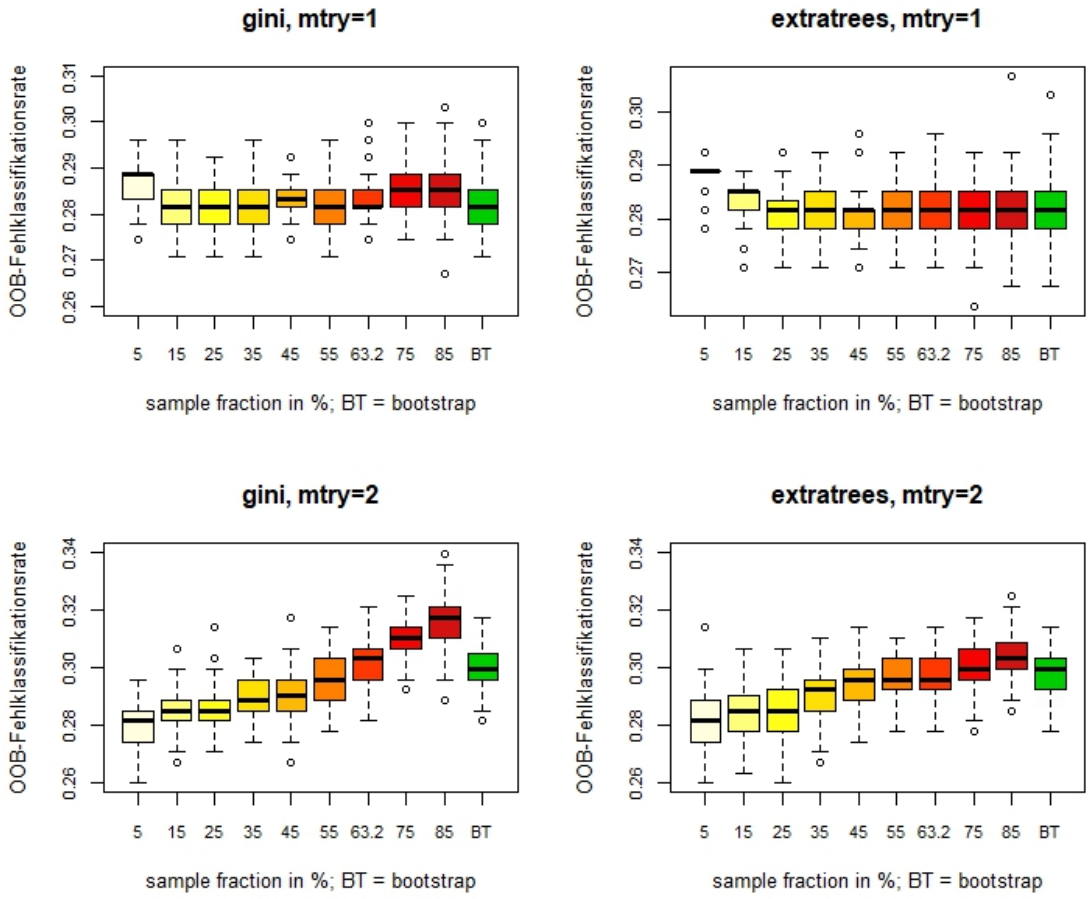


Abbildung 5.6: OOB-Fehlklassifikationsrate für die Brustkrebsdaten in Abhängigkeit von der Stichprobengröße

Abbildung 5.6 und Tabelle 5.2 zeigen, dass für beide Splitting-Regeln die Verteilung der Boxplots nach der Stichprobengröße recht ähnlich ist und sich auch die Höhe der Fehlklassifikationsrate für die meisten Anteilsgrößen zwischen den beiden Splitting-Regeln nicht stark unterscheidet. Einen deutlicheren Unterschied scheint es zu machen, ob `mtry` gleich 1 oder gleich 2 gesetzt wird. Für `mtry`=1 liegen die Werte des OOB-Fehlers im Mittel unabhängig von der Anteilsgröße ziemlich nahe beieinander, leicht höhere Fehler ergeben sich für beide Splitting-Regeln mit Anteilen von 5% und für `"gini"` auch mit Anteilen von 75% und 85%. Dagegen erzielt man für `mtry`=2 unabhängig von der Splitting-Regel mit einer Anteilsgröße von 0.05 die im Mittel niedrigste Fehlerrate, obwohl der Datensatz nicht so groß ist und 5% nur 13 Beobachtungen entsprechen. Allerdings sind auch die niedrigsten gemittelten Fehlerraten mit ca. 0.28 verhältnismäßig hoch. Mit steigender Anteilsgröße nimmt für `mtry`=2 die Fehlklassifikationsrate noch zu, für `"gini"` ewas stärker als

für `"extratrees"`.

Anteil	0.05	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	BT
gini, `mtry`=1	0.286	0.282	0.281	0.282	0.283	0.283	0.283	0.286	0.286	0.283
ET, `mtry`=1	0.288	0.283	0.281	0.281	0.280	0.281	0.282	0.281	0.282	0.282
gini, `mtry`=2	0.279	0.285	0.286	0.290	0.291	0.296	0.302	0.311	0.316	0.301
ET, `mtry`=2	0.282	0.284	0.286	0.292	0.294	0.296	0.297	0.301	0.304	0.298

Tabelle 5.2: Mittelwerte der OOB-Fehlklassifikationsraten für die Brustkrebsdaten in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 5.6). Der niedrigste Wert (bzw. der höchste für die AUC) in jeder Zeile ist rot markiert.

Mit Bootstrap-Stichproben erreicht man für die OOB-Daten nicht die Tiefstwerte, die man zum Teil für kleinere Stichproben ohne Zurücklegen erhält, aber für `mtry`=1 sind die Unterschiede sehr gering, so dass sich hier durch Subsampling kaum Verbesserungen erzielen lassen. Aus Tabelle 5.2 ist auch ersichtlich, dass für alle dargestellten Kombinationen aus `mtry` und `splitrule` die OOB-Fehlerrate für die Bootstrap-Stichprobe ziemlich genauso hoch ist wie für Subsampling mit der Default-Einstellung 0.632.

Analog zu den OOB-Fehlerraten aus Abbildung 5.6 zeigt Abbildung 5.7 die Fehlerraten für die Testdaten für alle Kombinationen aus `"gini"` vs. `"extratrees"` und `mtry`=1 vs. `mtry`=2. Die dazugehörigen Mittelwerte sind in Tabelle 5.3 zu sehen. Ähnlich wie bei den OOB-Fehlerraten sind hier für `mtry`=1 nur sehr geringe Unterschiede sowohl zwischen den einzelnen Anteilsgrößen als auch zwischen den beiden Splitting-Regeln. Für `mtry`=2 dagegen sind größere Unterschiede zwischen den einzelnen Anteilsgrößen und vor allem auch zwischen den beiden Splitting-Regeln zu erkennen. Für `"extratrees"` wird die Fehlklassifikationsrate um so höher, um so kleiner die Stichprobe ist, wohingegen für `"gini"` die Fehlklassifikationsrate mit zunehmender Stichprobengröße ansteigt und der niedrigste Wert im Mittel mit 0.05 erzielt wird.

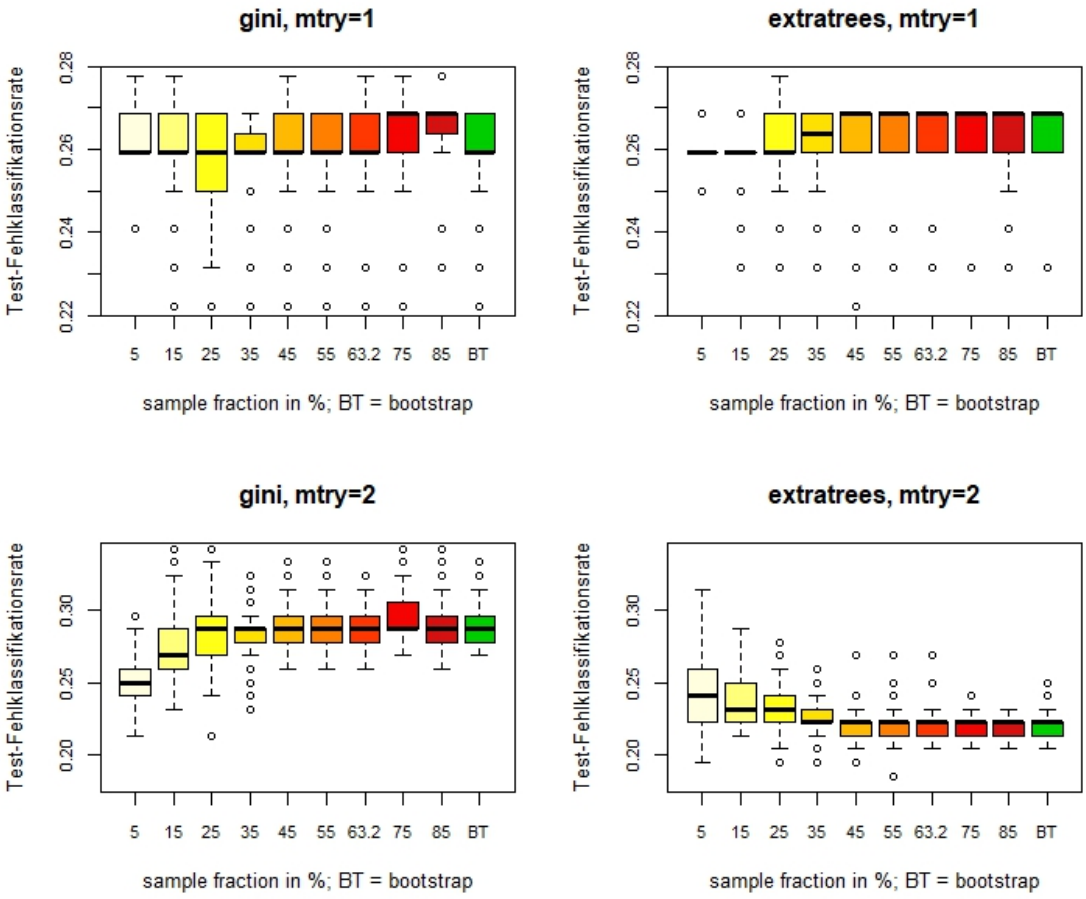


Abbildung 5.7: Test-Fehlklassifikationsrate für die Brustkrebsdaten in Abhängigkeit von der Stichprobengröße

Insgesamt lassen sich bei `mtry`=2 für alle Anteilsgrößen mit `"extratrees"` im Mittel niedrigere Fehlerraten für den Testdatensatz erreichen. Die niedrigsten Fehlerraten ergeben sich für die Kombination aus `"extratrees"`, `mtry`=2 und Anteilsgrößen zwischen 0.45 und 0.85, wobei der mittlere Fehler für die Bootstrap-Stichprobe mit `"extratrees"` und `mtry`=2 auch nur unwesentlich höher ist.

Anteil	0.05	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	BT
gini, `mtry`=1	0.263	0.260	0.255	0.256	0.261	0.260	0.262	0.264	0.266	0.259
ET, `mtry`=1	0.259	0.261	0.260	0.262	0.262	0.262	0.264	0.262	0.262	0.263
gini, `mtry`=2	0.251	0.274	0.285	0.282	0.288	0.289	0.285	0.295	0.293	0.292
ET, `mtry`=2	0.244	0.238	0.233	0.224	0.221	0.221	0.219	0.219	0.220	0.223

Tabelle 5.3: Mittelwerte der Test-Fehlklassifikationsraten für die Brustkrebsdaten in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 5.7). Der niedrigste Wert (bzw. der höchste für die AUC) in jeder Zeile ist rot markiert.

Betrachtet man anstatt der Fehlerrate die AUCs, so bietet sich ein einheitlicheres Bild. Unabhängig von der verwendeten Splitting-Regel und davon, ob `mtry` gleich 1 oder 2 gewählt wurde und unabhängig davon, ob man OOB-Daten oder Testdaten betrachtet, sind die Boxplots für die unterschiedlichen Anteilsgrößen sehr ähnlich verteilt (vgl. Abbildungen 8.2 und 8.3 sowie Tabellen 8.1 und 8.2 im Anhang). Den höchsten AUC-Wert erzielt man jeweils für `sample.fraction` = 0.15 (entspricht 41 von 277 Beobachtungen), und mit steigender Anteilsgröße nimmt die AUC ab. Die AUC-Werte der Bootstrap-Stichprobe gehören immer zu den niedrigsten. Vergleicht man die AUCs in Hinblick auf die Splitting-Regel, fällt auf, dass `"extratrees"` fast immer deutlich höhere Werte liefert. Besonders ausgeprägt ist der Unterschied zwischen den Splitting-Regeln für die Testdaten.

Zusammenfassung. Der Einfluss unterschiedlicher Parametereinstellungen auf die Prädiktionsgüte für die Brustkrebsdaten lässt sich nicht leicht zusammenfassen, da sich die Auswirkungen der Parameterveränderungen größtenteils sehr stark danach unterscheiden, ob die OOB-Daten oder die Testdaten betrachtet werden, und auch danach, ob man die Fehlklassifikationsrate oder die AUC betrachtet. Allein die Verwendung der Splitting-Regel `"extratrees"` (mit `respect.unordered.factors="ignore"`) bei sonst gleich bleibenden Default-Einstellungen verringert die Fehlklassifikationsrate für die Testdaten um 23.5% (0.221 anstatt 0.289, vgl. Tabelle 5.1). Die OOB-Fehlklassifikationsrate verringert sich hingegen kaum (0.297 anstatt 0.300). Die AUC erhöht sich mit `"extratrees"` für OOB- und Testdaten, allerdings auch in höherem Ausmaß für die Testdaten. Verwendet man, bei sonst gleich bleibenden Default-Einstellungen, `mtry=1` anstelle des Default-Werts 2, so verringert sich die Fehlklassifikationsrate von 0.300 auf 0.283 für die OOB-Daten und von 0.289 auf 0.259 für die OOB-Daten (vgl. Tabelle 5.2). Mit einer Verringerung von 10.7% gegenüber 5.7% ist die Auswirkung damit auch hier bei den Testdaten größer als bei den OOB-Daten.

Der Einfluss der minimalen Knotengröße unterscheidet sich besonders stark zwischen OOB- und Testdaten, aber auch danach, welcher Wert für `mtry` und welche Splitting-Regel verwendet wird. Bei Default-Einstellungen aller anderen Parameter kann durch eine Änderung der Knotengröße auf 46 eine Verringerung der OOB-Fehlklassifikationsrate um 7,7% (von 0.300 auf 0.277) und der Test-Fehlklassifikationsrate um 10% (von 0.289 auf 0.260) erreicht werden. Mit Knotengröße 96 erhält man eine OOB-Fehlklassifikationsrate von 0.288 und damit eine Verringerung von 4% sowie eine Test-Fehlklassifikationsrate von 0.233 und damit eine Verringerung um fast 20%.

Auch mit Subsampling kann für die Brustkrebsdaten eine erhebliche Verbesserung der Fehlklassifikationsrate gegenüber dem Default der Bootstrap-Stichprobe erreicht werden. Erstaunlicherweise erzielt man die größte Verbesserung mit ganz kleinen Stichproben, die aus nur 5% der Trainingsdaten bestehen. Für die OOB-Daten sinkt die Fehlklassifikationsrate damit von 0.301 auf 0.279 (vgl. Tabelle 5.2) und verringert sich damit um 7%. Die Test-Fehlklassifikationsrate ändert sich von 0.292 auf 0.251 und sinkt damit um etwa 14% (vgl. Tabelle 5.3). Die Zahlen beziehen sich wieder darauf, dass für alle übrigen Parameter die Default-Einstellungen verwendet werden.

5.3.2 Ergebnisse für GlaucomaM

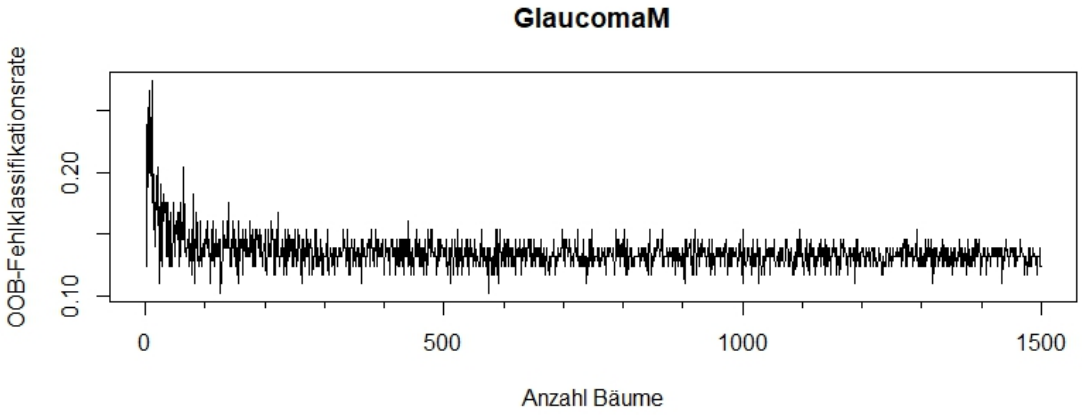


Abbildung 5.8: OOB-Fehlklassifikationsrate in Abhängigkeit von der Anzahl der Bäume (`num.trees`) für den Datensatz GlaucomaM

Die Fehlklassifikationsrate in Abhängigkeit von der Baumanzahl, dargestellt in Abbildung 5.8 für Random Forests mit $B = 1, ..., 1500$ Bäumen, sieht für den Glaukom-Datensatz ähnlich aus wie für die Brustkrebsdaten. Wegen den starken Schwankungen auch bei höherer Baumanzahl werden für die folgenden Analysen mit dem Glaukom-Datensatz die Gütemaße wieder über hundert Forests mit je 500 Bäumen gemittelt.

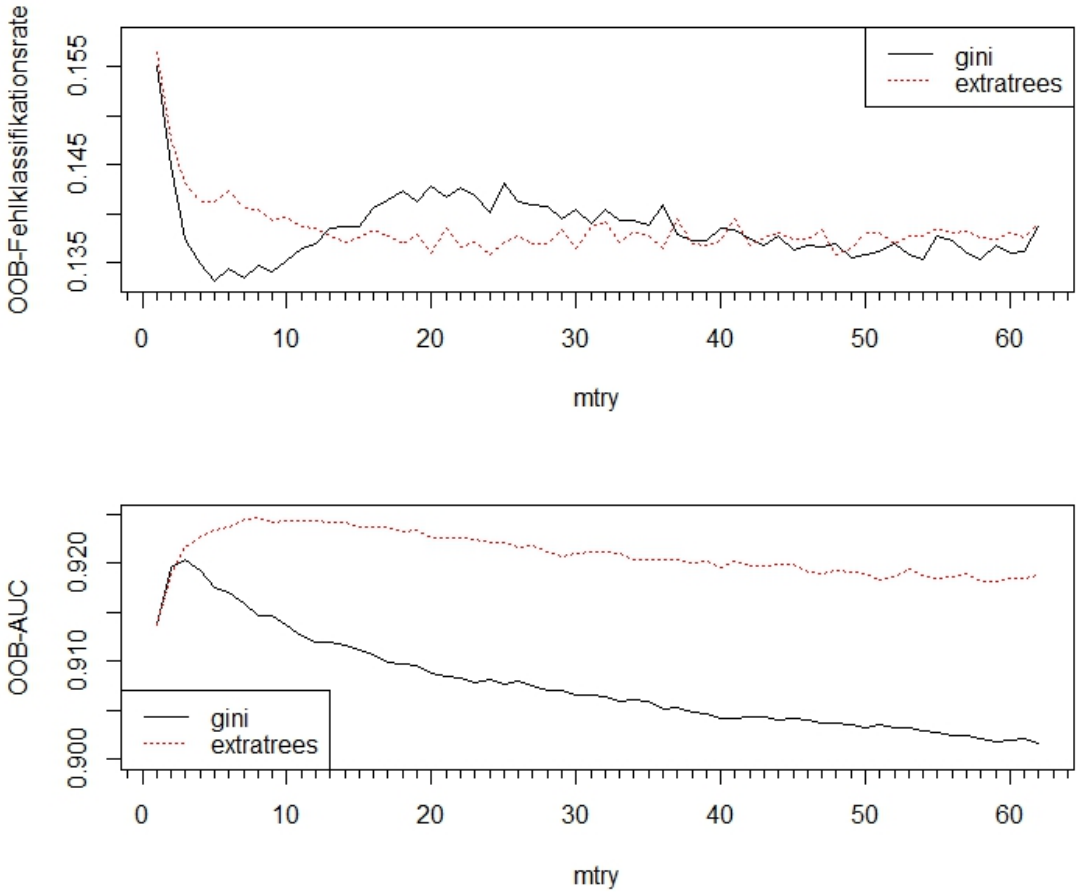


Abbildung 5.9: OOB-Fehlklassifikationsrate und OOB-AUC für den Glaukomdatensatz in Abhängigkeit von `mtry`

Der Parameter `mtry`. Zunächst soll wieder untersucht werden, wie die Wahl von `mtry` die Prädiktionsgüte der Random Forests beeinflusst, für beide bei Klassifikation zur Verfügung stehenden Splitting-Regeln. Die Anzahl der zufällig ausgewählten Splits scheint auch hier keinen nennenswerten Einfluss zu haben (vgl. Abbildungen 8.4 und 8.5 im Anhang) und wird deswegen auch für alle Analysen zu GlaucomaM auf 1 gesetzt. Da alle Kovariablen metrisch sind, spielt der Parameter respect.unordered.factors keine Rolle und muss nicht berücksichtigt werden.

