Unterschrift des Betreuers

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Random Forest Klassifikation bei unbalancierten Daten

Ausgeführt am Institut für
Stochastik und Wirtschaftsmathematik
der Technischen Universität Wien
unter der Anleitung von
Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser
durch

Sebastian Hackl
Rechte Wienzeile 233/13
1120 Wien

1. September 2015

IN TRIBUTE TO MIRALO

VIELEN DANK

In erster Linie möchte ich mich ganz herzlich bei Prof. Filzmoser dafür bedanken, dass er sich die Zeit nahm und es mir ermöglichte, diese Arbeit unter seiner Betreuung zu schreiben. Außerdem vermittelte er mir eine Anstellung, in der ich auch das Random Forest Verfahren anwenden konnte. An dieser Stelle auch ein Dankeschön an Frank L., der mich dort in der Arbeitswelt unterstützte.

Ein großes Dankeschön gilt natürlich allen meinen Korrekturlesern, angefangen bei meiner Mutter und meinem Betreuer Prof. Filzmoser, die viel Zeit dafür aufwendeten, dass meine Arbeit weitgehend fehlerfrei gelang, auch meinen Freunden Gerald und Bernhard, die mich immer wieder mit Tipps und Hilfestellungen unterstützten. Nicht zuletzt möchte ich noch Martin besonders erwähnen, den ich während meines gesamten Studiums mit zahlreichen Fragestellungen quälen durfte.

Natürlich gilt mein Dank meiner ganzen Familie, die mich sowohl finanziell aber auch bei all meinen Entscheidungen unterstützte.

Aghy, der sogar seine Dip-Arbeit mir gewidmet hat und all meinen Freunden, ehemaligen WG-Kollegen und der Tarock-Runde, die mich nicht nur während der Studienzeit sondern auch in der Freizeit ertragen mussten, ein herzliches Dankeschön.

Zusammenfassung

Das Random Forest Verfahren ist ein Klassifizierungs- bzw. Regressionsverfahren, welches bei der Anwendung in vielen Wissenschaftsgebieten viele Vorteile genießt. Einer dieser Vorteile ist der, dass der Random Forest rechensparsam ist und deswegen auch bei großen Datensätzen kein Problem darstellt. Auch kann dieses Verfahren gut mit dem Problem umgehen, wenn der Datensatz aus vielen Merkmalen aber nur wenigen Beobachtungen besteht.

Angefangen bei der Definition eines Entscheidungsbaumes, dem Baustein des Random Forest, werden in dieser Arbeit weitere grundlegende Definitionen angegeben, um dann den Algorithmus vorzustellen. Des Weiteren wird angeführt, wie sich bereits während der Trainingsphase die Merkmalswichtigkeiten, ein Schätzer für den Missklassifikationsfehler und ein Distanzmaß berechnen lassen.

Anschließend wird auf die Problematik hingewiesen, die auftritt, wenn man einen Random Forest auf einem unbalancierten Datensatz trainiert. Dazu werden zuerst die Methoden des "Over Sample" bzw. "Under Sample" vorgestellt. An zwei unterschiedlichen Datensätzen werden diese Methoden mit jeweils differenten Parameterwerten angewandt, um diese Ergebnisse dann gegenüberzustellen und zu analysieren. Daraus ist auch zu erkennen, welch wichtiger Faktor der Zufall in einem Randon Forest ist. Zuletzt wird in dieser Arbeit auch festgestellt, dass dieses Verfahren auch vom "Overfitting" Problem betroffen sein kann, gerade dann, wenn man die Bäume bis zur maximalen Größe wachsen lässt.

Inhaltsverzeichnis

Kapitel 1

Einführung

Seit einigen Jahren taucht immer häufiger der Begriff der künstlichen Intelligenz auf und wird z. B. in den Medien oft in Verbindung mit einem Roboter verwendet. Künstliche Intelligenz bedeutet, dass man einen Computer dahingehend trainiert bzw. programmiert, um ihn in die Lage zu versetzen, bei gegebenen Problemen richtige Entscheidungen zu treffen. Als Paradebeispiel dient oft eine nervengesteuerte Prothese. Diese soll Nervensignale eines Menschen messen um danach die gewünschte Bewegung zu tätigen. Auch die Spracherkennung wird oft als Beispiel angeführt, d. h. ein Computer oder Mobiltelefon versucht, aus den Frequenzen der gesprochenen Wörter diese wiederzugeben bzw. zu verarbeiten.

Es existieren mehrere Teilgebiete der künstlichen Intelligenz, eines davon ist z. B. das maschinelle Lernen. Dieses beschreibt ein automatisiertes Lernverfahren, das versucht, basierend auf einen Datensatz, Muster, Zusammenhänge und Strukturen zu erkennen. Diese Lerntechniken verwenden einen Teil des Datensatzes, den sogenannten Trainingsdatensatz, um damit ein Modell zu generieren und dieses anschließend mit den restlichen Daten, den sogenannten Testdaten, zu prüfen.

Maschinelles Lernen wird unterteilt in überwachtes und unüberwachtes Lernen. Mit überwachtem Lernen bezeichnet man Algorithmen, die anhand der Relationen zwischen Eingabe- und Zielwert im Datensatz lernen und mit Hilfe dieser Daten versuchen, ein Modell zu erstellen um bei neuen Daten bestmögliche Prognosen zu liefern.

Im Gegensatz dazu wird beim unüberwachten Lernen versucht, nur anhand der Ausgangsdaten die Zusammenhänge zu verstehen und entsprechende Antwortvariablen zu erzeugen.