Der Einfluss von `mtry` auf Fehlerrate und AUC für die OOB-Daten ist in Abbildung 5.9 dargestellt (Mittelwerte der Fehlerraten/AUCs von 100 Forests mit je 500 Bäumen). Der Default für p =62 Kovariablen liegt bei 7. Die gemittelte OOB-Fehlklassifikationsrate mit der Default-Splitting-Regel `"gini"` ist für Werte von 4-10 am niedrigsten, die gemittelte OOB-AUC ist für noch etwas kleinere Werte von ca. 2-5 am höchsten, aber auch für den Default 7 nur geringfügig niedriger. Die `"gini"`-

Fehlerrate für die Testdaten, vgl. Abbildung 5.10 oben, ist im Mittel für `mtry`-Werte zwischen 9 und 14 am niedrigsten, der `"gini"`-AUC ist für `mtry` zwischen 4 und 20 am höchsten und nimmt für steigende `mtry` nur langsam ab. Für die Splitting-Regel `"gini"` kann demnach für eine Änderung vom Default `mtry`=7 kaum eine Verbesserung der Prädiktionsgüte erzielt werden.

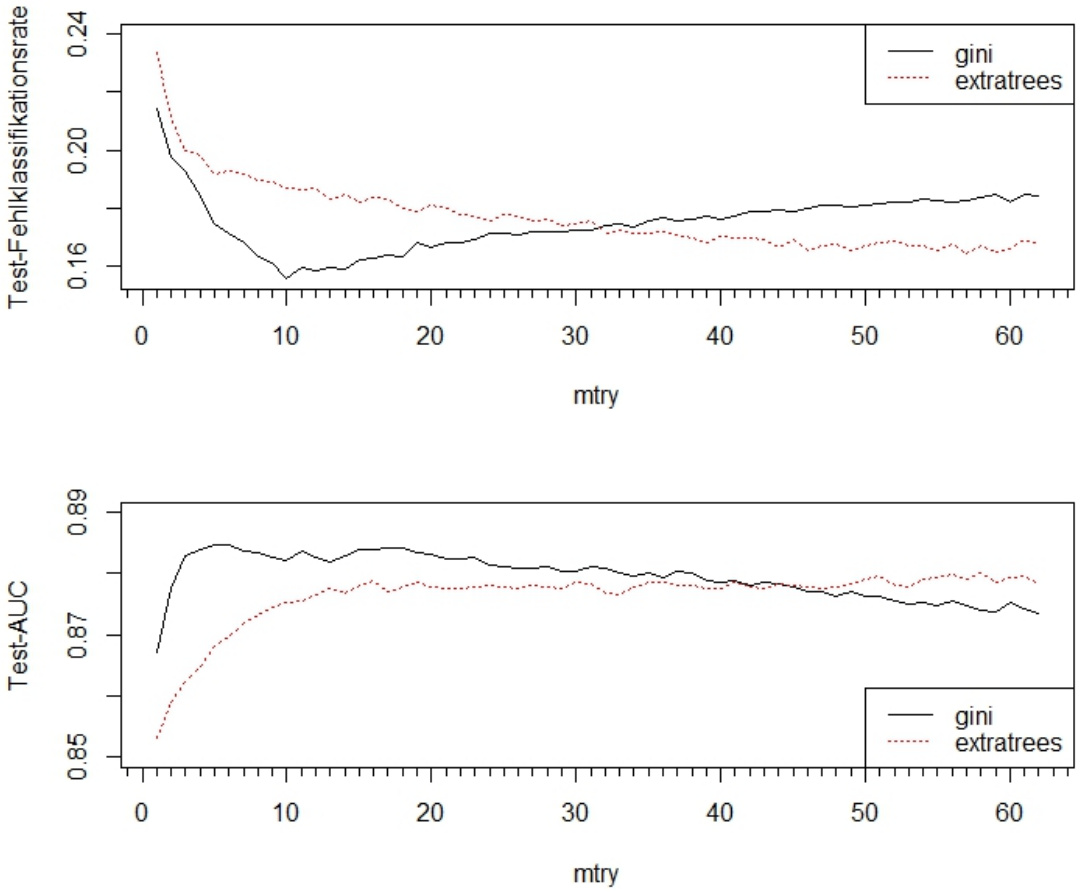


Abbildung 5.10: Test-Fehlklassifikationsrate und Test-AUC für den Glaukomdatensatz in Abhängigkeit von `mtry`

Mit der Splitting-Regel `"extratrees"` dagegen erzielt man tendenziell mit höheren `mtry`-Werten bessere Ergebnisse. Bei den OOB-Daten erreicht die gemittelte Fehlklassifikationsrate ab ca. `mtry`=15 ein für steigende `mtry` mit Schwankungen gleichbleibend niedrige Fehlerrate. Für die Testdaten nimmt der gemittelte Fehler mit steigendem `mtry` ab, wobei die Abnahme immer schwächer wird und ab ca. `mtry`=40 stagniert. Die niedrigsten `"extratrees"`-Fehlerraten sind sowohl für die OOB-Daten als auch für die Testdaten nur minimal höher als die niedrigsten `"gini"`-Fehlerraten. Die OOB-AUC ist mit `"extratrees"` deutlich höher als für

"gini", erreicht Höchstwerte für mtry=8 und nimmt dann mit steigendem mtry nur sehr schwach ab. Bei den Testdaten bleibt die "extratrees"-AUC ab ca. mtry=10 konstant hoch, höher als die "gini"-AUC ist sie allerdings nur für mtry¿45.

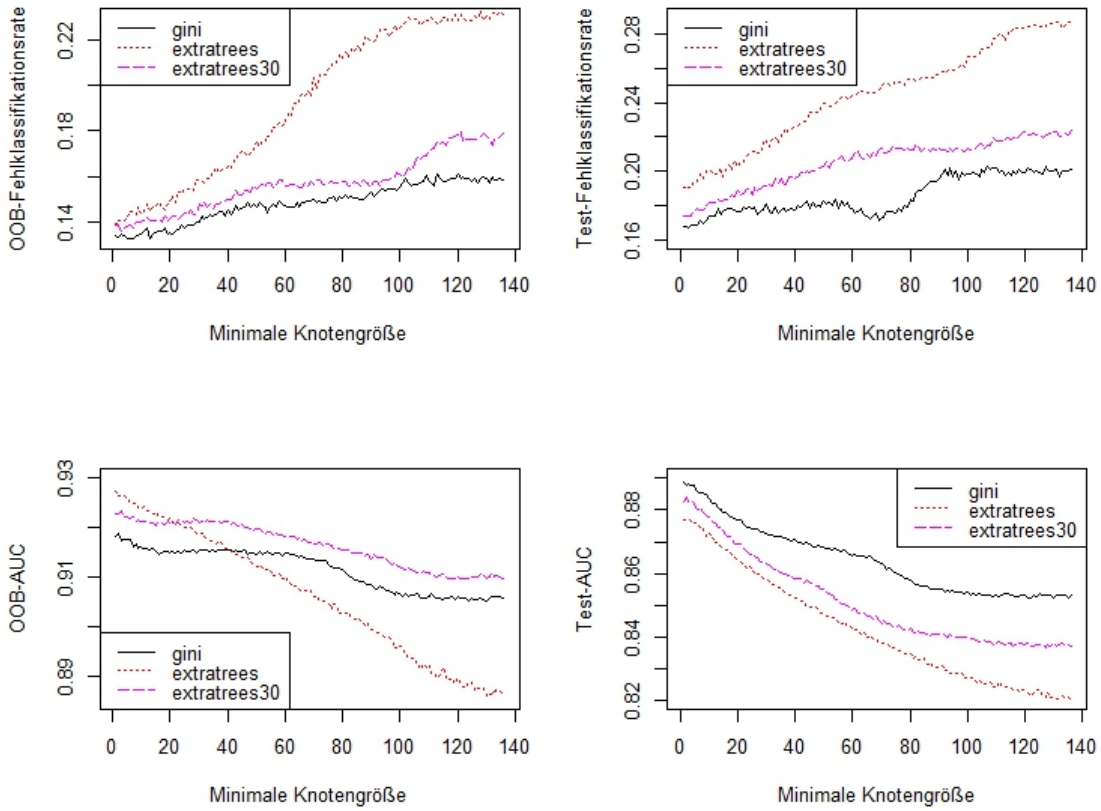


Abbildung 5.11: OOB- und Test-Fehlklassifikationsrate sowie OOB- und Test-AUC für den Glaukomdatensatz in Abhängigkeit von der minimalen Knotengröße

Minimale Knotengröße. Betrachtet man die Fehlerraten und AUCs in Abhängigkeit von der minimalen Knotengröße in Abbildung 5.11, so lässt sich sagen, dass die Prädiktionsgüte im Allgemeinen mit steigender Knotengröße abnimmt. Das gilt für beide Splitting-Regeln. Zwar erreicht man teilweise mit einer minimalen Knotengröße von 2 oder 3 eine niedrigere Fehlklassifikationsrate bzw. einen höheren AUC als mit Knotengröße 1, aber die Unterschiede sind sehr gering. Der Default-Wert 1 für min.node.size bei Klassifikation ist daher für diesen Datensatz eine gute Wahl. Der negative Einfluss großer Knotengrößen ist besonders stark für "extratrees" in Verbindung mit mtry=7.

Da sich gezeigt hat, dass "extratrees" in Verbindung mit höheren Werten für

`mtry` im Allgemeinen besser abschneidet als mit dem Default-Wert, ist in Abbildung 5.11 zusätzlich zu beiden Splitting-Regeln in Kombination mit dem Default `mtry`=7 `"extratrees"` auch noch in Kombination mit `mtry`=30 eingezeichnet. Auch hier zeigt sich, dass `"extratrees"` mit einem höheren Wert deutlich besser abschneidet als mit dem Default. Lediglich bei der OOB-AUC erzielt `"extratrees"` mit `mtry`=7 für Knotengrößen kleiner 20 einen höheren Wert. Für Knotengrößen bis 40 bewirkt `"extratrees"` mit `mtry`=7 auch einen im Mittel höheren AUC-Wert für die OOB-Daten als `"gini"`. Für `mtry`=30 und `"extratrees"` liegt die OOB-AUC durchgehend über der OOB-AUC mit `"gini"` und `mtry`=7. Bei den Fehlklassifikationsregeln für OOB- und Test-Daten sowie bei der Test-AUC schneidet hingegen immer `"gini"` besser ab als `"extratrees"`.

Stichprobengröße und subsampling vs. Bootstrap. Der Einfluss der Stichprobengröße auf Fehlklassifikationsrate und AUC ist in Abbildungen 5.12 und 5.13 jeweils für die OOB-Daten und die Test-Daten dargestellt. Da eine Anteilsgröße von 5% des Trainingdatensatzes durchwegs deutlich schlechtere Ergebnisse liefert als alle anderen Anteilsgrößen, ist sie nicht mit aufgeführt. Zum Vergleich ist zusätzlich zu den subsamples jeweils auch die Fehlerrate bzw. die AUC für Random Forests mit Bootstrap-Stichprobe in grün abgebildet. Fehlerrate und AUC sind für `"gini"` mit `mtry`=7 und für `"extratrees"` mit `mtry`=30 dargestellt.

Die den Boxplots in Abbildung 5.12 entsprechenden Mittelwerte sind in Tabelle 5.4 aufgeführt. Für die OOB-Daten erzielt man mit der Bootstrap-Stichprobe und Anteilsgrößen über 0.5 vergleichbar gute Ergebnisse. Für kleinere Anteilsgrößen ist die Fehlklassifikationsrate deutlich höher, die AUC minimal niedriger. Auch eine sehr große Anteilsgröße von 0.95 erzielt im Mittel gleich gute Ergebnisse wie die Bootstrap-Stichprobe, aber die Varianz ist hier deutlich höher.

Für die Testdaten erzielen sehr große Stichproben (0.85 und 0.95) ohne Zurücklegen im Mittel die niedrigsten Fehlerraten und höchsten AUCs. Je kleiner die Anteilsgröße, desto höher die Fehlerrate und desto höher die AUC. Die Bootstrap-Stichprobe kommt nur auf etwas schlechtere Werte. Mit der Default-Stichprobengröße für Ziehen ohne Zurücklegen, 0.632, erreicht man vergleichbar hohe Werte wie mit der Bootstrap-Stichprobe.

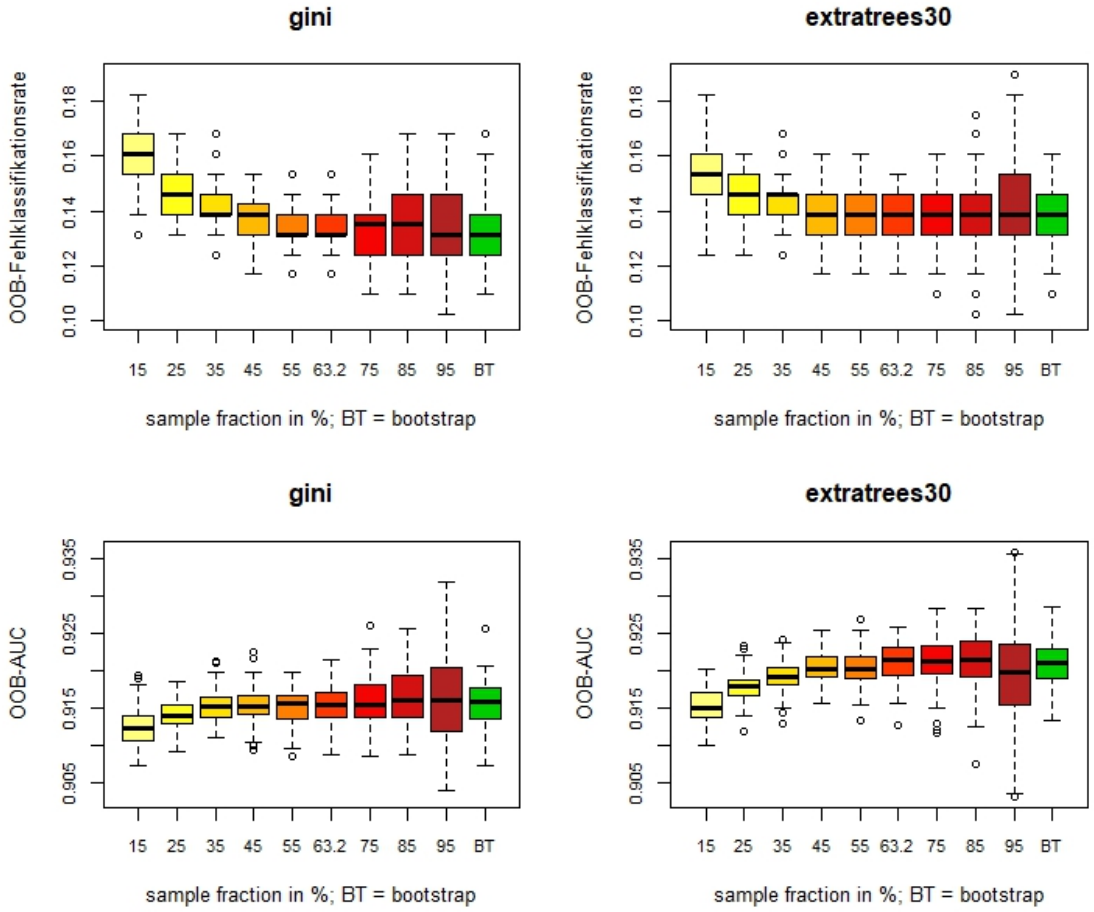


Abbildung 5.12: OOB-Fehlklassifikationsrate und OOB-AUC für den Glaukomdatensatz in Abhängigkeit von der Stichprobengröße. Links für Splitting-Regel `"gini"` mit Default `mtry=7`, rechts für Splitting-Regel `"extratrees"` mit `mtry=30`

Anteil	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	0.95	BT
gini FKR	0.159	0.146	0.142	0.136	0.134	0.133	0.134	0.135	0.134	0.133
ET FKR	0.155	0.144	0.143	0.140	0.138	0.137	0.137	0.138	0.140	0.137
gini AUC	0.912	0.914	0.915	0.915	0.915	0.915	0.916	0.916	0.916	0.916
ET AUC	0.915	0.918	0.919	0.920	0.920	0.921	0.921	0.921	0.920	0.921

Tabelle 5.4: Mittelwerte der OOB-Fehlklassifikationsraten und AUCs für den Glaukomdatensatz in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 5.12). Der niedrigste Wert (bzw. der höchste für die AUC) in jeder Zeile ist rot markiert.

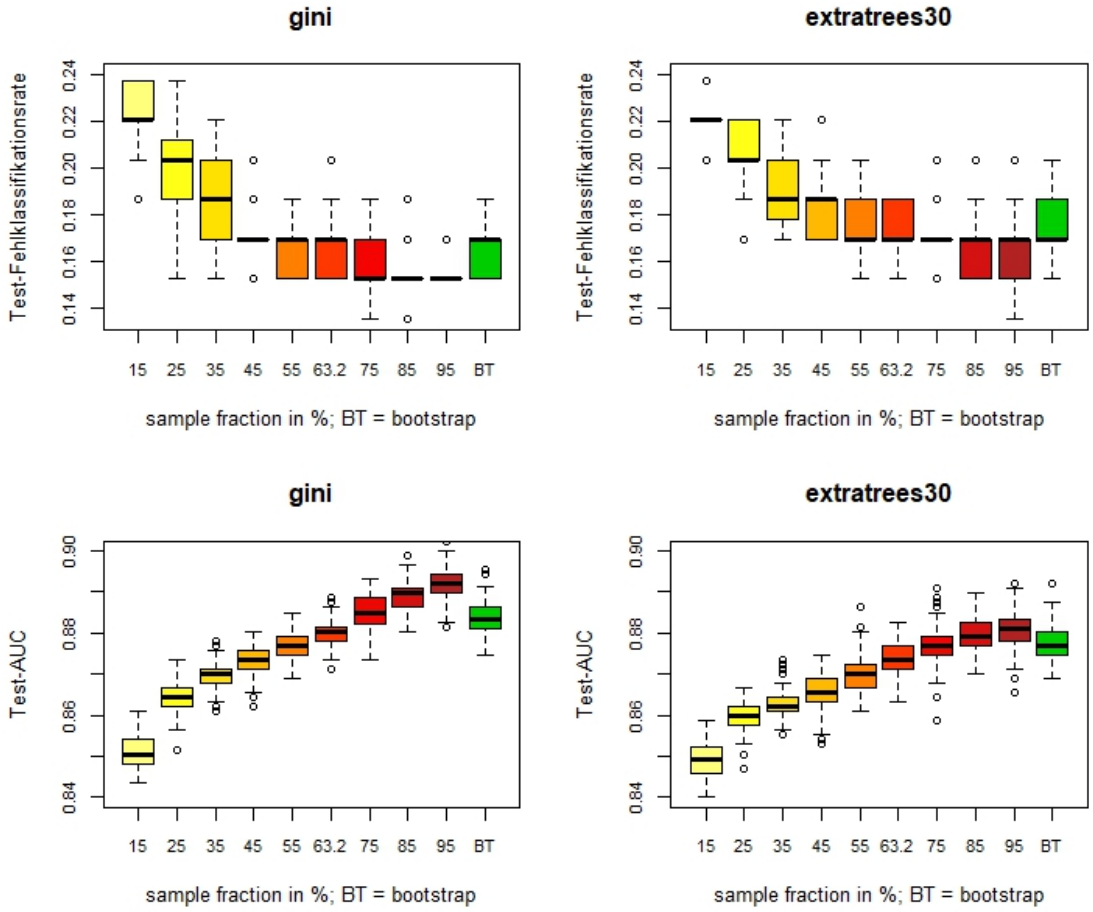


Abbildung 5.13: Test-Fehlklassifikationsrate und Test-AUC für den Glaukomdatensatz in Abhängigkeit von der Stichprobengröße. Links für Splitting-Regel `"gini"` mit Default `mtry=7`, rechts für Splitting-Regel `"extratrees"` mit `mtry=30`

Anteil	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	0.95	BT
gini FKR	0.223	0.201	0.187	0.173	0.166	0.163	0.158	0.155	0.155	0.166
ET FKR	0.222	0.206	0.191	0.181	0.175	0.172	0.170	0.167	0.168	0.174
gini AUC	0.851	0.864	0.870	0.874	0.877	0.880	0.885	0.889	0.892	0.884
ET AUC	0.849	0.859	0.863	0.866	0.870	0.874	0.877	0.880	0.881	0.878

Tabelle 5.5: Mittelwerte der Test-Fehlklassifikationsraten und AUCs für den Glaukomdatensatz in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 5.13). Der niedrigste Wert (bzw. der höchste für die AUC) in jeder Zeile ist rot markiert.

Zusammenfassung. Zusammenfassend lässt sich sagen, dass man mit einer Änderung der untersuchten Parameter bei den Glaukomdaten keine oder nur extrem kleine Verbesserungen der Prädiktionsgüte erreichen kann. Lediglich Subsampling mit großen Stichprobengrößen zwischen 0.85 und 0.95 bewirkt für die Testdaten

eine Verringerung der Fehlklassifikationsrate um 6.6%, nämlich von 0.164 auf 0.155 (vgl. Tabelle 5.5). Dieser Effekt zeigt sich aber nicht für die OOB-Daten.

Erwähnenswert ist noch, dass der Einfluss der Knotengröße und der Stichprobengröße für beide Splitting-Regeln sehr ähnlich ist, was für den Parameter `mtry` nicht gilt. Während `"gini"` nahezu die besten Ergebnisse für den `mtry`-Default 7 liefert, ist für `"extratrees"` ein deutlich höherer `mtry`-Wert besser. Eine niedrigere Fehlklassifikationsrate als mit der Default-Kombination `"gini"` und `mtry=7` lässt sich damit aber nicht erreichen.

5.3.3 Ergebnisse für die NHANES-Daten

Für die folgenden Analysen mit dem NHANES-Datensatz wird der MSE jeweils gemitttelt über 20 Forests mit je 400 Bäumen. Wie Abbildung 5.14 zeigt, nimmt der OOB-MSE schon ab ungefähr hundert Bäumen im Mittel kaum noch ab. Allerdings werden die Schwankungen noch etwas geringer mit steigender Baumzahl, ab ca. 400 Bäumen ist aber kaum noch eine Abnahme der Schwankung zu erkennen.

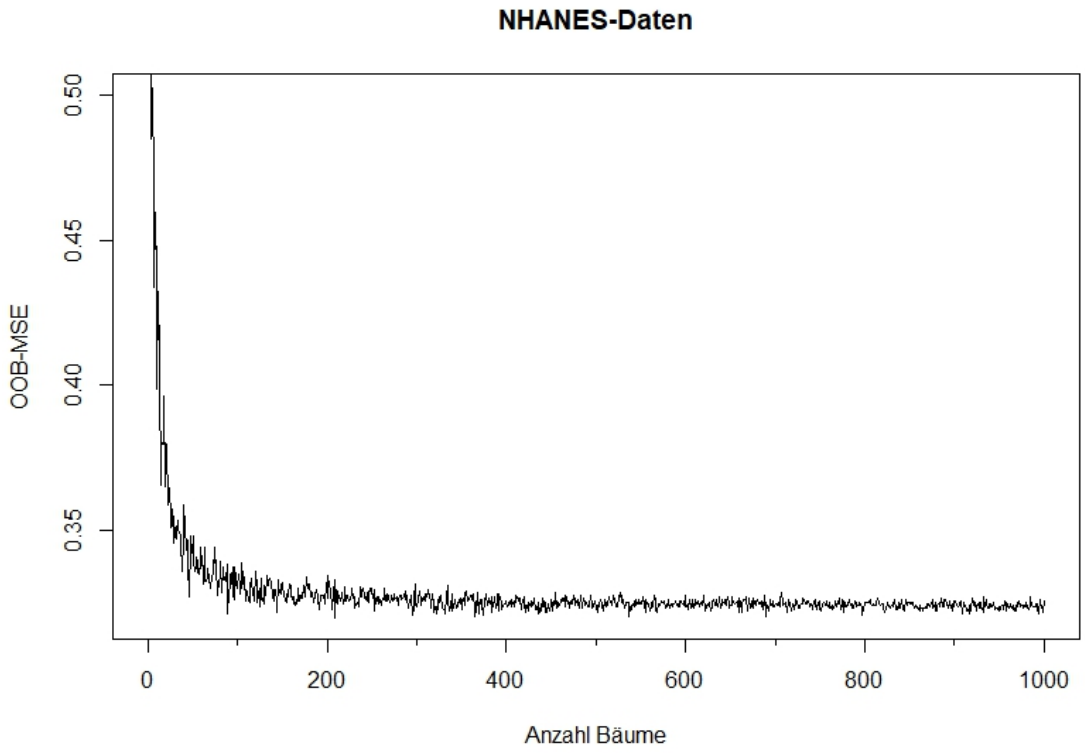


Abbildung 5.14: OOB-MSE in Abhängigkeit von der Anzahl der Bäume (`num.trees`) für den NHANES-Datensatz

Da einige der kategorialen Kovariablen im NHANES-Datensatz nominal skaliert

sind, wird der Parameter `respect.unordered.factors` für die folgenden Analysen für die Splitting-Regeln `"variance"` und `"extratrees"` auf `"order"` gesetzt. Für `"maxstat"` erzeugt `"order"` Warnmeldungen, deswegen bleibt der Parameter `respect.unordered.factors` für diese Splitting-Regel auf die Default-Einstellung `"ignore"` gesetzt.

Für die Splitting-Regel `"extratrees"` ist der Parameter `num.random.splits` bei den folgenden Analysen stets auf 2 gesetzt, da sich gezeigt hat, dass der MSE für `num.random.splits=1` im Mittel höher ist als für größere Werte. Für Werte > 1 hingegen gibt es keine großen Unterschiede mehr. (vgl. Abbildung 8.6 im Anhang) Für die Splitting-Regel `"maxstat"` wird bei dem Parameter `alpha` im Folgenden die Default-Einstellung 0.5 beibehalten, da für größere Werte der MSE gar nicht oder nur geringfügig niedriger ist und für kleinere Werte der MSE höher ist. (vgl. Abbildung 8.7 im Anhang) Der Parameter `minprop` wird im Folgenden immer auf 0 gesetzt, da der MSE mit steigenden Werten für `minprop` auch steigt. (vgl. Abbildung 8.8 im Anhang)

Der Parameter `mtry`. Der Einfluss von `mtry` auf den MSE ist in Abbildungen 5.15 und 5.16 jeweils für die OOB-Daten und die Testdaten für alle drei Splitting-Regeln dargestellt. Wie man sieht, ist der Default `mtry=5` für `"variance"` eine gute Wahl, während bei `"maxstat"` und `"extratrees"` mit etwas größeren Werten für `mtry` ein im Mittel etwas niedrigerer MSE erzielt werden kann. Sehr kleine Werte für `mtry` erhöhen für alle drei Splitting-Regeln im Mittel den MSE, am stärksten für `"maxstat"` und am schwächsten für `"variance"`. Sehr hohe Werte für `mtry` bewirken vor allem für `"variance"` einen höheren gemittelten MSE, `"maxstat"` und `"extratrees"` erreichen für die OOB-Daten ab `mtry=14` bzw. `mtry=9` und für die Testdaten ab `mtry=8` bzw. `mtry=13` einen im Mittel niedrigeren Fehler.

Vergleicht man die drei Splitting-Regeln miteinander, so gibt es Unterschiede zwischen OOB-MSE und TEST-MSE. Für die OOB-Daten erreicht `"extratrees"` ähnlich niedrige Fehler wie `"variance"`, während `"maxstat"` durchgehend einen höheren gemittelten MSE bewirkt als `"extratrees"`, für die Testdaten ist es genau umgekehrt. Das ist auch in Tabelle 5.6 zu sehen. Hier ist für jede Splitting-Regel der niedrigste MSE aufgelistet sowie der zugehörige `mtry`-Wert. Dabei ist jeweils der minimale OOB-MSE, der minimale Test-MSE sowie der niedrigste Mittelwert aus OOB- und Test-MSE dargestellt. Berücksichtigt man also OOB- und Testdaten, so schneidet `"variance"` insgesamt etwas besser ab als die beiden anderen Splitting-Regeln.

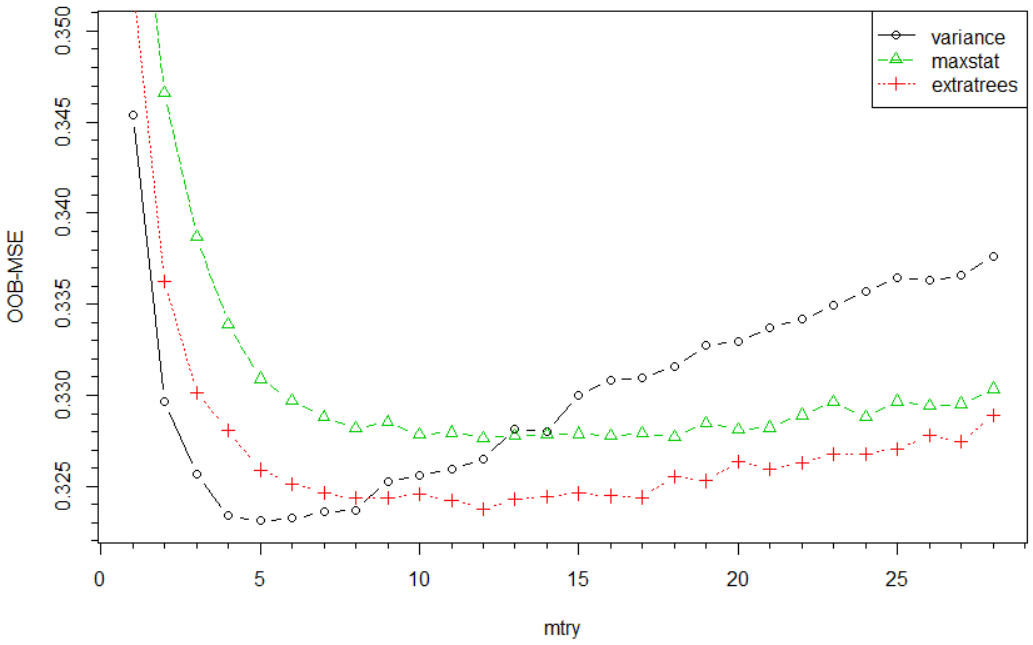


Abbildung 5.15: OOB-MSE in Abhängigkeit von `mtry` für die NHANES-Daten

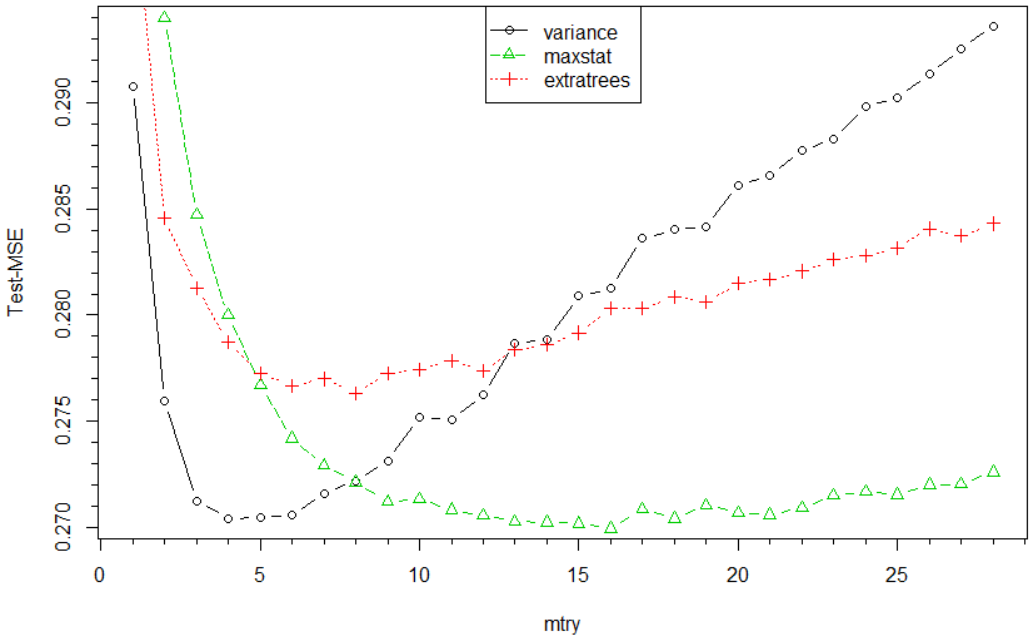


Abbildung 5.16: Test-MSE in Abhängigkeit von `mtry` für die NHANES-Daten

		minimaler MSE	`mtry`
OOB-Daten	`variance`	0.323	5
	`maxstat`	0.328	12
	`extratrees`	0.324	12
Testdaten	`variance`	0.270	4
	`maxstat`	0.270	16
	`extratrees`	0.276	8
Mittel OOB/Test	`variance`	0.297	5
	`maxstat`	0.299	16
	`extratrees`	0.300	8

Tabelle 5.6: Minimaler MSE und zugehöriger Wert für `mtry` für alle drei Splitting-Regeln, für OOB-Daten, Testdaten, und Mittel aus OOB- und Testdaten (vgl. Abbildungen 5.15 und 5.16).

Minimale Knotengröße. Abbildung 5.17 zeigt den MSE gemittelt über 20 Forests mit je 400 Bäumen in Abhängigkeit von der minimalen Knotengröße, oben für die OOB-Daten und unten für die Testdaten. Der MSE wurde für Knotengröße 1 und 5,10,15,...,1355 berechnet, für alle drei Splitting-Regeln mit `mtry=5`. Zusätzlich ist der gemittelte MSE für die Kombination `"maxstat"` und `mtry=16` sowie `"extratrees"` und `mtry=8` eingezeichnet, weil das die `mtry`-Werte sind, mit denen diese Splitting-Regeln den kleinsten gemittelten MSE erzielt haben (vgl. Tabelle 5.6). In beiden Abbildungen ist deutlich zu erkennen, dass sowohl `"maxstat"` als auch `"extratrees"` mit den höheren `mtry`-Werten für alle Knotengrößen einen kleineren MSE erreicht als mit `mtry=5`.

Fast alle dargestellten Varianten weisen einen ähnlichen Verlauf auf: Zuerst sinkt der MSE etwas mit steigender Knotengröße, und ab einer bestimmten Knotengröße steigt der MSE mit steigender Knotengröße. Das trifft gleichermaßen für die OOB-Daten und für die Testdaten zu. Lediglich bei der Kombination `"maxstat"` mit `mtry=5` ist der MSE im Mittel für die Testdaten bei Knotengröße 1 am kleinsten und steigt dann kontinuierlich. Die Knotengrößen, für die die verschiedenen `splitrule/mtry`-Kombinationen den kleinsten MSE haben, sowie die kleinsten MSE-Werte selbst, sind in Tabelle 5.7 aufgeführt. Die Werte werden wieder für den OOB-MSE, den Test-MSE und die Mittelung aus OOB- und Test-MSE gezeigt.

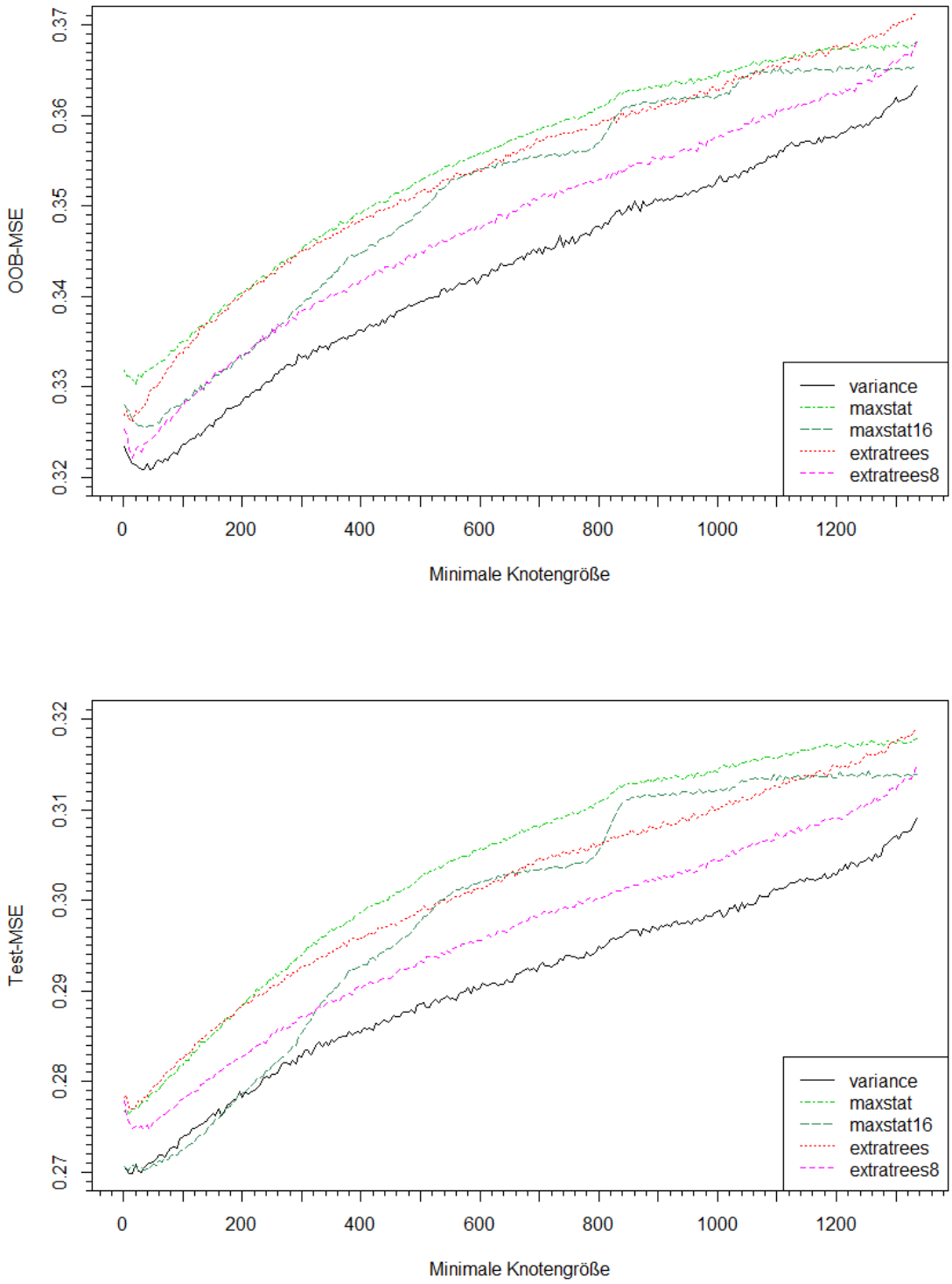


Abbildung 5.17: MSE in Abhängigkeit von der minimalen Knotengröße für die NHANES-Daten. Oben: für die OOB-Daten, unten: für die Testdaten.

Wie man in Tabelle 5.7 sehen kann, erzielen für fast alle dargestellten Kombinationen Knotengrößen, die größer sind als der Default 5, den im Mittel kleinsten Fehler (Ausnahme ist wie bereits oben erwähnt `"maxstat"` mit `mtry=5`). Die optimalen Knotengrößen unterscheiden sich je nach Splitting-Regel/`mtry`-Kombination und auch danach, ob OOB-MSE oder Test-MSE betrachtet werden, liegen aber alle im Bereich zwischen 1 und 40. Zum Vergleich ist in der Tebelle jeweils noch der MSE für den Knotengröße-Default 5 mit aufgeführt. Wie man sieht, ist der MSE im Mittel für den Default-Wert kaum schlechter als der kleinste gemittelte MSE. In den meisten Fällen ist der Unterschied so gering, dass er beim Runden auf drei Nachkommastellen verschwindet. Das heißt, durch Änderung des Knotengrößen-Parameters lässt sich keine nennenswerte Verbesserung der Prädiktionsgüte erreichen.

		min. MSE (m.n.s)	MSE für m.n.s=5
OOB-Daten	`variance`	0.321 (35)	0.321
	`maxstat`	0.331 (20)	0.331
	`maxstat16`	0.325 (40)	0.326
	`extratrees`	0.326 (20)	0.326
	`extratrees8`	0.323 (15)	0.323
Testdaten	`variance`	0.269 (15)	0.270
	`maxstat`	0.276 (1)	0.277
	`maxstat16`	0.270 (10)	0.270
	`extratrees`	0.277 (15)	0.277
	`extratrees8`	0.275 (40)	0.275
Mittel OOB/Test	`variance`	0.295 (35)	0.296
	`maxstat`	0.304 (10)	0.304
	`maxstat16`	0.298 (40)	0.298
	`extratrees`	0.302 (15)	0.302
	`extratrees8`	0.299 (15)	0.299

Tabelle 5.7: Minimaler MSE mit zugehöriger minimale Knotengröße (m.n.s) sowie zum Vergleich der MSE für den Default `min.node.size=5` (vgl. Abbildung 5.17). Für alle drei Splitting-Regeln mit Default `mtry=5` sowie für `"maxstat"` mit `mtry=16` und `"extratrees"` mit `mtry=8` jeweils für OOB-Daten, Testdaten, und Mittel aus OOB- und Testdaten

Stichprobengröße und subsampling vs. Bootstrap. In Abbildung 5.18 sieht man links den OOB-MSE und rechts den Test-MSE für verschiedene Stichprobengrößen (gezogen ohne Zurücklegen) sowie für eine Bootstrap-Stichprobe. Die Stichprobengrößen sind wieder als prozentualer Anteil des Trainingdatensatzumfangs $N = 1340$ angezeigt. Auf Anteile unter 15% ist in der Darstellung verzichtet worden, da sie fast immer den höchsten MSE hatten. OOB- und Test-MSE wird jeweils für alle

drei Splitting-Regeln dargestellt, wobei der Parameter für `mtry` für `"variance"` auf 5 gesetzt ist, für `"maxstat"` auf 16 und für `"extratrees"` auf 8. Jeder Boxplot in der Abbildung repräsentiert 20 Random Forests aus je 400 Bäumen. Die Mittelwerte zu den Boxplots aus Abbildung 5.18 sind in den Tabellen 5.8 und 5.9 jeweils für OOB- und Testdaten aufgeführt.

Abbildung 5.18 zeigt, dass die Stichprobengröße auf die beiden Splitting-Regeln `"variance"` und `"extratrees"` einen ähnlichen Effekt hat, der sich aber zwischen OOB-Daten und Testdaten unterscheidet. Bei den OOB-Daten erzielen für beide Splitting-Regeln kleinere Stichprobengrößen den im Mittel niedrigsten MSE. Die Bootstrap-Stichprobe sowie die Default-Größe von 63.2% erreichen aber fast ebenso niedrige Werte. Insgesamt sind die MSE-Werte für die beiden Splitting-Regeln fast gleich. Bei den Testdaten hingegen erreichen sowohl für `"variance"` als auch für `"extratrees"` größere Stichprobengrößen einen etwas niedrigeren MSE. Die Bootstrap-Stichprobe schneidet geringfügig schlechter ab. Hier erzielt `"variance"` durchgehend einen etwas niedrigeren MSE als `"extratrees"`. Insgesamt unterscheidet sich der MSE aber für diese beiden Splitting-Regeln nur sehr wenig zwischen den unterschiedlichen Stichprobengrößen. Lediglich ein Anteil von 95% liefert für die OOB-Daten bei beiden Splitting-Regeln einen deutlich höheren MSE.