Das in dieser Arbeit beschriebene Verfahren Random Forest (deutsch: Zufallwald) ist eine Methode des überwachten Lernens. Der Random-Forest-Algorithmus wurde im Jahr 2001 von Leo Breiman entwickelt und fand fortan große Beliebtheit in der Anwendung von Klassifizierungs- bzw. Regressionproblemen. Die grundlegende Idee dahinter ist die, dass man viele unterschiedliche Entscheidungsbäume kombiniert bzw. zusammenfügt, um nach der Mehrheit der Entscheidungsbäume zu klassifizieren. In dieser Arbeit wird

das Verfahren und einige Eigenschaften des Random Forest erläutert. Viele Klassifizierungsmethoden tendieren bei einem unbalancierten Datensatz in die Richtung, dass diese die Individuen vor allem der größeren Klasse korrekt klassifizieren und die Individuen der kleineren Klasse dabei oft fehlerhaft ist. Es wird ein Random Forest auf einem unbalancierten Datensatz aufgebaut und die Ergebnisse analysiert. Dazu werden auch noch Methoden vorgestellt, die bei einem unbalancierten Datensatz eine Verbesserung der Vorhersage bewirken.

Außerdem wird noch erklärt bzw. an einem Datensatz gezeigt, warum der Wert der Merkmalswichtigkeit teilweise über- bzw. unterschätzt wird. Im letzten Kapitel "Simulationen" werden, basierend auf dem Datensatz "Seismic pumps" mehrere Zufallswälder aufgebaut. Im R-Package "randomForest" sind hierfür viele nützliche und wichtige Funktionen implementiert. Diese werden erklärt und die erhaltenen Ergebnisse werden analysiert.

Zuerst ist es aber einmal sinnvoll, die Problemstellung und die Ziele zu deklarieren. Man verwendet hierfür einen Datensatz $\mathbb{X} := \{(X_1^q, y_1), ..., (X_N^q, y_N)\}$ mit $N \in \mathbb{N}$ unabhängig und identisch verteilten Zufallsvariablen und den dazugehörigen Antwortvariablen $y_i \in \mathbb{Y}$. Jede dieser Zufallsvariablen enthält q Beobachtungen bzw. Individuen. Diese Beobachtungen haben Merkmale, die unterschiedliche Skalenniveaus aufweisen können. Im Folgenden bezeichne mit x^j, $j = \{1, ..., q\}$, das $j - te$ Merkmal der Individuen. Die Merkmale können ordinal sein, das heißt, sie sind auf einem metrischen Raum messbar. Ordinale Merkmale sind beispielsweise Schulnoten, Körpergewicht, Temperatur, etc. Aber sie können auch kategorieller Natur sein, wie die Unterteilung von Geschlecht, Farbe, Geschmack, etc.

Der Antwortraum $\mathbb{Y}$ kann ebenfalls numerisch sein, beispielweise $\mathbb{Y} \in \mathbb{N}$ oder $\mathbb{Y} \in \mathbb{R}$, oder kategorisch wie z. B. die Einteilung von Blutdruckwerten $\mathbb{Y} = \{niedrig, normal, hoch\}$. Ziel ist es nun, anhand der Beobachtungen (X_i^q, y_i) eine Abbildung $f := \mathbb{R}^q \to \mathbb{Y}$: $f(X_i^q) = y_i$ zu konstruieren, die jeder Beobachtung eine Prognose der Antwortvariablen y_i zuweist. Um das Random-Forest-Verfahren beschreiben zu können, werden zuerst Entscheidungsbäume vorgestellt, da diese einen Random Forest aufbauen.

Kapitel 2

Entscheidungsbäume

Entscheidungsbäume dienen der übersichtlichen Darstellung von Entscheidungen und sind aufgrund ihrer sehr einfachen Interpretierbarkeit sehr populär. Zur Konstruktion eines Entscheidungsbaumes wird nur das a priori Wissen verwendet, das heißt, das Wissen, welches man allein aus dem Datensatz erhält. Aus einem Datensatz kann man etwa die Anzahl an Klassen, Klassenverteilungen etc. bestimmen. Meistens werden Entscheidungsbäume anhand des Top-Down-Prinzips erstellt. Das bedeutet, dass man ausgehend von der ersten getroffenen Entscheidung zu den nächsten Entscheidungen geleitet wird, bis man schlussendlich am Ziel angelangt ist. Diese Bäume stellen somit eine hierarchische Anordnung von Entscheidungen dar.

Ausgehend vom Stamm, an dem die erste Entscheidung zu treffen ist, verzweigt sich der Baum in weitere Äste, die zu den nächsten Knoten und Entscheidungen führen. Die Entscheidungskriterien werden immer anhand eines Merkmals gemacht. Somit können diese Kriterien numerisch sein, wie etwa $x^j \leq C$, oder kategorisch, beispielsweise $x^j \in \{gelb, blau, grün\}$ oder $x^j = männlich$.

Beginnend am Stamm, der alle Datenpunkte des Datensatzes beinhaltet, versucht man, Verzweigungskriterien zu finden, sodass man den Datensatz in die unterschiedlichen Klassen aufteilt. An jeder Verzweigung sind Entscheidungen zu treffen, anhand derer man dann zu den Folgeknoten gelangt, bis man schlussendlich beim Endknoten angelangt ist, der X_i der Klasse y_j zuweist. Ein "guter" Entscheidungsbaum wird unter anderem ein solcher sein, der in den Endknoten nur noch Elemente X_i enthält, die der selben Klasse angehören. Somit wird es ein Ziel sein, Entscheidungen zu finden, die die Klassen bestmöglich trennen.

In der Medizin wird oft ein Entscheidungsbaum zur Klassifizierung eines Patienten verwendet. Ein Arzt hat damit die Möglichkeit, einen Patienten anhand der Werte aus Geschlecht, Alter, Blutdruck, Alkoholkonsum etc. einer Risikogruppe zuzuordnen.

2.1 Binärer Entscheidungsbaum

Ein spezieller Entscheidungsbaum ist der sogenannte binäre Entscheidungsbaum. Man spricht von diesem, wenn man pro Knoten nur binäre Entscheidungen treffen kann, also wenn jeder Knoten nur zwei Folgeknoten besitzt. Eine weitere Besonderheit binärer Entscheidungsbäume ist die der einfachen graphischen Visualisierung, denn binäre Entscheidungen teilen den Merkmalsraum in Rechtecke auf. Natürlich kann man auch einen Entscheidungsbaum mit mehreren Entscheidungen pro Knoten visualisieren, allerdings ist dies nur mit mehrdimensionalen Graphiken möglich und somit nicht so einfach darstellbar wie bei binären Entscheidungsbäumen. In Abbildung 2.1 auf Seite 6 ist ein binärer Entscheidungsbaum und die räumliche Trennung anhand der Iris Daten (Liliengewächse) abgebildet. Diese Daten stammen aus dem R-Package "datasets", welche von Anderson (1935) gesammelt wurden. Der Datensatz beinhaltet drei unterschiedliche Schwertliliengewächse (Iris virginica, Iris versicolor, Iris setosa) mit Angaben über die Länge bzw. Breite ihres Kelchblattes und die Länge bzw. Breite des Blütenblattes.

Bezeichne mit $T \in \mathbb{N}$ die Anzahl der Knoten eines Entscheidungsbaumes und mit K_t, $t \in \{1, ..., T\}$ den t-ten Knotenpunkt und definiere $z_i := (X_i^q, y_i)$. Weiters benenne mit $\mathcal{D}_1 := \{z_1, ..., z_n\}$ jene Daten, die sich im Stamm K_1 befinden. Nach der ersten getroffenen Entscheidung wird der Datensatz in zwei disjunkte Teilmengen $\mathcal{D}_2$ und $\mathcal{D}_3$ aufgeteilt. $\mathcal{D}_2$ und $\mathcal{D}_3$ sind also jene Teilmengen vom Datensatz, welche sich im Knotenpunkt K_2 und K_3 befinden. An diesen Knotenpunkten werden weitere Entscheidungen getroffen, sodass man zu den Folgeknoten K_4, K_5, K_6 und K_7 gelangt, welche auch wieder disjunkte Teilmengen $\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6$ und $\mathcal{D}_7$ beinhalten. Die Knotenpunkte K_t und die dazugehörigen Datenpunkte $\mathcal{D}_t$ werden nach jeder Entscheidung von links nach rechts mit t fortführend durchnummeriert. Dies wird solange fortgesetzt, bis die relative Häufigkeit der Klassen in einem Knotenpunkt eine bestimmte Schranke unterschreitet.

K_t^L und K_t^R bzw. $\mathcal{D}_t^L$ und $\mathcal{D}_t^R$ stellen nun die jeweiligen Folgeknoten bzw. ihre Teilmengen dar, die daraus resultieren, wenn man $\mathcal{D}_t$ anhand einer Entscheidung unterteilt. Aufgrund der soeben beschriebenen Konstruktion eines Entscheidungsbaumes und der Eindeutigkeit jeder Entscheidung gilt sowohl $\mathcal{D}_t^L \cup \mathcal{D}_t^R = \mathcal{D}_t$ als auch $\mathcal{D}_t^L \cap \mathcal{D}_t^R = \{\emptyset\}$, $\forall t \in \{1, ..., T)$.

Es stellen sich nun folgende drei Fragen:

- Wie groß soll man einen Baum wachsen lassen um ein "gutes" Ergebnis zu bekommen?
- Welches Merkmal x^j ,$j \in \{1, .., q\}$, soll für eine Verzweigung in Betracht gezogen werden und an welcher Stelle soll man dann den Entscheidungspunkt setzen?
- Wie wird schlussendlich klassifiziert?

Die letzte Frage ist schnell beantwortet. Ist man im Endknoten angelangt, entscheidet man nach der Mehrheit der Klassen, die in diesem Knoten vorkommen.

Wenn man einen Baum bis zur maximalen Größe wachsen lässt, wenn sich somit in allen Blättern nur noch einzelne Datenpunkte befinden, besteht die Gefahr einer Überanpassung, da die Entscheidungen, die in der Nähe der Endknoten zu treffen sind, immer spezifischer sein werden. Im Gegensatz dazu besteht die Gefahr einer hohen Missklassifikationsrate, wenn man zu wenige Verzweigungen hat, weil in den Endknoten eine große Anzahl an Klassen vorkommen kann und somit eine eindeutige Klassifizierung nicht möglich ist. Eine Lösung dieser Problemstellung wird im Abschnitt 2.2.3 erörtert.