Die Splitting-Regel `"maxstat"` hingegen zeigt für OOB-Daten und Test-Daten das gleiche Bild: Der MSE sinkt mit steigender Anteilsgröße und der niedrigste MSE wird mit der Bootstrap-Stichprobe erreicht. Bei den Testdaten erzielt `"maxstat"` mit der Bootstrap-Stichprobe einen ähnlich niedrigen MSE wie `"variance"` mit subsampling und schneidet damit auch besser ab als `"extratrees"`. Bei den OOB-Daten hingegen schneidet `"maxstat"` auch mit der Bootstrap-Stichprobe schlechter ab als die beiden anderen Splitting-Regeln sowohl mit subsampling als auch mit Bootstrap.

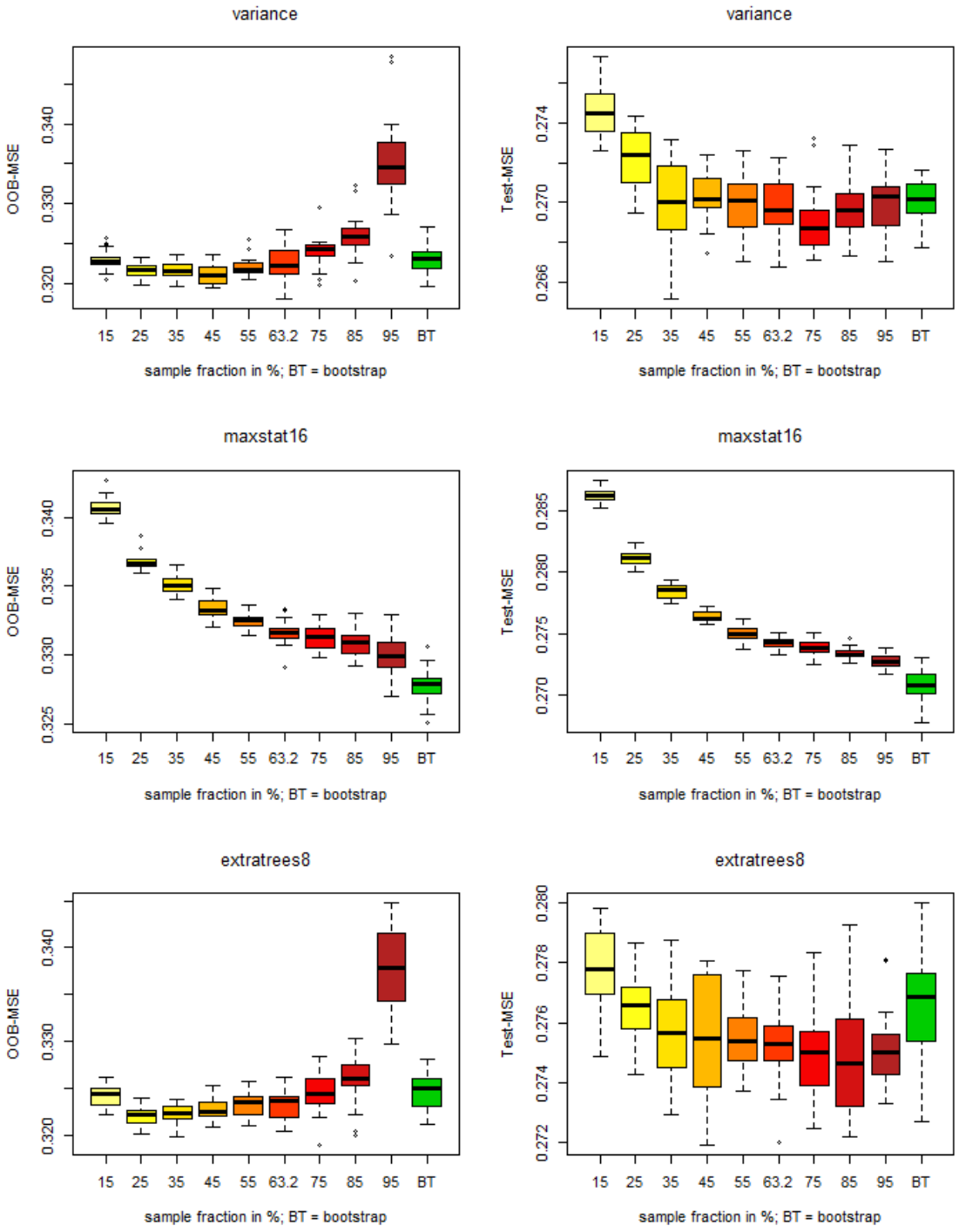


Abbildung 5.18: MSE in Abhängigkeit von der Stichprobengröße für die NHANES-Daten. Links für die OOB-Daten, rechts für die Testdaten. Oben für die Splitting-Regel `"variance"`, in der Mitte für `"maxstat"` (mit `mtry=16`), unten für `"extratrees"` (mit `mtry=8`).

Anteil	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	0.95	BT
`variance`	0.323	0.322	0.322	0.321	0.322	0.323	0.324	0.326	0.335	0.323
`maxstat16`	0.341	0.337	0.335	0.333	0.332	0.332	0.331	0.331	0.330	0.328
`extratrees8`	0.324	0.322	0.322	0.323	0.323	0.323	0.325	0.326	0.338	0.325

Tabelle 5.8: Mittelwerte der OOB-MSEs in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 5.18 links). Der niedrigste Wert in jeder Zeile ist rot markiert.

Anteil	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	0.95	BT
`variance`	0.275	0.272	0.270	0.270	0.270	0.270	0.269	0.270	0.270	0.274
`maxstat16`	0.286	0.281	0.278	0.276	0.275	0.274	0.274	0.273	0.273	0.271
`extratrees8`	0.278	0.276	0.276	0.276	0.275	0.275	0.275	0.275	0.275	0.277

Tabelle 5.9: Mittelwerte der Test-MSEs für die NHANES-Daten in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 5.18 rechts). Der niedrigste Wert in jeder Zeile ist rot markiert.

Zusammenfassung. Die Ergebnisse für die NHANES-Daten ähneln denen für die Glaukomdaten. Auch hier lässt sich durch eine Veränderung der Parametereinstellungen keine oder nur eine vernachlässigbar kleine Verbesserung der Prädiktionsgüte erreichen. Wie bei den Glaukomdaten unterscheidet sich der Einfluss von `mtry` je nach Splitting-Regel etwas, während der Einfluss der Knotengröße für alle Splitting-Regeln sehr ähnlich ist. Die beiden Splitting-Regeln `"variance"` und `"extratrees"` reagieren auch auf eine Änderung der Stichprobengröße sehr ähnlich, `"maxstat"` hingegen unterscheidet sich in dieser Hinsicht von den anderen beiden Splitting-Regeln.

5.4 Variablenwichtigkeit

Im Folgenden soll untersucht werden, wie sich unterschiedliche Einstellungen der Parameter `mtry`, `splitrule`, `min.node.size`, `replace` und `sample.fraction` auf die Variablenwichtigkeit auswirken. Die Variablenwichtigkeitswerte werden dafür wie die Maßzahlen zur Beurteilung der Prädiktionsgüte für den Brustkrebs- und den Glaukomdatensatz über 100 Forests mit je 500 gemittelt, und für den NHANES-Datensatz über 20 Forests mit je 400 Bäumen gemittelt. Abgesehen von den Parametern, deren unterschiedliche Einstellungen jeweils betrachtet werden, werden immer die Default-Einstellungen verwendet.

5.4.1 Ergebnisse für die Brustkrebsdaten

Splitting-Regel. Zunächst soll der Einfluss der Splitting-Regel auf die Variablenwichtigkeit untersucht werden, für beide in `ranger` implementierten Möglichkeiten, die Variablenwichtigkeit zu messen. In Abbildung 5.19 sind die Variablen mit ihren jeweiligen *importance*-Werten der Größe nach absteigend geordnet dargestellt, oben gemessen mit der Permutationsmethode (pink), unten mit der Unreinheitsmethode (blau), links für `splitrule="gini"` und rechts für `splitrule="extratrees"`. Es fällt gleich auf, dass die Rangordnung der Variablen sehr unterschiedlich ist je nachdem, welches Wichtigkeitsmaß verwendet wird. Die Variable *age* hat mit `importance="impurity"` einen negativen Wert und damit überhaupt keinen Einfluss auf den Response, während sie mit `importance="permutation"` die einflussreichste Kovariable ist. Auch die restliche Rangordnung unterscheidet sich stark nach der Messart. Für die beiden unterschiedlichen Splitting-Regeln besteht dagegen für `"permutation"` gar kein Unterschied in der Rangordnung, während für `"impurity"` zumindest der erste, dritte und letzte Rang gleich bleiben.

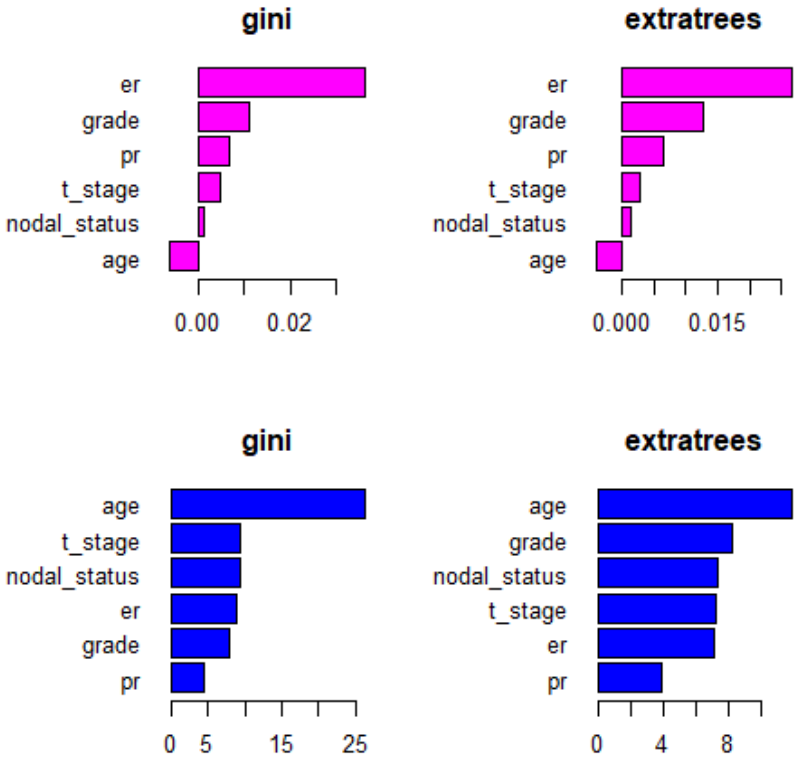


Abbildung 5.19: Variablenwichtigkeit der Brustkrebsdaten für die beiden unterschiedlichen Splitting-Regeln. In pink für `importance="permutation"`, in blau für `importance="impurity"`

Der Parameter `mtry`. Der Einfluss des Parameters `mtry` auf die Variablenwichtigkeit wird in Abbildung 5.20 jeweils für die Permutationsmethode (in pink oben) und die Unreinheitsmethode (in blau unten) dargestellt. Wie man sieht, ändert sich bei beiden Methoden die Rangfolge der Variablen kaum für unterschiedliche `mtry`. Mit `"permutation"` tauschen nur beim Wechsel von `mtry=2` zu `mtry=3` die Variablen *pr* und *t_stage* die Ränge. Mit `"impurity"` findet nur beim Wechsel von `mtry=1` zu `mtry=2` eine Änderung der Rangordnung statt: *er* und *grade* rutschen beide zwei Ränge nach hinten, *t_stage* und *nodal_status* dafür zwei Ränge nach vorne. Wie bereits beim Vergleich der Splitting-Regeln fällt auch in Abbildung 5.20 wieder auf, dass sich die Rangordnungen der Variablenwichtigkeiten sehr stark danach unterscheiden, welche Messart verwendet wird.

Minimale Knotengröße. In Abbildung 5.21 sind zwei *heatmaps* zu sehen, die die Rangkorrelationen nach Spearman zwischen den Variablenwichtigkeiten (Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`) bei verschiedenen minimalen Knotengrößen zeigen. Die jeweiligen Knotengrößen sind an den Rändern der *heatmaps* aufgelistet. Jede *heatmap* entspricht damit einer Korrelationstabelle, in der die einzelnen Korrelationen mit unterschiedlichen Farbabstufungen dargestellt werden. Dementsprechend ist die *heatmap* symmetrisch entlang der Diagonale von oben links nach unten rechts. Der *color key* jeweils oben links zeigt, welcher Farbton für welchen Korrelationswert steht. Er ist für diese und die folgenden Abbildungen (außer für die zu den NHANES-Daten) innerhalb einer Abbildung jeweils so angepasst, dass eine Farbstufe für beide *heatmaps* in etwa dem selben Korrelationswert entspricht. Die Knotengrößen in der Abbildungen gehen in 10er-Schritten von 10 bis 200, und zusätzlich sind noch die Knotengröße 1, 3 und 5 dabei.

Wie man an den roten Blöcken in der oberen `permutation`-*heatmap* sieht, bestehen zum Teil gleiche Rangordnungen für unterschiedliche Knotengrößen. Zum Beispiel ergeben Knotengrößen von 40 bis 100 die gleiche Rangordnung für die Variablenwichtigkeit. Die schwächste Korrelation und damit die größten Unterschiede zwischen den Rangordnungen gibt es zum Beispiel zwischen Knotengrößen 1,10,20 und Knotengröße 40-100. Die genauen Rangordnungen für Knotengröße 1, 80 und 200 sind in Abbildung 8.9 im Anhang abgebildet. Es wäre zu erwarten, dass die Korrelationen umso größer sind, umso näher die Knotengrößen beieinander liegen, aber das ist hier nicht der Fall. Zum Beispiel ist die Knotengröße 1 mit Knotengrößen von 170 bis 200 höher korreliert als mit Knotengrößen zwischen 40 und 100.

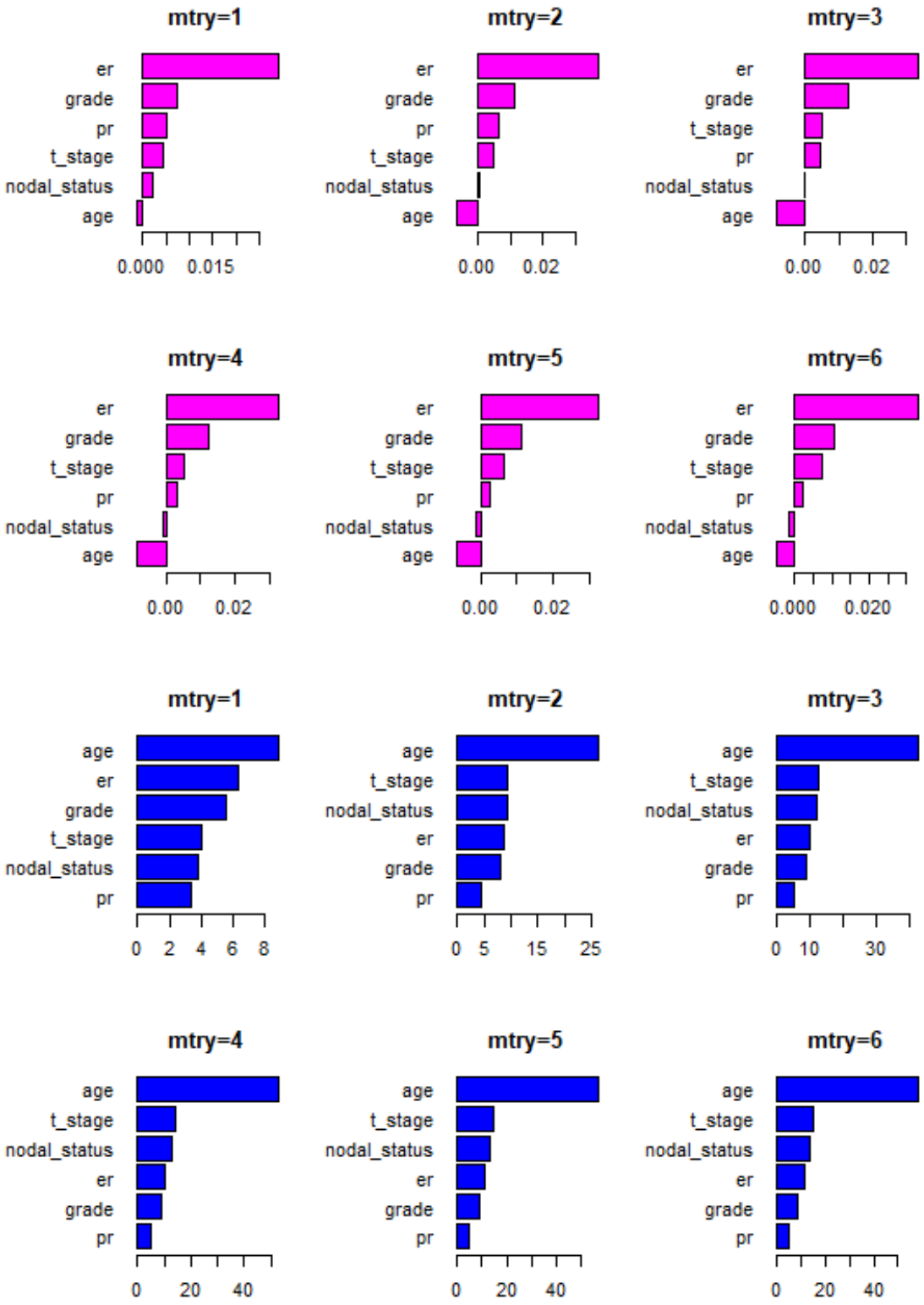


Abbildung 5.20: Die Variablenwichtigkeit in Abhängigkeit von `mtry` für die Brustkrebsdaten. Oben (in pink): gemessen mit `"permutation"`, unten (in blau): gemessen mit `"impurity"`.

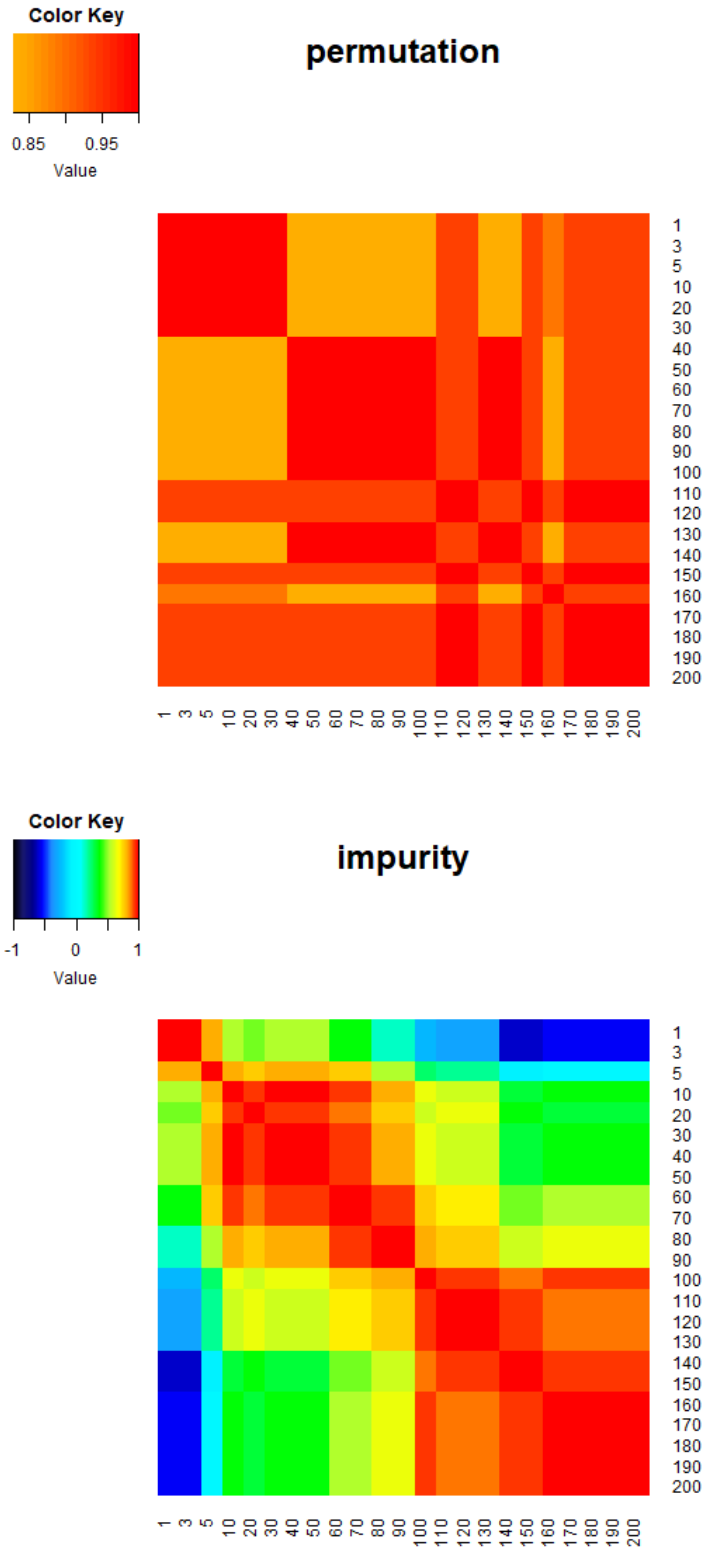


Abbildung 5.21: Spearman-Korrelationen zwischen den Variablenwichtigkeiten der Brustkrebsdaten für unterschiedliche minimale Knotengrößen. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`

Wie man in der unteren `impurity`-*heatmap* sieht, sind die Korrelationen hier zum Teil sehr niedrig oder sogar negativ. Das heißt, die Variabilität der Variablenwichtigkeit ist für die Brustkrebsdaten abhängig von der minimalen Knotengröße für die Messart `"impurity"` sehr groß. Vor allem die Variablenwichtigkeiten mit Knotengröße 1 unterscheiden sich sehr stark von denen mit großen Knotengrößen. Das wird auch in Abbildung 8.10 im Anhang deutlich, die die genauen Rangordnungen für Knotengröße 1, 20, 80, 100, 150 und 200 zeigt. Auch die Rangordnungen für die Knotengrößen 10 bis 70 sind nur schwach mit denen für Knotengrößen ≥ 140 korreliert. Größere Korrelationen finden sich hingegen innerhalb der Gruppe von Knotengrößen 10 bis 70 und innerhalb der Gruppe 100 bis 200.