Welche Merkmale man für Verzweigungspunkte verwendet hängt davon ab, wie genau eine Entscheidung auf diesem Merkmal die Individuen nach Klassen trennen kann. Wie man in Abbildung 2.1 erkennen kann, wurde der erste Verzweigungspunkt beim Merkmal "Blütenlänge" an der Stelle 2,45 gesetzt. Dies hat zur Folge, dass die Klasse der Setosa Liliengewächse komplett von den restlichen zwei Klassen getrennt wurde und man anhand einer Entscheidung zumindest schon feststellen kann, ob ein Liliengewächs zur Familie der Setosa Gewächse gehört oder zu einer der beiden anderen Liliengewächse. Dazu wird ein Maß eingeführt, welches die Unreinheit eines Knotens misst. In diesem Fall soll dieses Maß einen Knoten nach der Anzahl an unterschiedlichen Klassen und auch deren Klassengröße (Anzahl der Individuen) bewerten.

2.2 Klassifikations- und Regressionsbäume

Der CART-Algorithmus (**C**lassification **a**nd **R**egression **T**rees) wird bei der Erstellung von Regressions- und Klassifikationsbäumen sehr oft verwendet. Er wurde von Breiman et al. (1984) im Jahre 1984 veröffentlicht.

Dieser Algorithmus erzeugt jeweils binäre Klassifikations- oder Regressionsbäume. Wenn nach einer Regression verlangt wird, muss Y numerisch sein. Im Gegensatz dazu, das heißt, wenn das Ziel eine Klassifizierung ist, muss Y kategorisch sein, also jedes y_i muss Element von einer Klasse $\mathbb{Y} := \{c_1, c_2, ..., c_M\}$, $M \in \mathbb{N}$ sein. Da diese Arbeit fast ausschließlich Klassifikationsprobleme behandelt, wird dieses Thema gesondert betrachtet. Man verfährt bei einer Regression größtenteils identisch, bei der Suche nach Verzweigungspunkten verwendet man allerdings eine andere Methode. Mehr dazu findet man am Ende dieses Abschnittes.

Bevor auf die zweite Frage des vorhergehenden Abschnittes eingegangen wird, also nach welchen Kriterien man einen Verzweigungspunkt setzt, sollte noch auf die Anzahl an Möglichkeiten, einen solchen zu setzen, verwiesen werden. Weil die Verzweigungspunkte immer auf den Merkmalen gesetzt werden, wird wieder zwischen ordinalen und kategorischen Merkmalen unterschieden.

Hat man eine Variable mit $k \in \mathbb{N}$ unterschiedlichen Merkmalen, welche ordinal angeordnet sind, hat man $(k-1)$ Möglichkeiten, einen Verzweigungspunkt

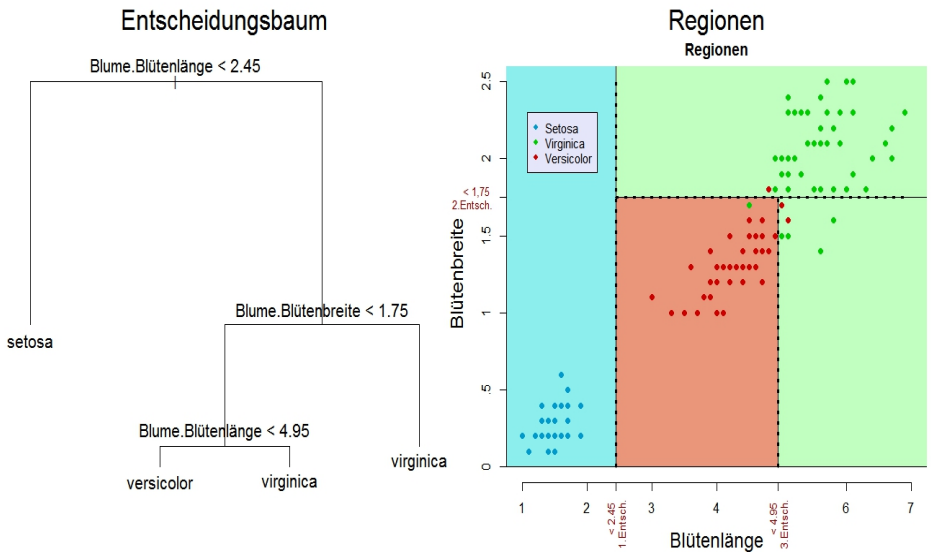

Abbildung 2.1: Ein binärer Entscheidungsbaum und die dazugehörigen räumlichen Trennungen anhand der Iris Daten

zu setzen. Wenn zum Beispiel als Merkmal die Schulnoten österreichischer Schüler herangezogen werden, bestehen für das Setzen von Verzweigungspunkten vier Möglichkeiten. Diese sind $x^j \leq 1, x^j \leq 2, x^j \leq 3$ und $x^j \leq 4$. Die Entscheidung $x^j \leq 5$ würde keinen Sinn machen, da jeder Schüler zumindest einen Fünfer hat, und somit wird diese Entscheidung den Datensatz nicht in die unterschiedlichen Klassen trennen.

Bei einer Variablen mit einer Anzahl von k unangeordneten Merkmalen hat man hingegen $(2^{k-1} - 1)$ mögliche Verzweigungspunkte. Hier sind mit unangeordneten Merkmalen solche gemeint, die nicht vergleichbar sind, wie zum Beispiel Farben, Formen oder Geschlecht. Unterteilt beispielsweise ein Merkmal die Elemente in unterschiedliche Farben $\{grün, gelb, rot, blau\}$, so gibt es genau sieben Möglichkeiten für einen Verzweigungspunkt, die wären:

$x^j = grün$, $x^j = gelb$, $x^j = rot$ und $x^j = blau$

$x^j \in \{grün, gelb\}$, $x^j \in \{grün, rot\}$ und $x^j \in \{grün, blau\}$

Man beachte, dass hier bereits alle Entscheidungen aufgelistet sind, denn die Unterteilung bei $x^j \in \{grün, gelb, rot\}$ ist die selbe wie die bei $x^j = blau$. Und die Frage nach $x^j \in \{grün, gelb, rot, blau\}$ würde nach der selben Argumentation wie beim zuvor erwähnten Beispiel keinen Sinn machen.

Um nun einen Verzweigungspunkt bewerten zu können, wird ein Maß für die Heterogenität oder Unreinheit eines Knotens eingeführt. Wie oben bereits gezeigt wurde, ist es sinnvoll, einen Verzweigungspunkt in einem Knoten so zu

setzen, dass unterschiedliche Klassen in den beiden Folgeknoten bestmöglich voneinander getrennt sind. Deswegen werden die Verzweigungspunkte immer anhand der Heterogenität in den beiden Folgeknoten bewertet, die durch diese Entscheidung erzeugt werden. Somit wird ein "perfekter" Entscheidungsbaum ein solcher sein, bei dem jeder der Endknoten nur noch Datenpunkte der gleichen Klasse enthält. In Summe sollte somit ein Entscheidungsbaum den Datensatz so gut wie möglich in die unterschiedlichen Klassen trennen. Im Ausgangspunkt bzw. im Stamm herrscht die größte Heterogenität, da noch alle Klassen vorhanden sind. Das Maß soll am Ausgangspunkt maximal sein und in weiterer Folge minimal werden, wenn fast alle Datenpunkte im Knoten zur selben Klasse gehören.

Des weiteren empfiehlt es sich, dass das Maß die Eigenschaft der Symmetrie aufweist. Mit anderen Worten, das Maß soll sich neutral gegenüber den Klassen verhalten und bei keiner Entscheidung irgendeine Klasse bevorzugen. Beispielsweise ist es sinnvoll, dass das Maß den gleichen Wert hat, wenn drei Klassen im Verhältnis von $(\frac{2}{4}, \frac{1}{4}, \frac{1}{4})$, $(\frac{1}{4}, \frac{2}{4}, \frac{1}{4})$ oder $(\frac{1}{4}, \frac{1}{4}, \frac{2}{4})$ vorkommen.

In Abbildung 2.2 auf Seite 11 sollen diese drei Eigenschaften des Maßes bei einer Klassengröße von zwei Klassen nochmals verdeutlicht werden.

Sei $\mathcal{D}_t := \{z_1, ..., z_{t_n}\}$ definiert wie zuvor als die Menge jener Datenpunkte, die sich im Knoten K_t befinden. Weiters sollen die Zielvariablen in M unterschiedliche Klassen eingeteilt werden, sodass also jedem X_i^q $i \in \{1, ..., n\}$ eindeutig ein y_m, $m \in \{1, ..., M\}$, zugeordnet ist. Sei im Folgenden $\mid \mathcal{D}_t \mid$ die Mächtigkeit der Datenpunkte im Knoten K_t und I die Indikatorfunktion. Weiters definiere:

$$p_{t,m}(\mathcal{D}_t) := \frac{\sum_{X_i^q \in \mathcal{D}_t} I(X_i^q = m)}{|\mathcal{D}_t|}$$

$p_{t,m}(\mathcal{D}_t)$ stellt die relative Häufigkeit der zur Klasse m gehörigen Individuen im Knoten K_t dar. Es gilt somit: $\sum_{m=1}^{\mathrm{M}} p_{t,m}(\mathcal{D}_t) = 1$. Man beachte, dass nur über m summiert wird und t festgehalten ist.

Ein Maß für die Heterogenität ist eine Funktion:

$$f_t := [0,1] \to \mathbb{R} : (p_{t,1}(\mathcal{D}_t), p_{t,2}(\mathcal{D}_t), ..., p_{t,M}(\mathcal{D}_t)) \to \mathbb{R} \tag{2.1}$$

mit folgenden Eigenschaften:

- f ist maximal bei $(\frac{1}{M}, \frac{1}{M}, \ldots, \frac{1}{M})$
- f ist minimal bei $(1, 0, 0, ..., 0), (0, 1, 0, ..., 0), ..., (0, 0, 0, ..., 1)$
- f ist symmetrisch bezüglich der Klassen.

Mit diesem Maß hat man ein Werkzeug, mit dem man in der Lage ist, einen Verzweigungspunkt zu bewerten. Dazu bezeichnet man das Heterogenitätsmaß im Knoten t mit:

$$\phi(K_t) := f_t\left(p_{t,1}(\mathcal{D}_t), p_{t,2}(\mathcal{D}_t), ..., p_{t,M}(\mathcal{D}_t)\right).$$

Um nun einen Verzweigungspunkt s zu bewerten, vergleicht man die Heterogenität im Knoten K_t mit der Heterogenität der beiden Folgeknoten $K_t^{R_s}$ und $K_t^{L_s}$, die man durch den Verzweigungspunkt s erhält.