Stichprobengröße und subsampling vs. Bootstrap. Analog zu den *heatmaps* für die Knotengrößen zeigt Abbildung 5.22 die Rangkorrelationen zwischen den Variablenwichtigkeiten für unterschiedliche Stichprobengrößen, jeweils gemessen mit `"permutation"` (oben) und `"impurity"` (unten). Die Stichprobengrößen gehen von 0.05 (entspricht 5% des Trainingsdatensatzumfangs) in 0.05-Schritten bis 1, wobei jeweils ohne Zurücklegen gezogen wurde, außer für Stichprobengröße 1 (entspricht damit der Bootstrap-Stichprobe). Anstelle von 0.65 ist wie schon bei den Analysen zur Prädiktionsgüte 0.632 gesetzt. Vergleicht man die beiden Abbildungen, fällt gleich auf, dass die Variabilität der Rangordnungen für die Unreinheitsmethode wieder größer ist als bei der Permutationsmethode. Der Unterschied zwischen den beiden Messarten ist aber hier viel geringer als bei den Knotengrößen.

Für `importance="permutation"` unterscheiden sich die Rangordnungen der Variablen zwischen Stichprobengrößen 0.05 bis 0.7 und der Bootstrap-Stichprobe gar nicht. Die schwächste Korrelation findet man zwischen den Wichtigkeiten für die Stichprobengrößen 0.9 bis 0.95 und den Wichtigkeiten für die Größen 0.05 bis 0.7, sowie zwischen 0.9 bis 0.95 und der Bootstrap-Stichprobe.

Für `importance="impurity"` ist die schwächste Korrelation zwischen den Variablenwichtigkeiten mit Stichprobengröße unter 0.5 und denen für Stichprobengrößen von 0.75 bis 0.95. Mit der Bootstrap-Stichprobe sind die kleinen Stichprobengrößen dagegen deutlich höher korreliert. Die Bootstrap-Stichprobe korreliert wieder am schwächsten mit den größten Stichprobengrößen ohne Zurücklegen (0.75 bis 0.95), gar kein Unterschied bei der Rangordnung der Variablen besteht hingegen zwischen der Bootstrap-Stichprobe und Stichprobengrößen von 0.5 bis 0.6.

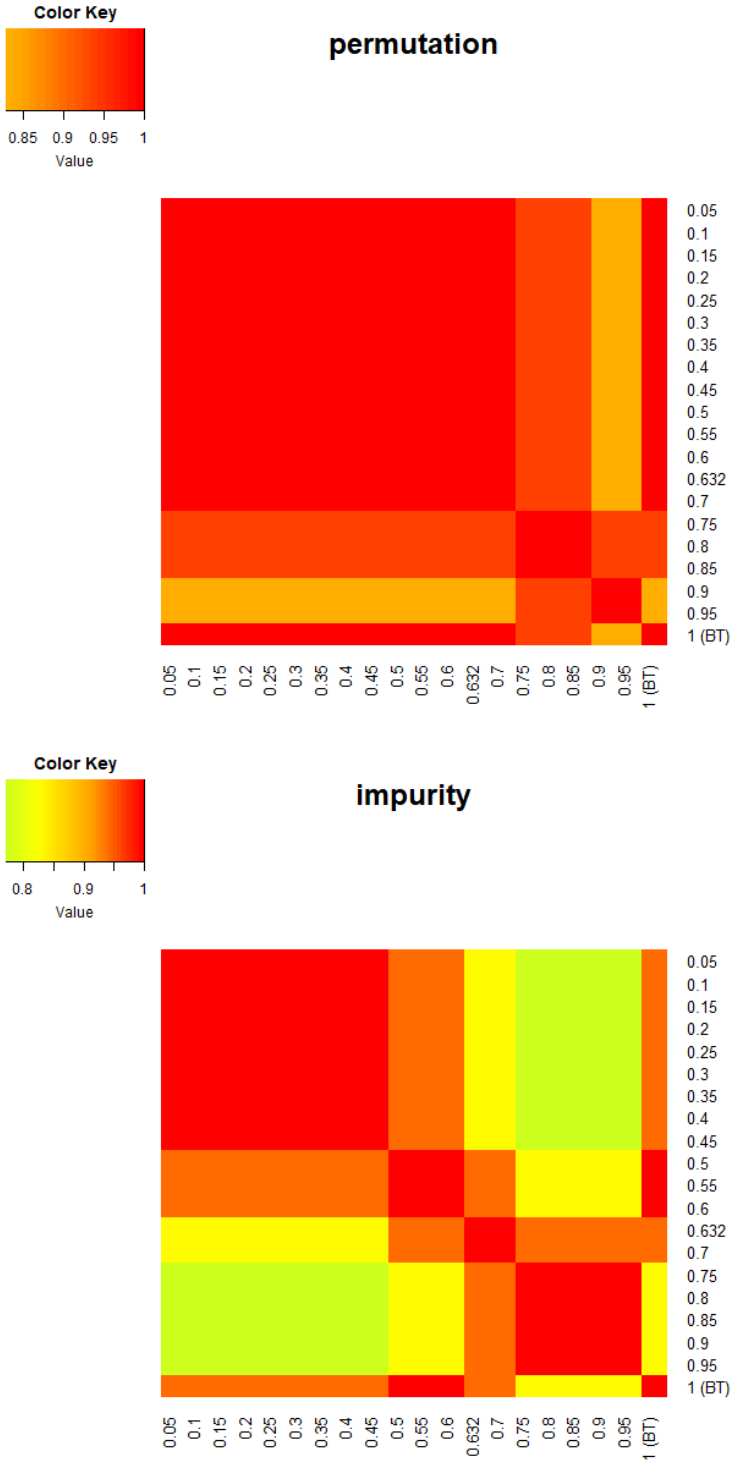


Abbildung 5.22: Spearman-Korrelationen zwischen den Variablenwichtigkeiten der Brustkrebsdaten für unterschiedliche Stichprobengrößen. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

5.4.2 Ergebnisse für GlaucomaM

Splitting-Regel. Für den Datensatz GlaucomaM sind in Abbildung 5.23 die zehn Variablen mit den höchsten Variablenwichtigkeitswerten dargestellt, absteigend geordnet. Oben wieder gemessen mit der Permutationsmethode (pink), unten mit der Unreinheitsmethode (blau), links für `splitrule="gini"` und rechts für `splitrule="extratrees"`. Die ersten drei Plätze sind bei allen vier Varianten gleich, danach variiert die Rangfolge etwas. Für `importance="permutation"` beträgt die Korrelation nach Spearman zwischen den Variablenwichtigkeiten (für alle Kovariablen) mit `splitrule="gini"` und den Variablenwichtigkeiten mit `splitrule="extratrees"` 0.982, für `importance="impurity"` ist sie mit 0.960 etwas niedriger. Die Rangordnungen für die beiden Splitting-Regeln unterscheiden sich also mit der Unreinheitsmethode etwas stärker als mit der Permutationsmethode. Beide Korrelationswerte sind aber sehr hoch und zeigen damit, dass die Unterschiede zwischen den Splitting-Regeln für beide Methoden nur gering sind.

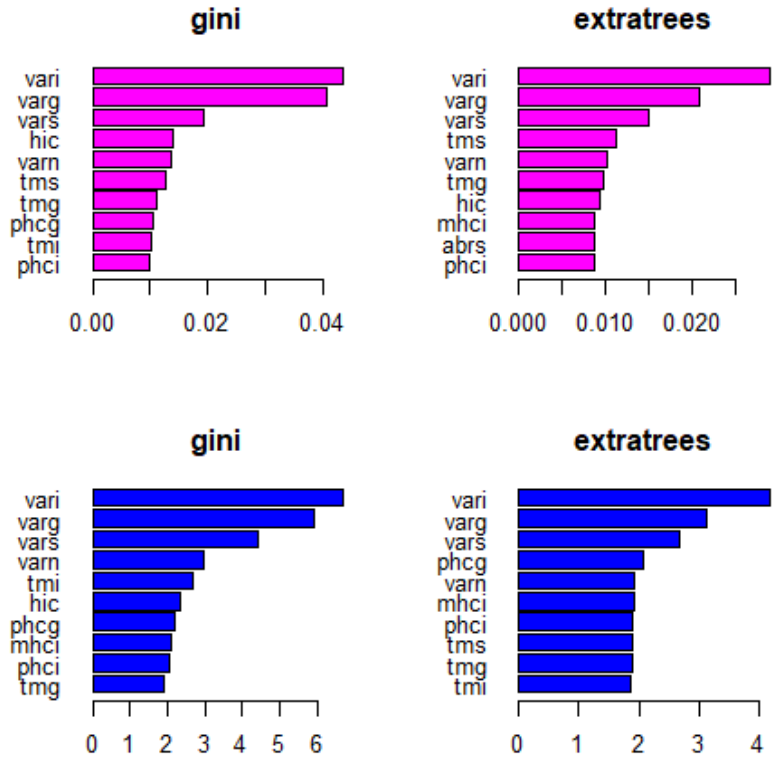


Abbildung 5.23: Variablenwichtigkeit für die beiden unterschiedlichen Splitting-Regeln für den Glaukomdatensatz. In pink für `importance="permutation"`, in blau für `importance="impurity"`

Der Parameter `mtry`. Der Einfluss von `mtry` auf die Variablenwichtigkeit ist für beide Messarten in Abbildung 5.24 dargestellt, jeweils wieder in Form einer *heatmap*. Die Werte für `mtry` laufen von 1 bis 62, aus Gründen der Übersichtilichkeit besteht die Beschriftung an den Rändern der *heatmap* aber nur aus jedem 10. Wert. Die im *color key* abgebildeten Farbabstufungen entsprechen wieder der Korrelationshöhe.

Für beide Messarten sind die Korrelationen zwischen den Variablenwichtigkeiten um so höher, umso näher die beiden Werte von `mtry` aneinander liegen. Dies gilt insbesondere für kleine Werte; mit steigendem `mtry` wird der Effekt schwächer, und die Korrelationen sind auch zwischen weiter auseinanderliegenden Werten sehr hoch. Die Korrelationen werden aber auch für die größeren Werte umso schwächer, umso weiter der zweite `mtry`-Wert vom ersten entfernt ist.

Auch zu sehen ist, dass die Korrelationen für die mit der Permutationsmethode gemessenen Wichtigkeiten für kleine `mtry` deutlich höher sind als für die Wichtigkeiten, die mit der Unreinheitsmethode gemessen wurden. Das heißt, für kleine `mtry` ist die Variabilität zwischen den Rangordnungen der Variablen höher, wenn die Unreinheitsabnahme als Variablenwichtigkeitsmaß verwendet wird.

Minimale Knotengröße. Abbildung 5.25 zeigt die Korrelationen zwischen den Variablenwichtigkeiten, die jeweils für unterschiedliche Einstellungen der minimalen Knotengröße berechnet wurden. Oben sieht man wieder die Korrelationen die sich für die Permutations-Wichtigkeiten ergeben, unten die für die Unreinheits-Wichtigkeiten. Wie zu erwarten ist, sind für beide Messarten die Korrelationen für benachbarte Knotengrößen am größten, also in der Nähe der Diagonale. Je weiter die Knotengrößen auseinander liegen, desto schwächer werden die Korrelationen. Bei den Wichtigkeiten der Unreinheitsmethode sind außerdem die Korrelationen innerhalb der Gruppe der kleinen Knotengrößen von 1 bis 50 sehr groß (1 oder fast 1). Allerdings sind die Korrelationen an den Rändern schwächer als bei der Permutationsmethode, das heißt, die ganz kleinen Knotengrößen korrelieren hier weniger stark mit den ganz großen.

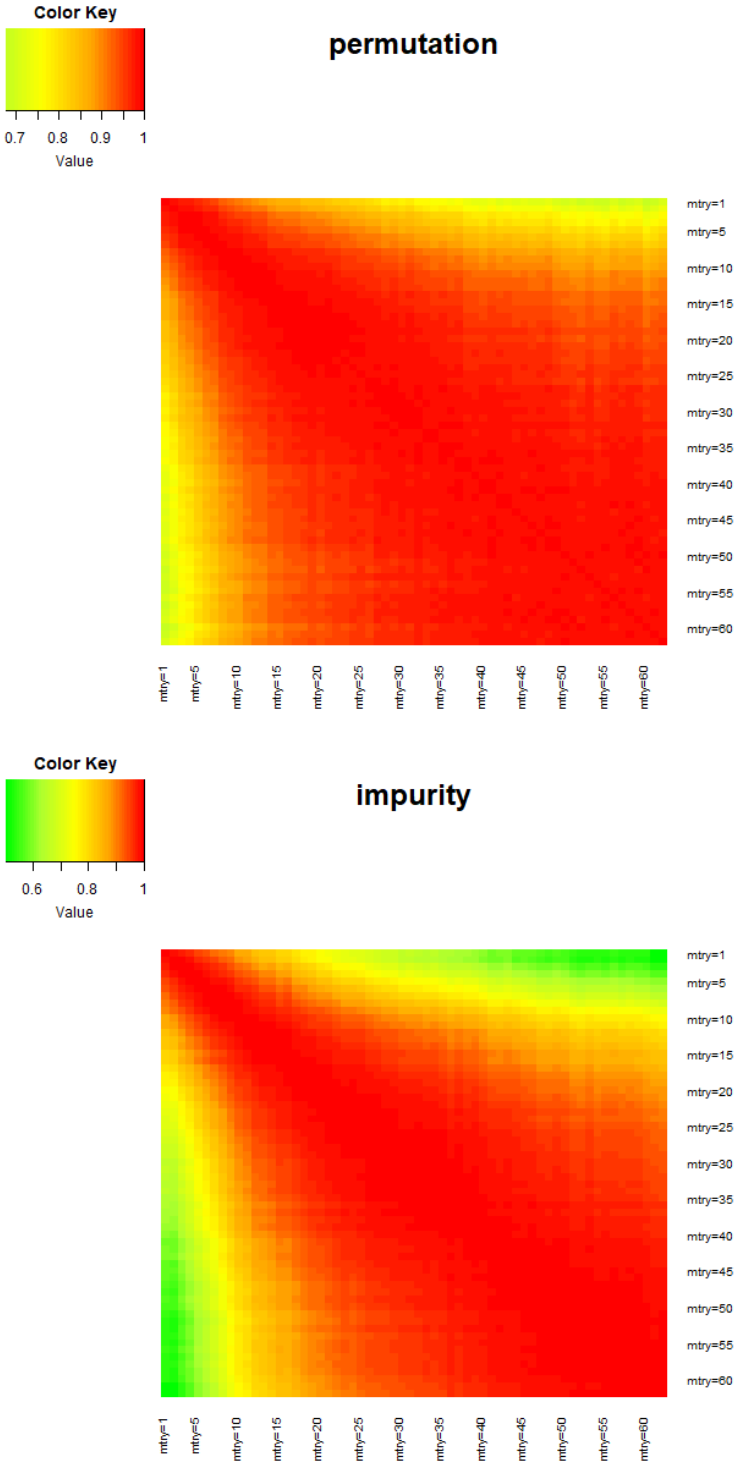


Abbildung 5.24: Spearman-Korrelationen zwischen den Variablenwichtigkeiten des Glaukomdatensatzes für unterschiedliche `mtry`. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

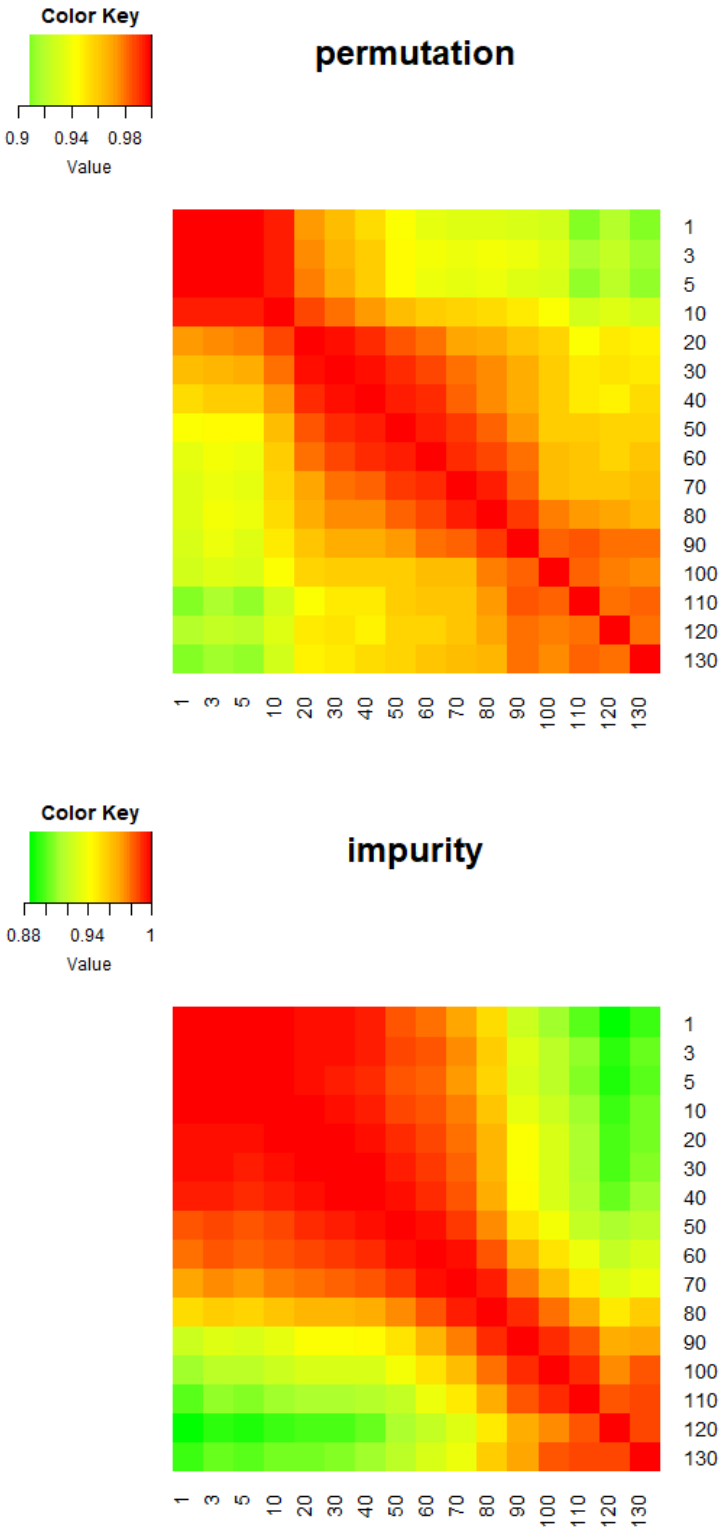


Abbildung 5.25: Spearman-Korrelationen zwischen den Variablenwichtigkeiten des Glaukomdatensatzes für unterschiedliche minimale Knotengrößen. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

Stichprobengröße und subsampling vs. Bootstrap. Korrelationen zwischen den Variablenwichtigkeiten für unterschiedliche Stichprobengrößen sind in Abbildung 5.26 dargestellt, wieder für beide Messarten. Die Auswahl der Stichprobengrößen ist genau wie bei den Brustkrebsdaten; auch die Wichtigkeiten für eine Bootstrap-Stichprobe werden wieder mit den kleinere Stichprobengrößen ohne Zurücklegen verglichen. Die *heatmaps* sehen für beide Messarten sehr ähnlich aus, allerdings ist die Variabilität dieses mal bei der Permutationsmethode etwas höher al sbei der Unreinheitsmethode. Für die Stichprobengrößen < 1 ohne Zurücklegen gilt wie bei den Knotengrößen, dass die Korrelationen für nahe beieinanderliegende Stichprobengrößen entlang der Diagonale am stärksten sind und zu den Rändern hin abnehmen.

Die Bootstrap-Stichprobe ist wie bei den Brustkrebsdaten mit sehr großen Stichprobengrößen von 0.9 oder 0.95 schwächer korreliert als mit Stichprobengrößen zwischen 0.3 und 0.85. Anders als bei den Brustkrebsdaten korreliert die Bootstrap-Stichprobe allerdings am schwächsten mit den ganz kleinen Stichprobengrößen. Dies gilt für beide Messarten.