Ein Wert für die Anpassungsgüte ist nun gegeben als:

$$\Delta\phi(s,t) := \phi(K_t) - \phi(K_t^{L_s}) - \phi(K_t^{R_s})$$

mit:

ϕ	Heterogenitätsmaß
s	Verzweigungspunkt
K_t	der Knotenpunkt t
$K_t^{L_s}, K_t^{R_s}$	die beiden Folgeknoten, wenn man K_t durch s unterteilt

Das Ziel wird es sein, $\Delta\phi(s,t)$ zu maximieren. Wie man anhand der Definition von $\Delta\phi(s,t)$ erkennen kann, ist das Maximieren gleichbedeutend mit

$$\max_s \Delta\phi(s,t) = \max_s \left(\phi(K_t) - \left[\phi(K_t^{L_s}) + \phi(K_t^{R_s})\right]\right) \Leftrightarrow \min_s (\phi(K_t^{L_s}) + \phi(K_t^{R_s}))$$

weil $\phi(K_t)$ nicht vom Verzweigungspunkt s abhängt. Dies bedeutet, dass man Verzweigungspunkte so wählt, dass das Heterogenitätsmaß in den beiden Folgeknoten $K_t^{L_s}$ und $K_t^{R_s}$ minimal wird und somit die Klassen einheitlicher bzw. homogener geteilt werden.

Wenn man keinen weiteren Verzweigungspunkt findet, der die Individuen in den Folgeknoten besser trennt, dann gilt:$\Delta\phi(s,t) = \phi(K_t)\ \forall s$. Dies kann folgende zwei Gründe haben:

- $min_s(\phi(K_t^{L_s}) + \phi(K_t^{R_s})) = 0$, d. h. die Individuen sind bereits perfekt in den Klassen getrennt
- $min_s(\phi(K_t^{L_s}) + \phi(K_t^{R_s})) \neq 0$ aber $\Delta\phi(s,t) = \phi(K_t)\ \forall s$. D. h. es gibt keinen Verzweigungspunkt, der verursacht, dass die Individuen in den Folgeknoten nach deren Klassen besser getrennt sind. Zum Beispiel tritt dieser Fall dann auf, wenn zwei Individuen unterschiedlicher Klasse exakt die gleichen Merkmale besitzen.

2.2.1 Maße der Unreinheit

Um die Notationen übersichtlicher zu machen, wird im Folgenden die Proportion der Klassen m im Knoten K_t als $p_{t,m} := p_{t,m}(\mathcal{D}_t)$ bezeichnet. Das im CART-Algorithmus verwendete Maß für die Unreinheit ist der Gini-Index. Breiman et al. (1984) führte in seiner Arbeit diesen Index ein und formulierte ihn folgendermaßen:

$$\phi_{Gini}(K_t) = \sum_{m=1}^{M}(\sum_{\substack{l=1 \\ l\neq m}}^{M} p_{t,m}p_{t,l}) = \sum_{m=1}^{M} p_{t,m}(1-p_{t,m}) = 1 - \sum_{m=1}^{M} p_{t,m}^2 \qquad (2.2)$$

Man beachte beim zweiten "="-Zeichen, dass $\sum_{m=1}^{M} p_{t,m} = 1$ ist und man deswegen die Doppelsumme zu einer Summe zusammenfügen darf.

Bei Wissen über die Klassenverteilung im Knotenpunkt t gibt $p_{t,m}^2$ die Wahrscheinlickeit an, dass, wenn man zufällig ein Element zieht und dieses Element einer Klasse zuweist, diese Zuweisung auch richtig ist. Hat man z. B. eine Klassenverteilung von $(1/2, 1/2)$, ist die Wahrscheinlichkeit $(\frac{1}{2})^2$, dass man Element 1 zieht und zur Klasse 1 zuordnet. Folglich ist nach zweimaligem Ziehen und zweimaliger Zuordnung die Wahrscheinlichkeit $1/2$, dass diese Zuordnung richtig ist. Andererseits wird bei einer Klassenverteilung von $(2/2, 0/2)$ jedes Element mit Wahrscheinlichkeit 1 der richtigen Klasse zugeordnet, weil man weiß, dass alle Elemente der ersten Klasse angehören.

Bildet man nun die Gegenwahrscheinlichkeit wie beim Gini Index (2.2), bedeutet dies die Wahrscheinlichkeit, dass bei bekannter Klassenverteilung nach zufälliger Ziehung und beliebiger Zuordnung alle Element falsch klassifiziert werden.

Das heißt, dass der Gini Index minimal wird, wenn sich im Knoten nur noch Individuen einer Klasse befinden. Wie anhand der zuvor angeführten Beispiele ersichtlich wird, erfüllt dieses Maß auch alle Eigenschaften eines Heterogenitätsmaßes (2.1).

Bewertet wird nun der Verzweigungspunkt, wenn man die Verbesserung der Reinheit in den Folgeknoten betrachtet, die diese Verzweigung nach sich zieht.

$$\Delta\phi_{Gini}(s,t) = \phi_{Gini}(K_t) - \left[\phi_{Gini}(K_t^{L_s}) + \phi_{Gini}(K_t^{R_s})\right] \qquad (2.3)$$

mit:

ϕ_{Gini}	Gini Index
s	Verzweigungspunkt
K_t	der Knotenpunkt t

$K_t^{L_s}, K_t^{R_s}$ die beiden Folgeknoten, wenn man K_t durch s unterteilt

Die beste Entscheidung $\dot{s}$ ist nun jene, welche $\Delta\phi_{Gini}(s,t)$ maximiert. Der CART Algorithmus setzt die Verzweigungspunkte gemäß dem Gini Index.

Bezeichne mit $\bar{K}_t$ einen Endknoten und mit $\overline{E} := \#\{\bar{K}\}$ die Anzahl der Endknoten des Baumes E . Wenn man die Unreinheiten aller Endknoten des Entscheidungsbaumes aufsummiert, erhält man einen Wert, der die Trennschärfe der Individuen nach ihren Klassen misst:

$$Q_{class}(E) := \sum_{i=1}^{\overline{E}} \phi_{Gini}(\bar{K}_t) \tag{2.4}$$

mit:

$\overline{E}$ Anzahl der Endknoten des Baumes E

$\bar{K}_i$ ein Endknoten des Baumes E

Vollständigerweise seien neben dem Gini noch folgende Unreinheitsmaße erwähnt:

- $Gini : \phi_{Gini}(K_t) := 1 - \sum_{i=1}^{M} p_{t,i}^2$
- $Entropie : \phi_{entropie}(K_t) := -p_{t,1}\log(p_{t,1}) - .... - p_{t,M}\log(p_{t,M})$
- $Missklassifikationsfehler : \phi_{Miss}(K_t) := 1 - \max_{i\in\{1,...,M\}}\{p_{t,i}\}$

In Abbildung 2.2 sind die drei Maße mit jeweils zwei Klassen abgebildet. Wie man sieht, erfüllen alle drei Maße die Voraussetzungen für eine Bewertung der Heterogenität. Sie weisen bei einem Gruppenverhältnis von 50% zu 50% im Knoten den größten Wert auf und den niedrigsten, wenn der Knoten zu 100% aus nur einer Klasse besteht.

Regression

Bei einer Regression sind die Antwortvariablen keine Klassen mehr, sondern numerische Werte, zum Beispiel $y_i \in \mathbb{R}$. Deswegen kann man auch keine der vorher erwähnten Maße für einen Verzweigungspunkt verwenden, weil man dadurch keine Heterogenität messen kann. Wie bei einem Klassifikationsbaum wird auch bei einer Regression der Merkmalsraum in Rechtecke unterteilt, siehe Abbildung 2.1 auf Seite 6.

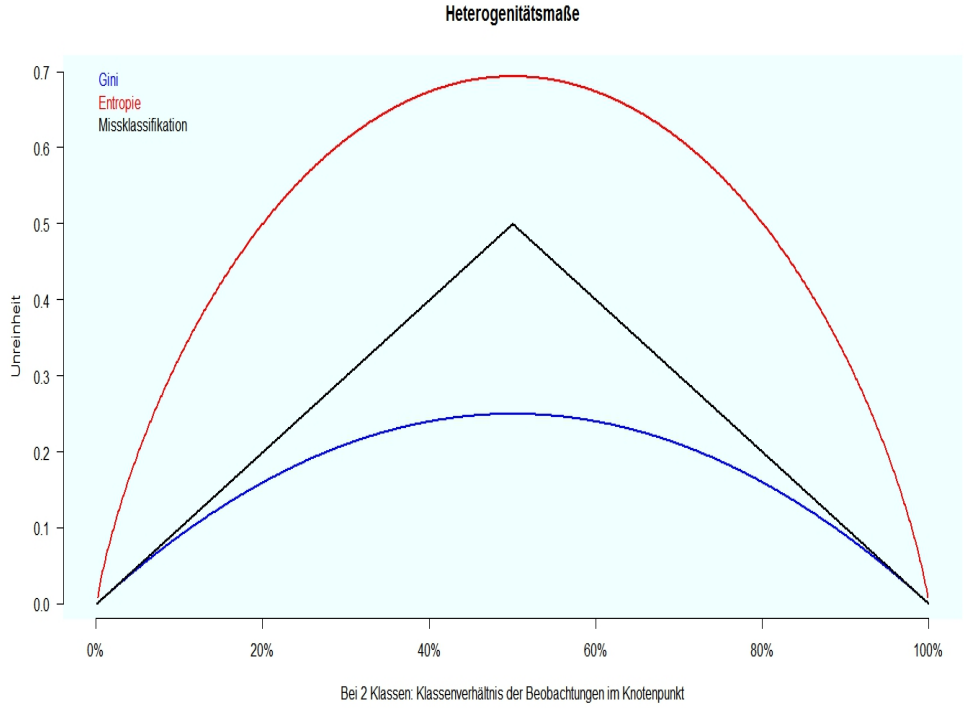

Abbildung 2.2: Die drei vorgestellten Unreinheitsmaße

Bezeichne die Endknoten eines Baumes E mit $\bar{K}_t$ und mit $N_{\bar{K}_t} := \#\{X_i^q \in \bar{K}_t\}$ die Mächtigkeit der Individuen des Endknotens $\bar{K}_t$. Wenn ein Individuum X_i^q in den Endknoten $\bar{K}_t$ fällt, wird diesem eine Zahl c_t zugeordnet mit $f(X_i^q) = c_t$. Wobei c_t das arithmetische Mittel

$$c_t = \frac{1}{N_{K_t}} \sum_{(X_j, y_j) \in K_t} y_j \tag{2.5}$$

aller in $\bar{K}_t$ liegenden Daten ist.

Man bezeichnet weiters mit $N_{K_t}^L$ und $N_{K_t}^R$ die Mächtigkeit der Individuen in den beiden Folgeknoten $K_t^{L_s}$ und $K_t^{R_s}$, wenn man K_t durch s unterteilt. $y_{t,i}^{L_s}$ bzw. $y_{t,i}^{R_s}$ sind die Zielwerte jener X_i^q, die nach einer Unterteilung in den linken bzw. in den rechten Folgeknoten fallen und c_1 bzw. c_2 sind die Mittelwerte der $y_{t,i}^{L_s}$ bzw. $y_{t,i}^{R_s}$.