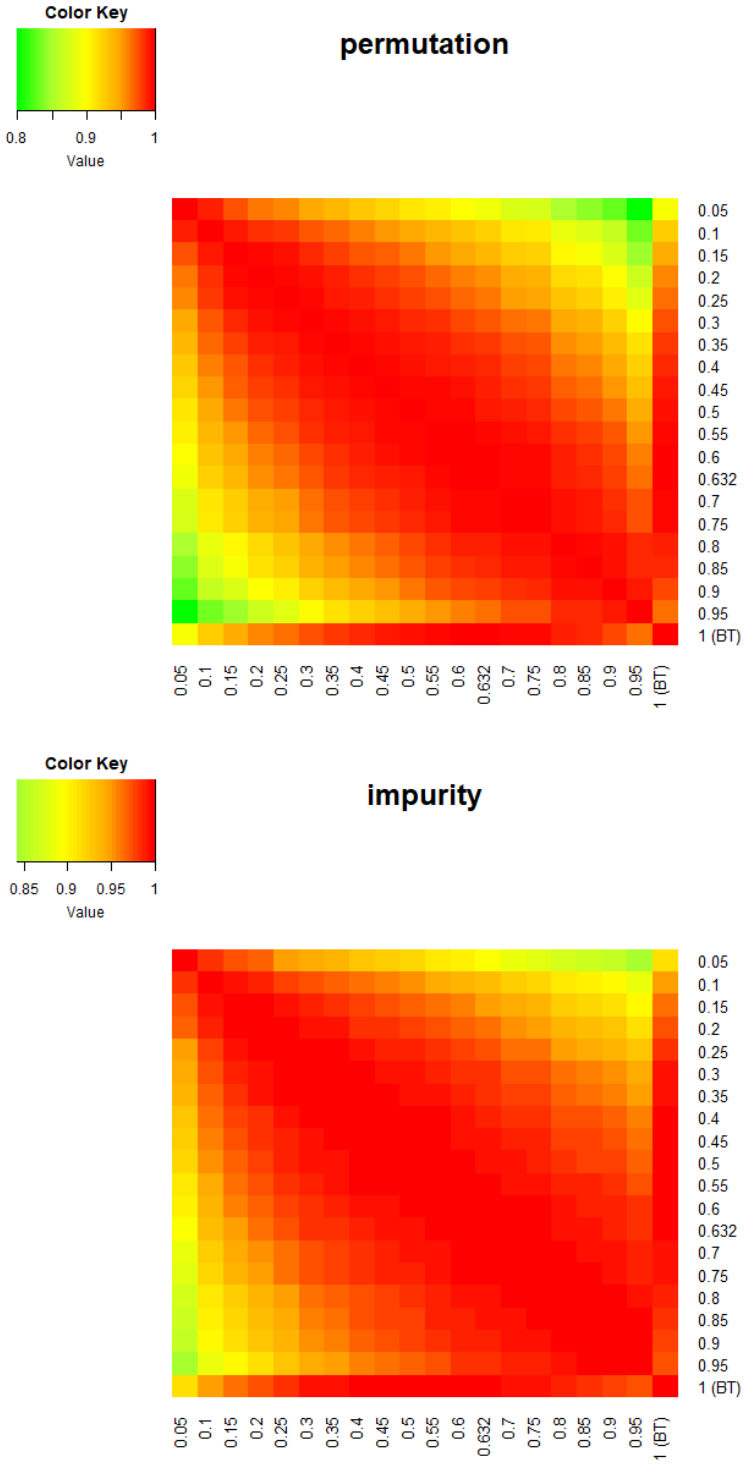


Abbildung 5.26: Spearman-Korrelationen zwischen den Variablenwichtigkeiten des GLaukomdatensatzes für unterschiedliche Stichprobengrößen. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

5.4.3 Ergebnisse für die NHANES-Daten

Splitting-Regel. Auch für die NHANES-Daten soll der Einfluss der verschiedenen Parametereinstellungen auf die Variablenwichtigkeit untersucht werden. In Abbildung 5.27 werden die zehn Variablen mit den höchsten Variablenwichtigkeitswerten gezeigt, absteigend geordnet. Oben wieder gemessen mit der Permutationsmethode (pink), unten mit der Unreinheitsmethode (blau), links für `"variance"`, in der Mitte für `"maxstat"` und rechts für `"extratrees"`. Allerdings steht die Unreinheitsmethode für `"maxstat"` nicht zur Verfügung, deswegen fehlen hier die Werte. Für die Permutationsmethode sind bei allen drei Splitting-Regeln die ersten drei Ränge gleich, für `"maxstat"` und `"extratrees"` sogar die ersten fünf Ränge. Bei der Unreinheitsmethode stehen dieselben drei Variablen auf den ersten drei Plätzen wie bei der Permutationsmethode, aber in anderer Reihenfolge. Für `"variance"` und `"extratrees"` stimmen hier die ersten sechs Ränge überein, und die vier darauffolgenden Ränge werden von denselben Variablen, allerdings in unterschiedlicher Reihenfolge belegt.

Die Korrelation nach Spearman (für die Ränge aller 28 Kovariablen) beträgt bei der Permutationsmethode zwischen `"variance"` und `"maxstat"` 0.868, zwischen `"variance"` und `"extratrees"` 0.833, und zwischen `"maxstat"` und `"extratrees"` ist sie mit 0.762 am niedrigsten. Bei der Unreinheitsmethode ist die Korrelation nach Spearman zwischen `"variance"` und `"extratrees"` mit 0.895 deutlich höher, das heißt, die Wahl der Splitting-Regel hat bei der Unreinheitsmethode weniger Auswirkungen auf die Variablenwichtigkeiten.

Der Parameter `mtry`. Die Korrelationen zwischen den Wichtigkeiten mit unterschiedlichen `mtry` werden in Abbildung 5.28 dargestellt. Der *color key* ist, wie bereits oben erwähnt, hier nicht so angepasst, dass ein Farbton für beide *heatmaps* dem gleichen Korrelationswert entspricht, da die `"impurity"`-*heatmap* nur extrem hohe Korrelationen aufweist und sonst für sie kaum noch Abstufungen zu erkennen wären. Die Variabilität für die Wichtigkeiten mit verschiedenen `mtry` ist also für die Permutationsmethode größer, was auch trotz der unterschiedlichen *color keys* gut zu erkennen ist. Ansonsten sind für beide Messarten wie zu erwarten die Korrelationen um so stärker, je näher sie an der Diagonale liegen. Für die Permutationamethode sind zusätzlich die Korrelationen insgesamt um so schwächer, je kleiner `mtry` ist.

Der Parameter `mtry`. Die Korrelationen zwischen den Wichtigkeiten mit unterschiedlichen `mtry` werden in Abbildung 5.28 dargestellt. Der *color key* ist, wie

bereits oben erwähnt, hier nicht so angepasst, dass ein Farbton für beide *heatmaps* dem gleichen Korrelationswert entspricht, da die `"impurity"`-*heatmap* nur extrem hohe Korrelationen aufweist und sonst für sie kaum noch Abstufungen zu erkennen wären. Die Variabilität für die Wichtigkeiten mit verschiedenen `mtry` ist also für die Permutationsmethode größer, was auch trotz der unterschiedlichen *color keys* gut zu erkennen ist. Ansonsten sind für beide Messarten wie zu erwarten die Korrelationen um so stärker, je näher sie an der Diagonale liegen. Für die Permutationamethode sind zusätzlich die Korrelationen insgesamt um so schwächer, je kleiner `mtry` ist.

Der Parameter `mtry`. Die Korrelationen zwischen den Wichtigkeiten mit unterschiedlichen `mtry` werden in Abbildung 5.28 dargestellt. Der *color key* ist, wie bereits oben erwähnt, hier nicht so angepasst, dass ein Farbton für beide *heatmaps* dem gleichen Korrelationswert entspricht, da die `"impurity"`-*heatmap* nur extrem hohe Korrelationen aufweist und sonst für sie kaum noch Abstufungen zu erkennen wären. Die Variabilität für die Wichtigkeiten mit verschiedenen `mtry` ist also für die Permutationsmethode größer, was auch trotz der unterschiedlichen *color keys* gut zu erkennen ist. Ansonsten sind für beide Messarten wie zu erwarten die Korrelationen um so stärker, je näher sie an der Diagonale liegen. Für die Permutationamethode sind zusätzlich die Korrelationen insgesamt um so schwächer, je kleiner `mtry` ist.

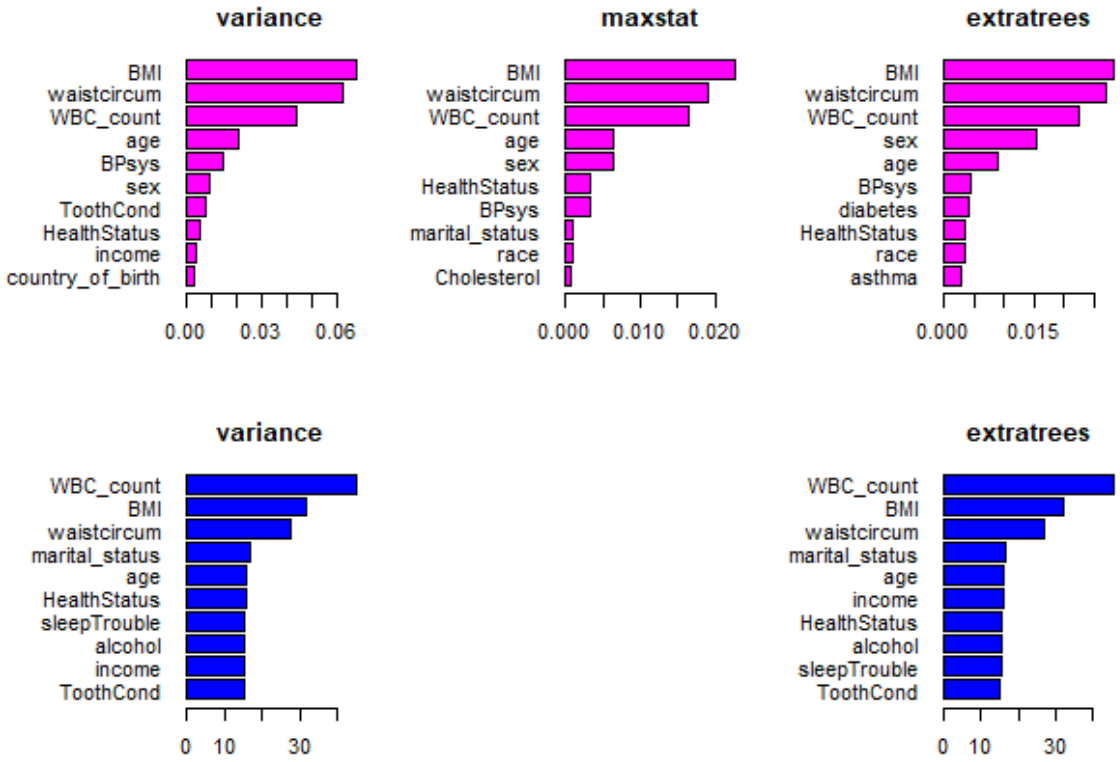


Abbildung 5.27: Variablenwichtigkeit für die unterschiedlichen Splitting-Regeln. In pink für `importance="permutation"`, in blau für `importance="impurity"`. Für `"maxstat"` ist die Unreinheitsmethode nicht implementiert.

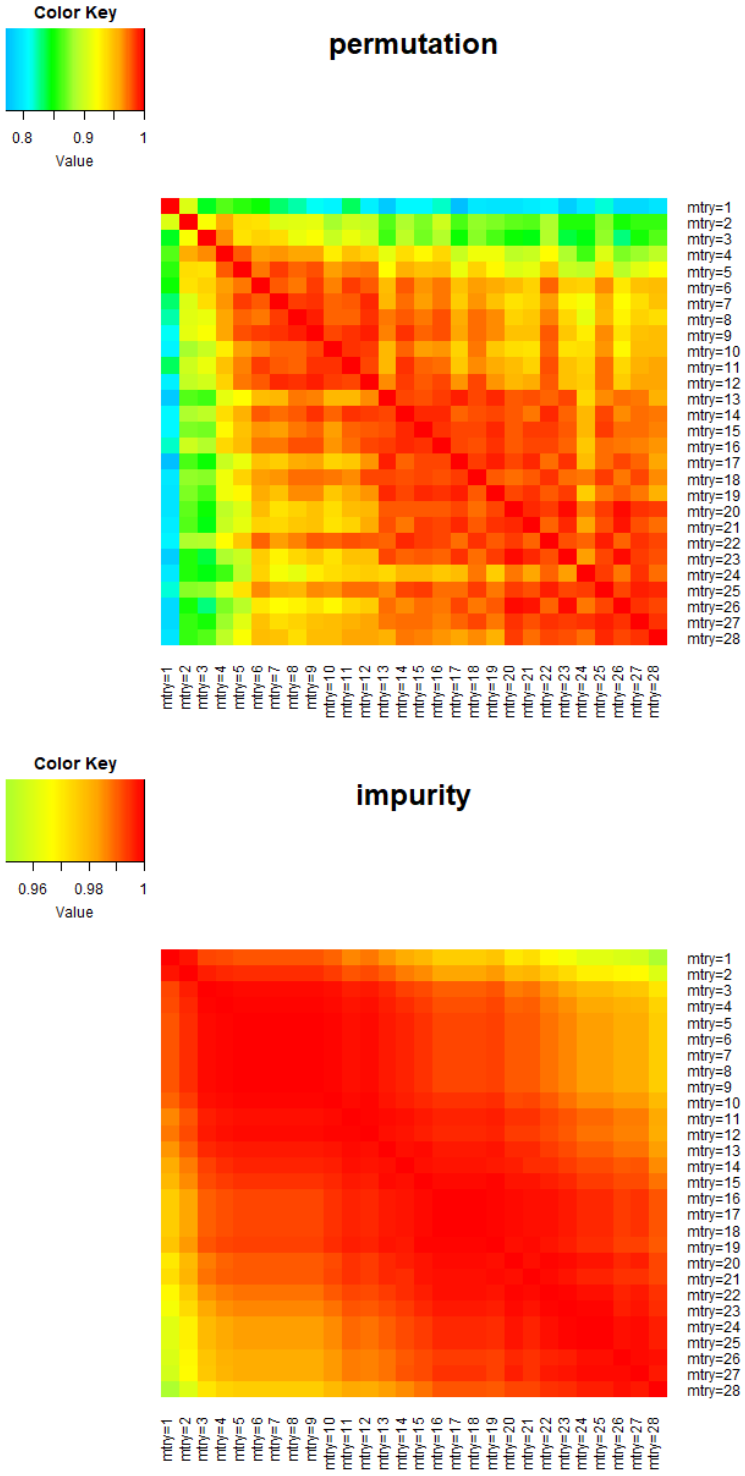


Abbildung 5.28: Spearman-Korrelationen zwischen den Variablenwichtigkeiten der NHANES-Daten für unterschiedliche `mtry`. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

Minimale Knotengröße. Auch für die beiden *heatmaps* in Abbildung 5.29, die die Korrelationen zwischen den Wichtigkeiten mit unterschiedlicher Knotengröße zeigen, ist der *color key* nicht einheitlich, da die Korrelationen für die mit der Unreinheitsmethode gemessenen Wichtigkeiten viel stärker sind. Auch hier gilt wieder, dass die Korrelationen zur Diagonale hin, also für nahe beieinander liegende Knotengrößen, besonders stark werden. Diesmal werden aber die Korrelationen für beide Messarten zusätzlich umso kleiner, umso größer die minimalen Knotengrößen sind. Gut zu erkennen ist auch eine Blockstruktur, beispielsweise ist für die Permutationsmethode die Grupper der Knotengrößen von 20 bis 300 untereinander sehr stark korreliert, während die Korrelation zwischen Knotengröße 300 und 400 viel schwächer ist.

Stichprobengröße und subsampling vs. Bootstrap. Als letzter Parameter wird auch für den NHANES-Datensatz noch der Einfluss der Stichprobengröße ohne Zurücklegen sowie zum Vergleich einer Bootstrap-Stichprobe gezeigt. Die entsprechenden *heatmaps* sind in Abbildung 5.30 zu sehen. Wieder ist leicht zu erkennen, dass die durch unterschiedliche Stichprobengrößen verursachte Variabilität für die Permutationsmethode deutlich größer ist (der *color key* ist auch hier nicht einheitlich). Für die Unreinheitsmethode sind die Korrelationen alle größer 0.986, die unterschiedlichen Stichprobengrößen bewirken also kaum Änderungen für die Variablenwichtigkeiten. Der zuvor immer wieder beobachtete Effekt, dass die Bootstrap-Stichprobe mit mittleren Größen höher korreliert ist als mit den größten Stichprobengrößen zeigt sich auch hier bei der Permutationsmethode, nicht aber bei der Unreinheitsmethode. Für die Unreinheitsmethode werden die Korrelationen umso stärker, umso größer die Stichprobengrößen werden. Schon ab einer Größe von 0.3 gehen die Korrelationen allerdings so nahe an die 1, dass sich kaum noch Abstufungen erkennen lassen. Bei der Permutationsmethode fällt auf, dass besonders die ganz kleinen (0.5) und die ganz großen Stichprobengrößen (0.95) besonders schwach mit anderen Stichprobengrößen korrelieren.

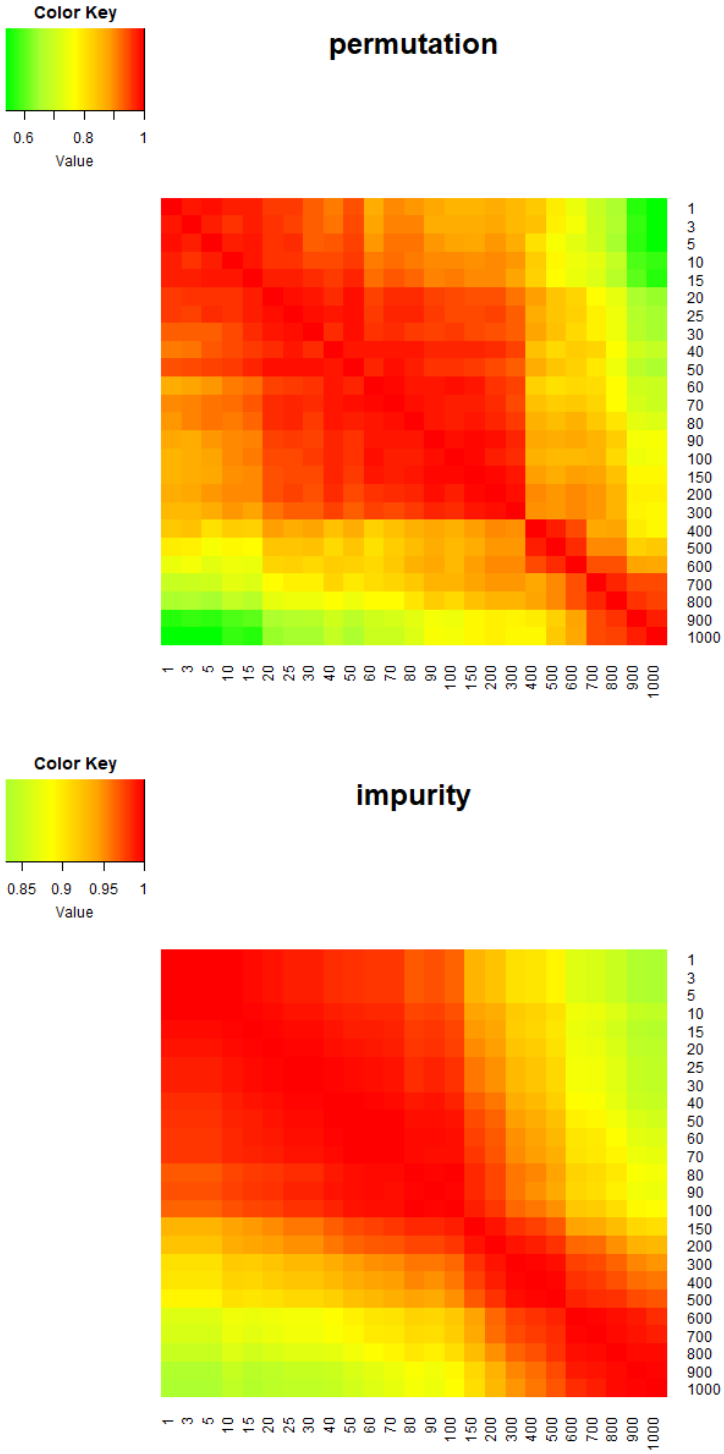


Abbildung 5.29: Spearman-Korrelationen zwischen den Variablenwichtigkeiten der NHANES-Daten für unterschiedliche minimale Knotengrößen. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

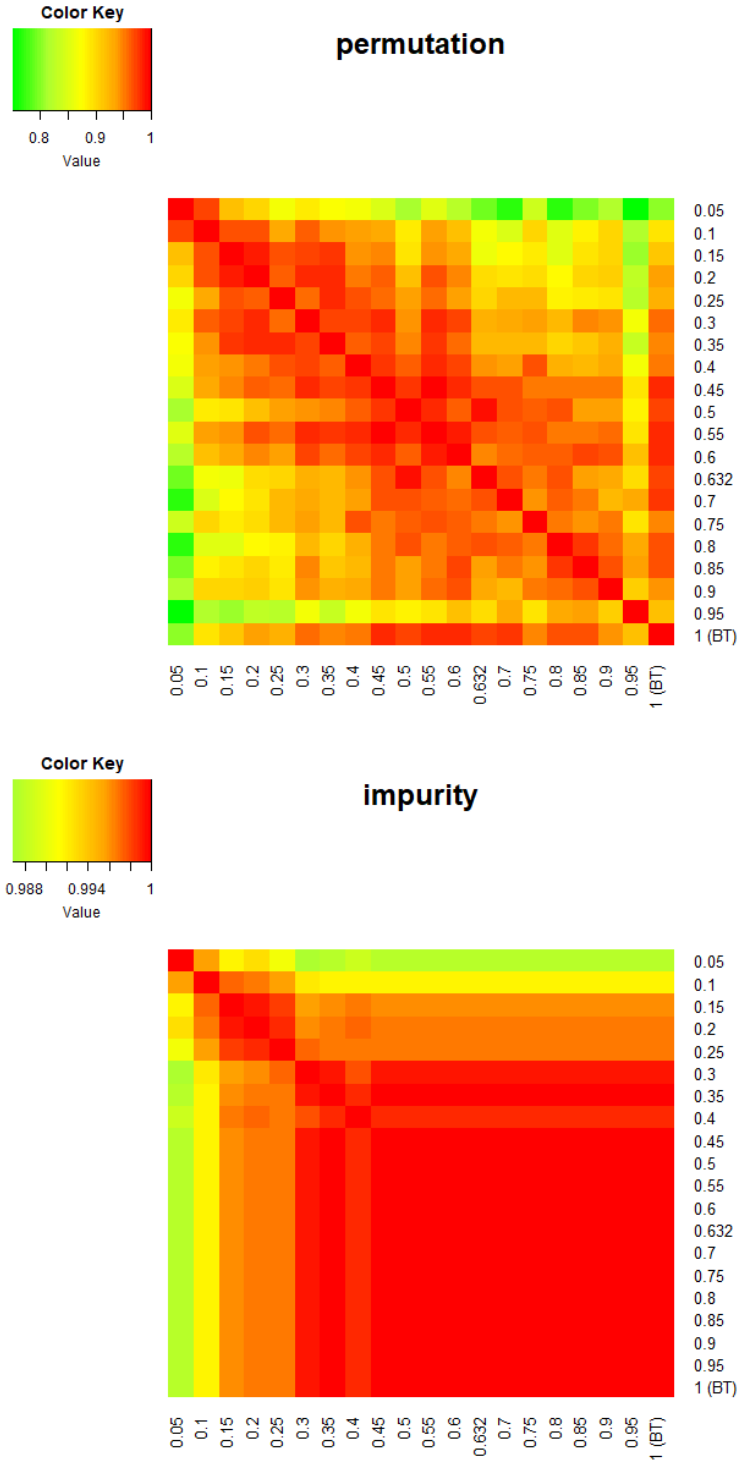


Abbildung 5.30: Spearman-Korrelationen zwischen den Variablenwichtigkeiten der NHANES-Daten für unterschiedliche Stichprobengrößen. Oben: gemessen mit `"permutation"`, unten: gemessen mit `"impurity"`.

6 Zusammenfassung

6.1 Parameterwahl und Prädiktionsgüte

Die Auswirkungen der Parameterveränderungen auf die Prädiktiongüte unterscheiden sich sehr stark zwischen den Brustkrebsdaten und den beiden anderen Beispieldatensätzen. Für die Brustkrebsdaten lassen sich mit allen untersuchten Parametern zum Teil erhebliche Verringerungen der Fehlklassifikationsrate bewirken. Dabei sind die Verbesserungen immer für die Testdaten viel stärker ausgeprägt als für die OOB-Daten. Allerdings stellt dieser Datensatz mit seinen unerwartet inkongruenten Ergebnissen vielleicht insgesamt einen Sonderfall dar.

Für den Glaukomdatensatz und die NHANES-Daten ähneln sich die Ergebnisse hingegen sehr. Bei beiden Datensätzen haben die Parameter `mtry` und `min.node.size` zwar durchaus Einfluss auf die Prädiktionsgüte, doch mit den Default-Einstellungen erreicht man schon nahezu minimale Werte für die Fehlklassifikationsrate bzw. den MSE. Durch eine Änderung der Einstellungen lässt sich also keine oder nur eine vernachlässigbar geringe Verbesserung der Prädiktionsgüte erreichen. Auch mit subsampling statt der Verwendung einer Bootstrap-Stichprobe lässt sich keine merkliche Verbesserung erzielen, wobei die Stichprobengröße und die Entscheidung zwischen subsampling und Bootstrap-Stichprobe ohnehin nur wenig Einfluss haben. Für die Glaukomdaten schneiden lediglich sehr kleine Stichproben ≤ 0.35 deutlich schlechter ab, für die NHANES-Daten dagegen nur sehr große Stichproben ohne Zurücklegen (≥ 0.95). Auch durch eine Veränderung der Splitting-Regel zeigt sich für beide Datensätze keine Verbesserung der Prädiktionsgüte.

Als Fazit lässt sich festhalten, dass die Default-Einstellungen für Random Forest im `R`-Paket `ranger` durchaus sinnvoll gewählt scheinen. Datenabhängig kann es aber durch eine Änderung der Einstellungen doch zu einer deutlichen Verbesserungen der Prädiktionsgüte kommen. Eine Überprüfung unterschiedlicher Parametereinstellungen mittels Kreuzvalidierung kann also durchaus lohnend sein. Um herauszufinden, ob es bestimmte Merkmale eines Datensatzes gibt, die möglicherweise ein Hinweis darauf sind, dass die Default-Einstellungen nicht zu optimalen Ergebnissen führen (wie zum Beispiel die sehr ungleiche Verteilung der Klassen bei binärem Response), könnten weitere Analysen mit mehr Datensätzen durchgeführt werden. Interessant wäre es außerdem, die Analysen auf Datenbeispiele mit mehrkategorialem Response oder Überlebensdauern als Response auszuweiten oder auch andere Random-Forest-Implementierungen zu verwenden.

6.2 Parameterwahl und Variablenwichtigkeit

Auch die Variablenwichtigkeiten werden von den Parametereinstellungen beeinflusst. Die Art und das Ausmaß dieses Einflusses unterscheidet sich zum Teil zwischen den Datensätzen, zwischen den Parametern und zwischen den Messarten. Es lassen sich aber auch Gemeinsamkeiten finden.

Es ist nicht überraschend, dass meistens kleine Änderungen in den Parametereinstellungen kleinere Unterschiede für die Variablenwichtigkeiten verursachen als große Änderungen. Gemeint ist, dass beispielsweise zwischen Knotengröße 1 und 5 weniger Unterschiede in den Variablenwichtigkeiten bestehen als zwischen Knotengrößen 1 und 100. Dieser Effekt zeigt sich bei allen drei Datensätzen, allen abstufbaren Parametern und für beide Messarten. Eine Ausnahme dafür sind die Variablenwichtigkeiten für unterschiedliche Knotengrößen bei den Brustkrebsdaten gemessen mit der Permutationsmethode. Hier korrelieren sehr kleine Knotengrößen mit sehr großen Knotengrößen stärker als mit mittelgroßen.

Obwohl unterschiedliche Parametereinstellungen Veränderungen in der Variablenrangfolge bewirken können, sind diese Unterschiede in der Rangfolge fast immer klein genug, dass die unterschiedlichen Rangfolgen immer noch stark positiv miteinander korrelieren. Die einzige Ausnahme bildet die Änderung der minimalen Knotengröße bei den Brustkrebsdaten gemessen mit der Unreinheitsmethode. Während hier bei der Permutationsmethode die unterschiedlichen Knotengrößen nur wenig Veränderung verursachen, kommt es bei der Unreinheitsmethode zwischen sehr kleinen und sehr großen Knotengrößen sogar zu negativen Korrelationen.

In Hinblick auf die Stichprobengröße und den Vergleich zwischen subsampling und Bootstrap-Stichprobe muss noch erwähnt werden, dass die Bootstrap-Stichprobe im Allgemeinen mit mittleren Stichprobengrößen ohne Zurücklegen immer höher korreliert als mit ganz großen Stichproben ≥ 0.9 ohne Zurücklegen. Einzige Ausnahme hiervon sind die mit der Unreinheitsmethode gemessenen Variablenwichtigkeiten der NHANES-Daten.

Vergleicht man die Ergebnisse für die beiden Messarten miteinander, so fällt auf, dass für die Brustkrebsdaten und die Glaukomdaten die Unterschiede in den Variablenrangfolgen zwischen den verschiedenenen Parametereinstellungen fast immer für die Unreinheitsmethode größer sind (einzige Ausnahme bildet die Stichprobengröße für die Glaukomdaten; hier ist die Variabilität mit der Permutationsmethode geringfügig größer). Bei den NHANES-Daten hingegen ist die Variabilität der Rangfolgen immer für die Permutationsmethode größer.

7 Literatur

Boulesteix, A.-L., R. De Bin, X. Jiang und M. Fuchs (2017). IPF-LASSO: Integrative L_1-Penalized Regression with Penalty Factors for Prediction Based on Multi-Omics Data. *Computational and Mathematical Methods in Medicine* 2017. 1-14.

Breiman, L. (1996). Bagging Predictors. *Machine Learning* 24:2. 123–140.

Breiman, L., J. H. Friedman, R. A. Olshen und C.J. Stone (1998). Classification and Regression Trees. (Reprint). Boca Raton u.a.: Chapman & Hall/CRC.

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

Cutler, A., D. R. Cutler und J. R. Stevens (2012). Random Forests. *Ensemble Machine Learning: Methods and Applications*. New York: Springer. 157-176.

Fahrmeir, L., W. Brachinger, A. Hamerle und G. Tutz (1996). *Multivariate statistische Verfahren*. Berlin, New York: Walter de Gruyter.

Ferri, C., J. Hernández-Orallo und R. Modroiu (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters* 30(1): 27-38.

Geurts, P., D. Ernst und L. Wehenkel (2006). Extremely randomized trees. *Machine Learning* 63. 3-42.

Hanley, J. A. und B. J. McNeil (1982). The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology* 143(1). 29-36.

Harrell, F. E. (2018). *Hmisc: Harrell miscellaneous*. R package version 4.1.1.

Hastie, T., R. Tibshirani und J. H. Friedman (2009). *The elements of statistical learning: Data mining, inference and prediction* (Second edition, corrected 7th printing ed.). Springer series in statistics. New York: Springer.

C. Hatzis, L. Pusztai, V. Valero et al. (2011). A genomic predictor of response and survival following taxane-anthracycline chemotherapy for invasive breast cancer. *JAMA* 305(18). 1873–1881.

Ho, T. K. (1998). The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8). 832–844.

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

Hothorn, T. (2017). *TH.data: TH's Data Archive*. R package version 1.0.8.

Rospleszcz, S., S. Janitza und A.-L. Boulesteix (2016). Categorical variables with many categories are preferentially selected in bootstrap-based model selection procedures for multivariable regression models. *Biometrical Journal* 58(3). 652-673.

Strobl, C., A.-L. Boulesteix und T. Augustin (2007). Unbiased split selection for classification trees based on the Gini index. *Computational Statistics & Data Analysis* 52(1). 483–501.

Strobl, C., A.-L. Boulesteix, T.Kneib, T. Augustin und A. Zeileis (2008). Conditional variable importance for random forests. *BMC Bioinformatics* 9(1). 307-317.

Strobl, C., J. Malley und G. Tutz (2009). An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests. *Technical Report 55, Department of Statistics, University of Munich.*
https://epub.ub.uni-muenchen.de/10589/1/partitioning.pdf

Tutz, G. (2012). *Regression for categorical data*. Cambridge series in statistical and probabilistic mathematics. Cambridge: Cambridge University Press.

Warnes, G. R., B. Bolker et al. (2016). *gplots: Various R Programming Tools for Plotting Data*. R package version 3.0.1.

Wright, M. N. (2018). *ranger: A Fast Implementation of Random Forests*. R package version 0.9.0.

Wright, M. N., T. Dankowski und A. Ziegler (2017). Unbiased split variable selection for random survival forests using maximally selected rank statistics. *Statistics in Medicine* 36. 1272–1284.

Yan, Y. (2016). *MLmetrics: Machine Learning Evaluation Metrics*. R package version 1.1.1.

8 Anhang

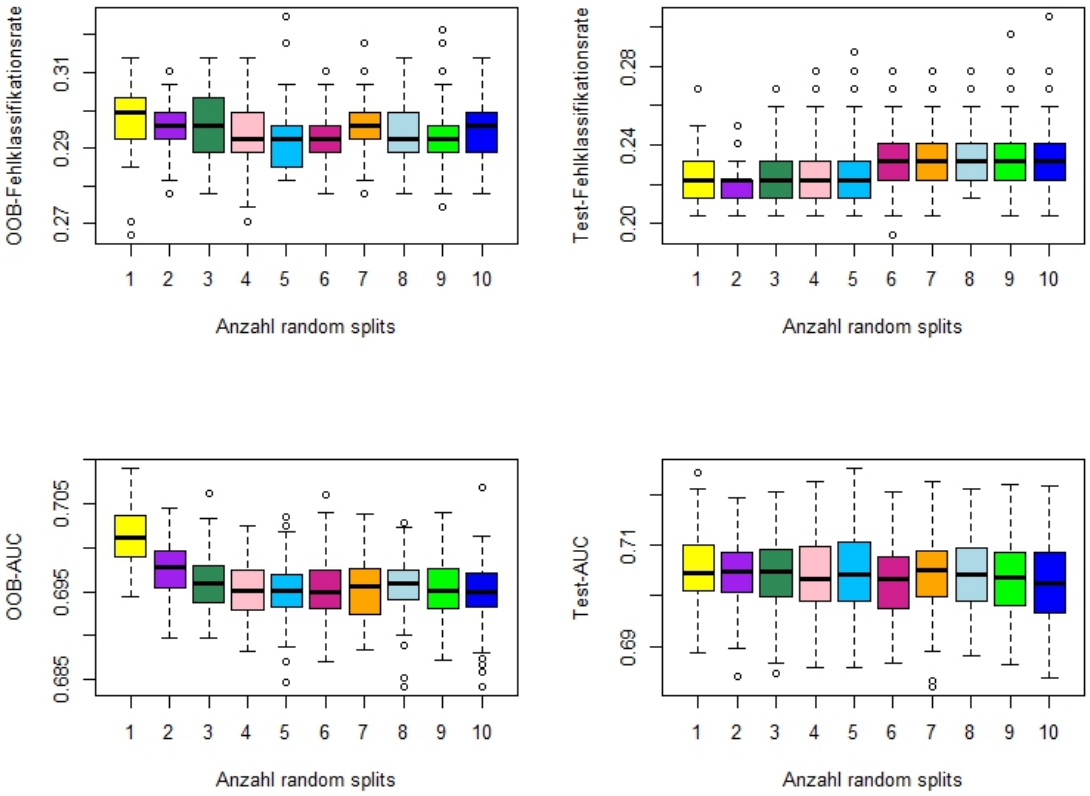


Abbildung 8.1: Fehlklassifikationsrate (oben) und AUC (unten) in Abhängigkeit von der Anzahl der random splits bei der Splitting-Regel `"extratrees"`. Links jeweils für die OOB-Daten, rechts jeweils für die Testdaten.

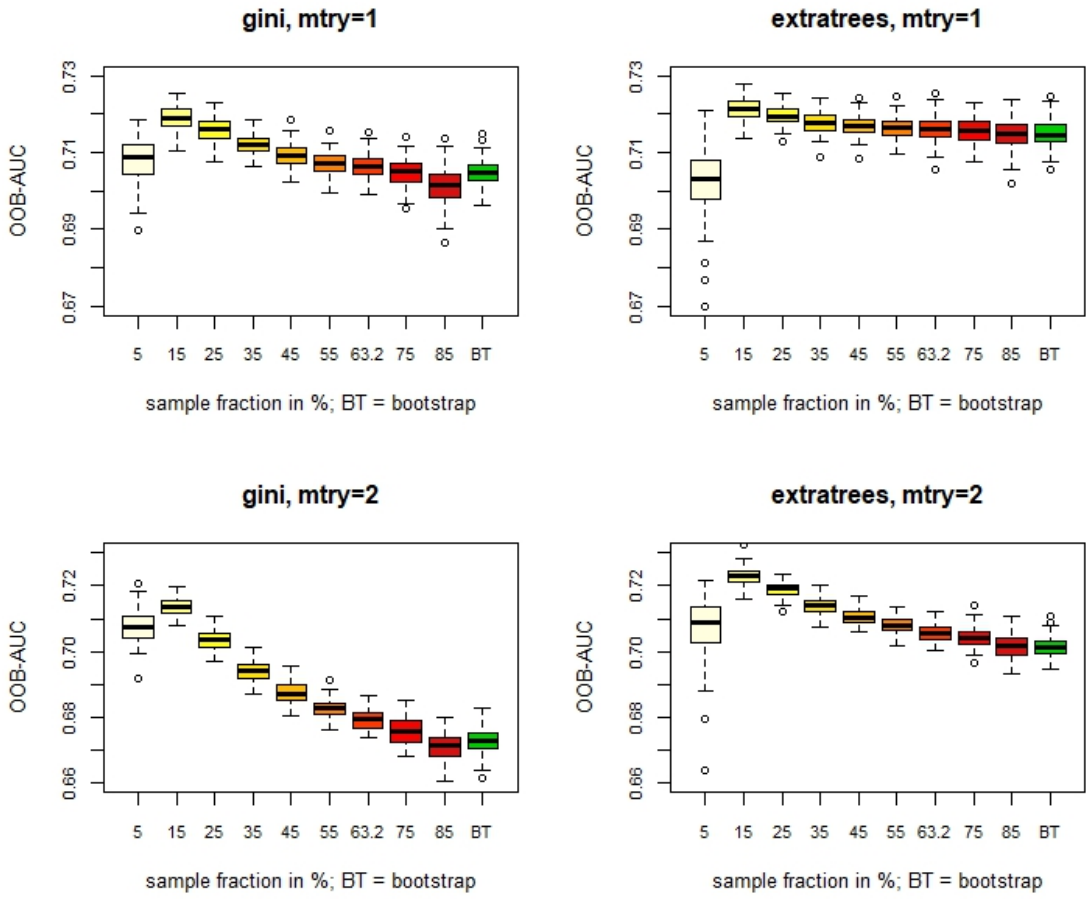


Abbildung 8.2: OOB-AUC für die Brustkrebsdaten in Abhängigkeit von der Stichprobengröße. Oben für `mtry=1`, unten für `mtry=2`), links für die Splitting-Regel `"gini"`, rechts für `"extratrees"`.

Anteil	0.05	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	BT
gini, `mtry`=1	0.708	0.719	0.716	0.712	0.709	0.707	0.706	0.705	0.701	0.705
ET, `mtry`=1	0.702	0.721	0.720	0.718	0.717	0.716	0.716	0.716	0.715	0.715
gini, `mtry`=2	0.708	0.714	0.703	0.694	0.688	0.683	0.679	0.676	0.671	0.673
ET, `mtry`=2	0.708	0.723	0.719	0.714	0.711	0.708	0.706	0.704	0.702	0.701

Tabelle 8.1: Mittelwerte der OOB-AUCs in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 8.2). Der höchste Wert in jeder Zeile ist rot markiert.

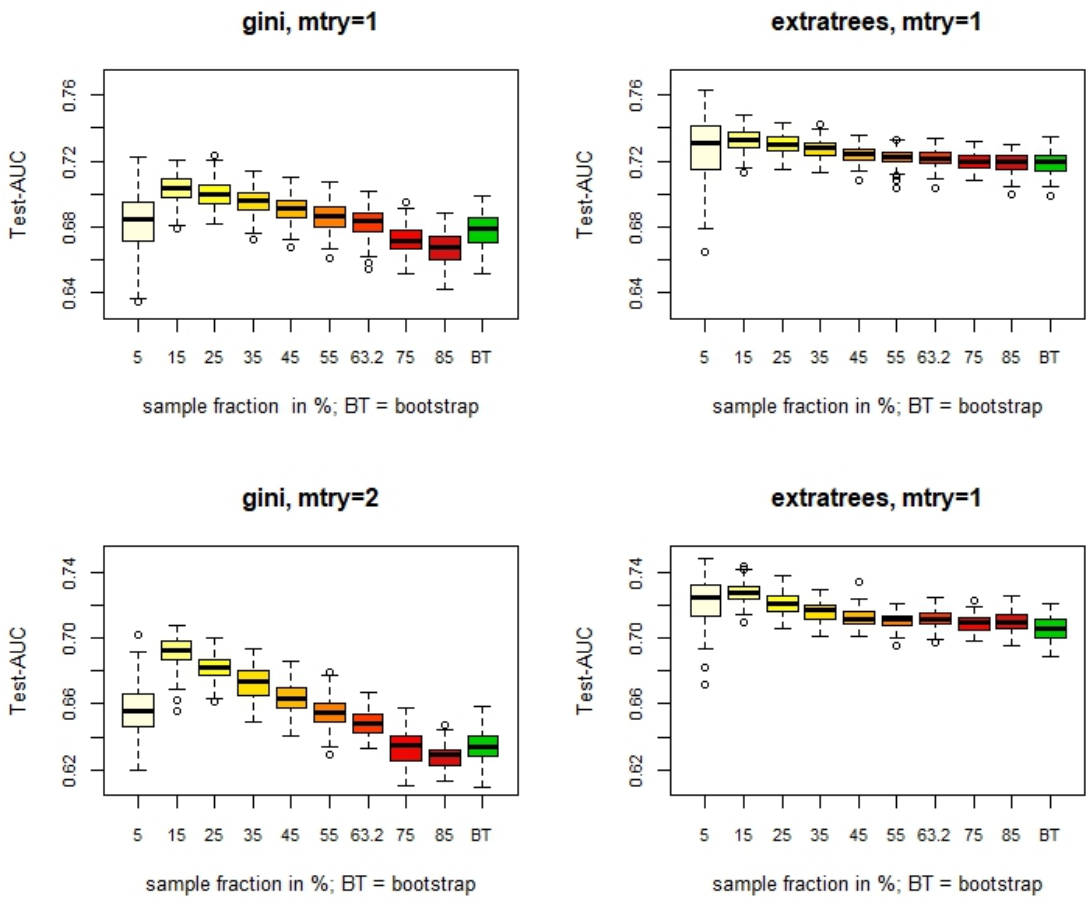


Abbildung 8.3: Test-AUC für die Brustkrebsdaten in Abhängigkeit von der Stichprobengröße. Oben für `mtry=1`, unten für `mtry=2`), links für die Splitting-Regel `"gini"`, rechts für `"extratrees"`.

Anteil	0.05	0.15	0.25	0.35	0.45	0.55	0.632	0.75	0.85	BT
gini, `mtry`=1	0.683	0.703	0.700	0.696	0.691	0.686	0.683	0.672	0.667	0.678
ET, `mtry`=1	0.727	0.732	0.730	0.727	0.724	0.722	0.722	0.720	0.719	0.719
gini, `mtry`=2	0.656	0.692	0.682	0.673	0.663	0.655	0.648	0.633	0.628	0.634
ET, `mtry`=2	0.722	0.727	0.721	0.716	0.712	0.711	0.712	0.709	0.710	0.706

Tabelle 8.2: Mittelwerte der Test-AUCs in Abhängigkeit von der Stichprobengröße (vgl. Abbildung 8.3). Der höchste Wert in jeder Zeile ist rot markiert.

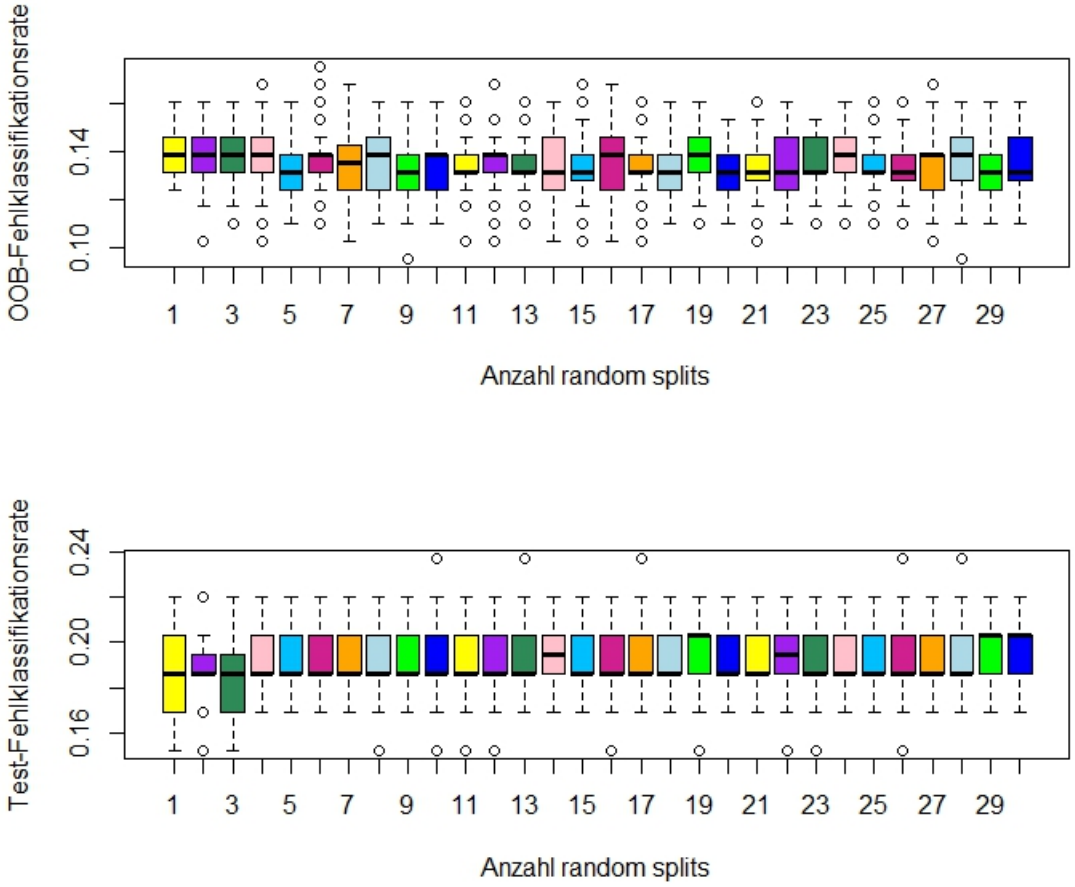


Abbildung 8.4: Fehlklassifikationsrate für den Glaukomdatensatz in Abhängigkeit von der Anzahl der random splits bei `splitrule = "extratrees"`. Links für die OOB-Daten, rechts für die Testdaten.

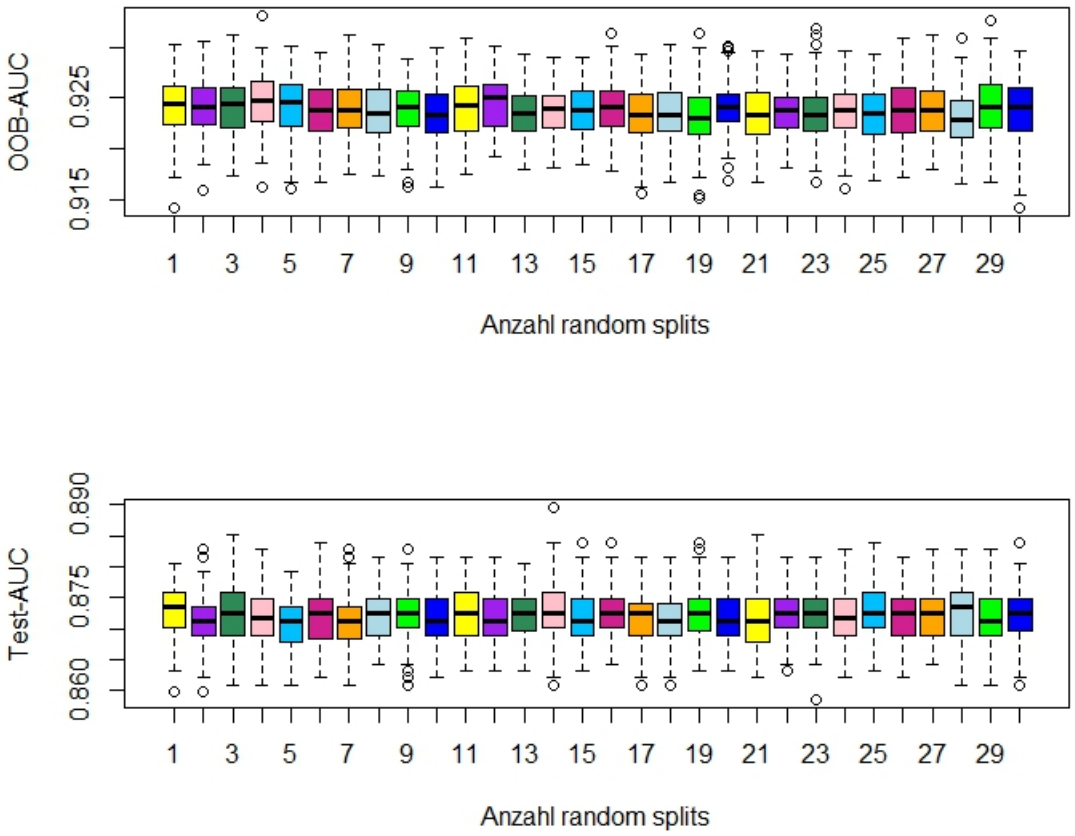


Abbildung 8.5: AUC für den Glaukomdatensatz in Abhängigkeit von der Anzahl der random splits bei `splitrule = "extratrees"`. Links für die OOB-Daten, rechts für die Testdaten.

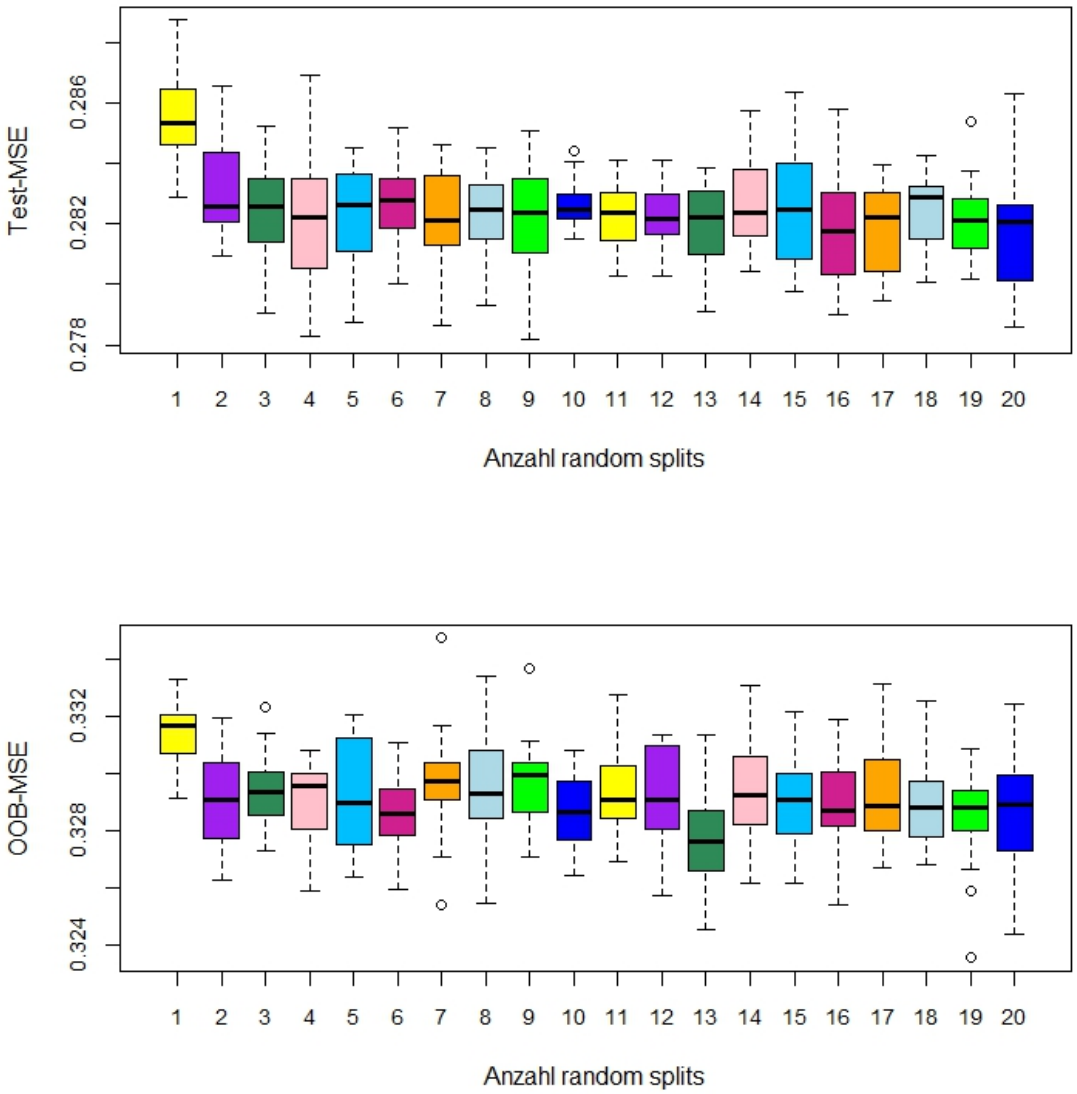
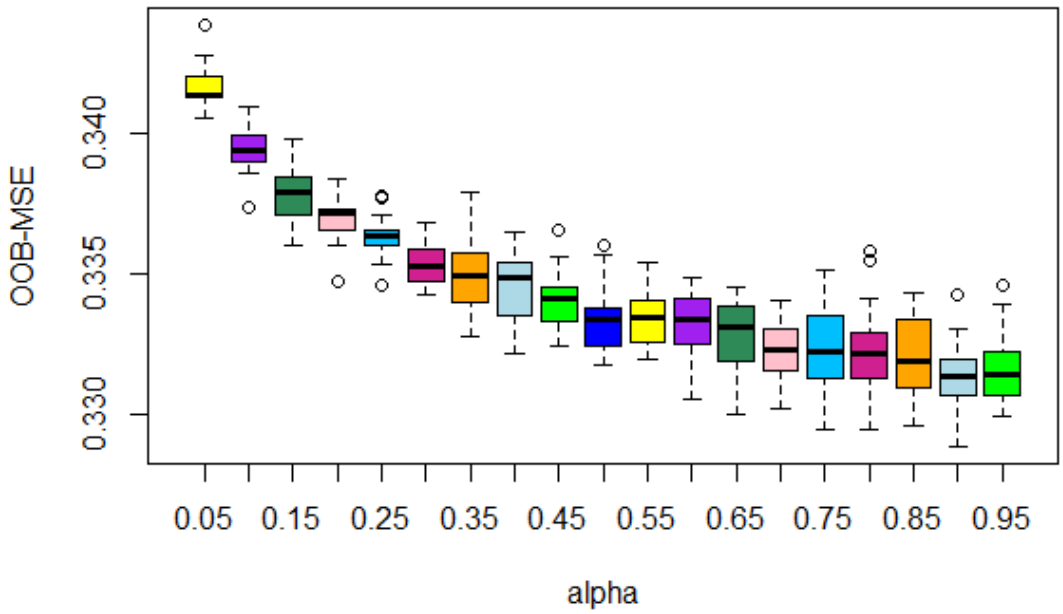


Abbildung 8.6: OOB- und Test-MSE für die NHANES-Daten in Abhängigkeit von der Anzahl der random splits bei `splitrule = "extratrees"`



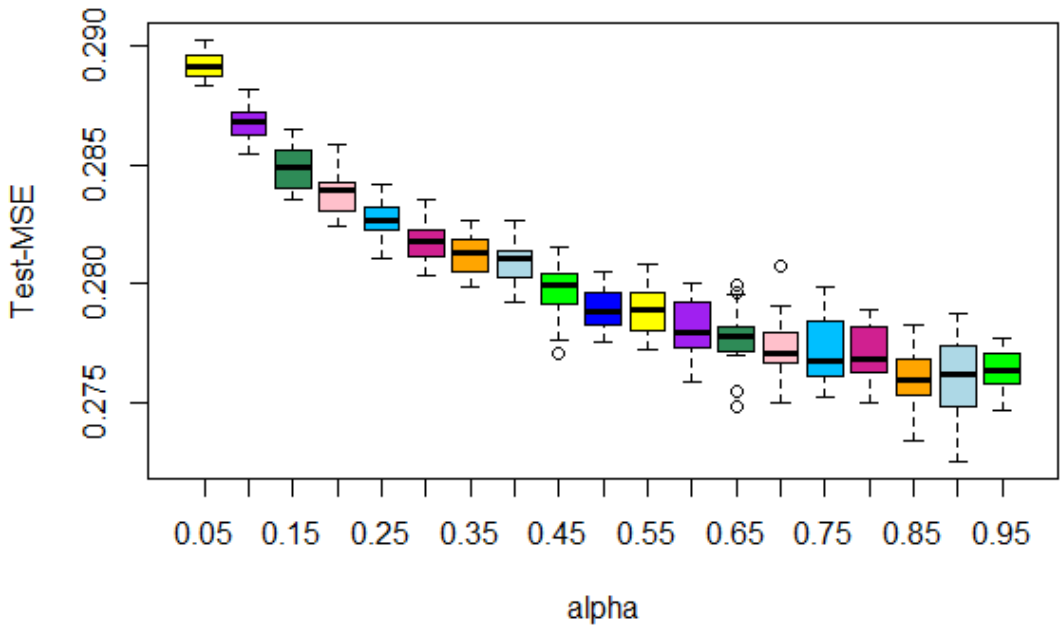


Abbildung 8.7: OOB- und Test-MSE für die NHANES-Daten in Abhängigkeit von dem Parameter `alpha` für die Splitting-Regel `"maxstat"`

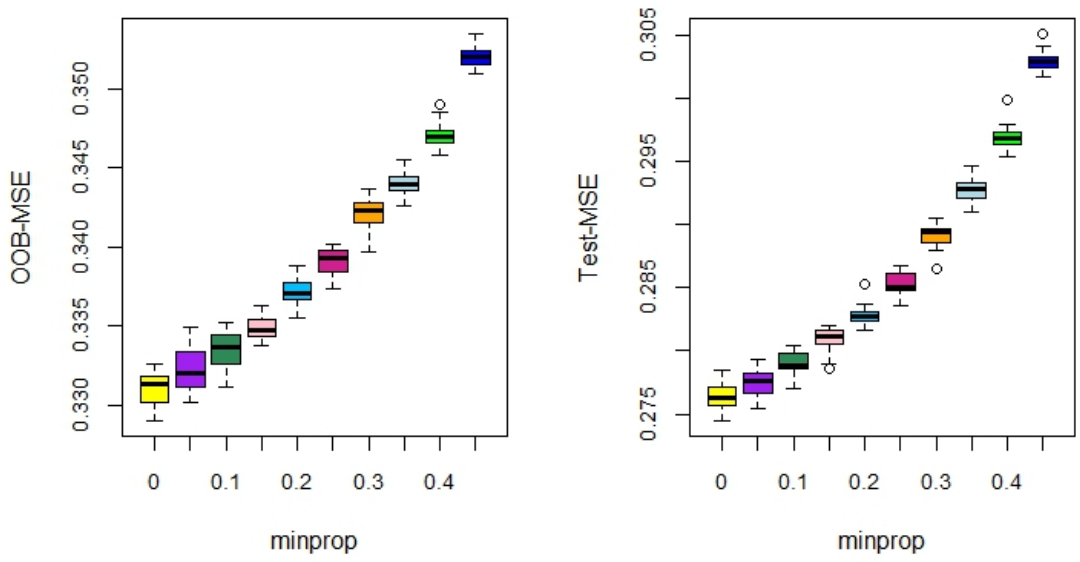


Abbildung 8.8: OOB- und Test-MSE für die NHANES-Daten in Abhängigkeit von dem Parameter `minprop` für die Splitting-Regel `"maxstat"`

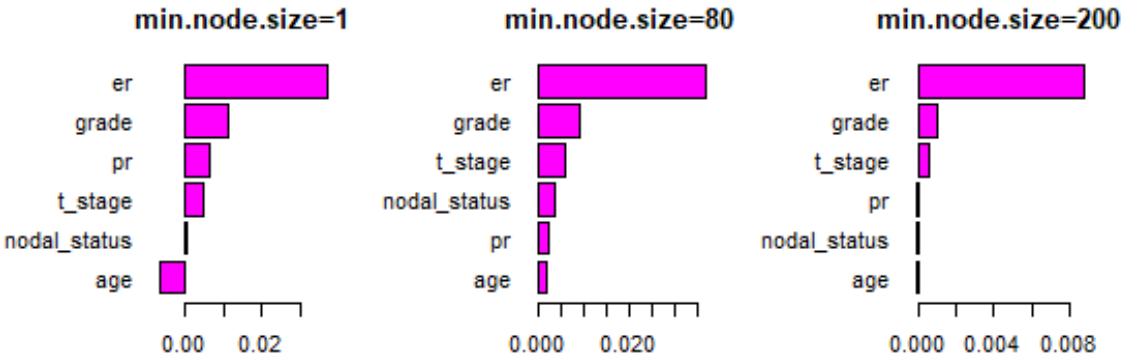


Abbildung 8.9: Rangordnungen der Variablenwichtigkeiten der Brustkrebsdaten für minimale Knotengrößen 1, 80 und 200 (gemessen mit der Permutationsmethode)

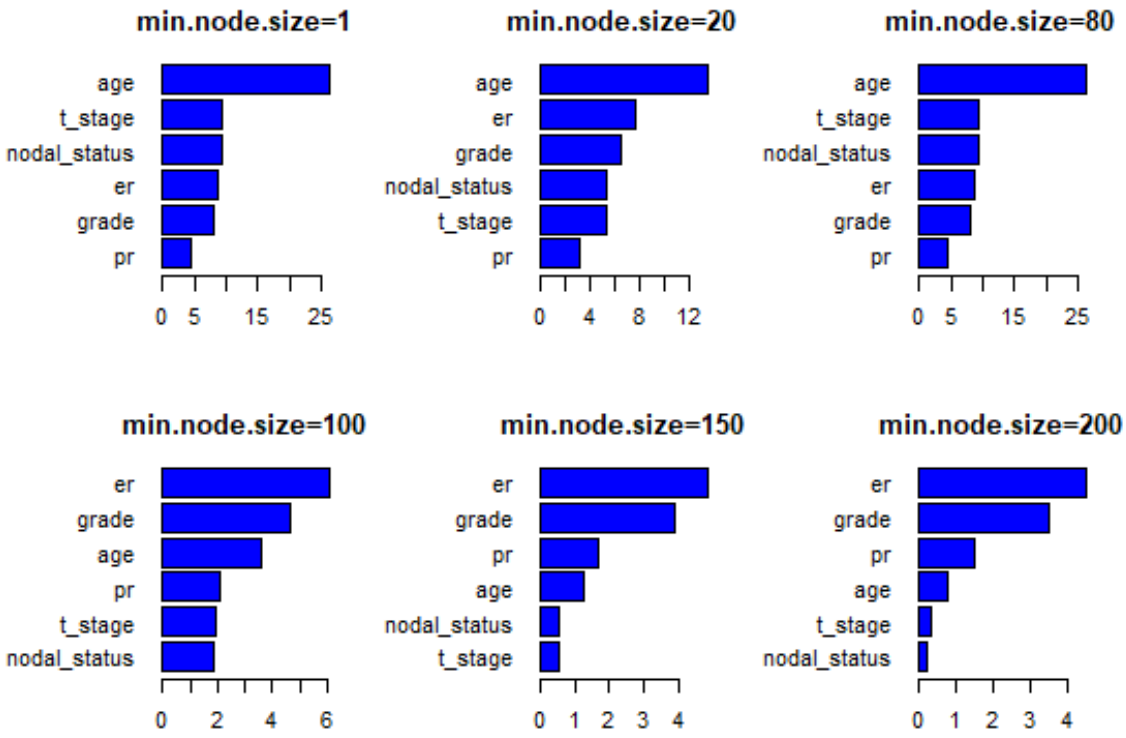


Abbildung 8.10: Rangordnungen der Variablenwichtigkeiten der Brustkrebsdaten für minimale Knotengrößen 1,20, 80, 100, 150 und 200 (gemessen mit der Unreinheitsmethode)

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

München, 05.03.2018

_____

Katrin Racic-Rachinsky