Der Ansatz, einen guten Verzweigungspunkt zu finden, ist der, wenn man den mittleren quadratischen Fehler $\sum_{i=1}^{N_{K_t}^L}(y_{t,i}^{L_s} - c_1)^2$ und $\sum_{i=1}^{N_{K_t}^R}(y_{t,i}^{R_s} - c_2)^2$ in den beiden Folgeknoten betrachtet. Ein idealer Verzweigungspunkt sollte diesen Fehler minimieren. Damit ist die Lösung c_1 und c_2 genau das arithmetische Mittel der $y_{t,i}^{L_s}$ und $y_{t,i}^{R_s}$.

Ein Maß für die Unreinheit im Knoten K_t ist nun definiert als:

$$\phi_{reg}(K_t) := \frac{1}{N_{K_t}} \sum_{i=1}^{N_{K_t}} (y_i - c)^2 \tag{2.6}$$

mit:

N_{K_t}	Anzahl der Individuen im Knoten K_t
y_i	Zielwert des Individuums $X_i^q \in K_t$
c	arithmetisches Mittel der Zielwerte, vgl: (2.5)
K_t	der Knotenpunkt t

Der beste Verzweigungspunkt s im Knoten K_t ist nun der, wenn

$$\Delta\phi(K_t, s) := \phi(K_t) - \left[\phi(K_t^{L_s}) + \phi(K_t^{R_s})\right] \tag{2.7}$$

maximal ist.

Das Ziel ist es somit, jenen Verzweigungspunkt s zu finden, der den mittleren quadratischen Fehler in den Folgeknoten größtmöglich minimiert.

Wenn man mit $\overline{E} := \#\{\bar{K}\}$ die Anzahl der Endknoten eines Baumes bezeichnet, kann man die Summe der Fehlerquadrate des Baumes E wie folgt angeben:

$$Q_{reg}(E) := \sum_{i=1}^{\overline{E}} \phi_{reg}(\bar{K}_i) \tag{2.8}$$

mit:

$\overline{E}$	Anzahl der Endknoten des Baumes E
$\bar{K}_i$	ein Endknoten des Baumes E

2.2.2 Der CART-Algorithmus

Da der Stamm eines Entscheidungsbaumes aus dem gesamten Datensatz besteht und somit auch alle Klassen enthält, hat er die höchste Heterogenität. Ziel ist es nun, die Verzweigungspunkte so zu setzen, dass die Klassen bestmöglich getrennt werden. Das dazu benötigte Werkzeug wurde im Abschnitt zuvor bereits vorgestellt. Es sei nochmals daran erinnert, dass der CART-Algorithmus nur binäre Entscheidungsbäume erzeugt. Im Folgenden ist mit $\mathcal{D}_t \subseteq X$ die Menge jener Individuen des Datensatzes definiert, die der Knoten

t des Baumes beinhaltet. Bezeichne weiters mit M_j die Anzahl der Möglichkeiten, einen Entscheidungspunkt am Merkmal x^j zu setzen.

Die im folgenden Algorithmus vorgestellten Punkte $(a) - (e)$ bewirken, dass sich jeder Knotenpunkt (mit Ausnahme der Endknoten) in den beiden Folgeknoten unterteilt. Das heißt, dass sich jeder Knotenpunkt des Baumes bzgl. des verwendeten Kriteriums bestmöglich verzweigt.

Die in den Punkten 2 und 3 des Algorithmus definierten n_h stellen jeweils die im Schritt h noch zu unterteilenden Knotenpunkte dar. Folglich endet das Wachstum des Baumes genau dann bei Schritt h, wenn gilt: $n_h = 0$. Die Anzahl der Schritte gibt an, wie viele Entscheidungen maximal zu treffen sind, um am Endknoten anzukommen. Wenn man Abbildung 2.1 betrachtet, sind höchstens drei Entscheidungen zu treffen, um zu einem Endknoten zu gelangen. Weiters dienen sie auch zur Initialisierung dieser Knotenpunkte.

Beginnend mit dem ersten Knoten K_1 (dem Stamm des Baumes) wird unter Beachtung des Gini Indexes wie folgt vorgegangen:

Setze $h = 1$, $n_0 = 0$ und $n_1 = 1$.

1. Schritt h: Setze $t = 1 + \sum_{l=0}^{h-1} n_l$

   (a) Berechne zu jedem der q Merkmale für die $\mathcal{D}_t$ des Knotens K_t jenen Verzweigungspunkt $\bar{s}_j$ mit
   $\bar{s}_j := \max_{i \in \{1,\ldots,M_j\}} \Delta\phi(s_{j,i}, t)$, $\forall j \in \{1, 2, \ldots, q\}$.

   (b) Setze $s_{max} := max\{\bar{s}_1, \ldots, \bar{s}_q\}$

   (c) Gilt: $\Delta\phi(s_{max}, t) < \phi(K_t)$:

      i. Wähle jenes Merkmal x^j als Entscheidungsmerkmal, für das gilt: $s_j = s_{max}$.

      ii. Unterteile die Elemente $\mathcal{D}_t$ des Knotens K_t anhand dieses Punktes.

   (d) Gilt: $\Delta\phi(s_{max}, t) = \phi(K_t)$:

      i. Dann stellt der Knoten K_t einen Endknoten des Baumes dar und wird im Folgenden nicht mehr unterteilt.

   (e) Setze $t = t + 1$ und wiederhole die Schritte (a)-(e) so lange bis gilt: $t > \sum_{l=1}^{h} n_l$.
   (Wenn t größer ist als die Summe der n_l, heißt das, dass unter Schritt j bereits alle vorhandenen Knotenpunkte fertig verzweigt wurden)

2. Bezeichne mit $\bar{n}_h$ die Anzahl der unter den Punkten $(a) - (e)$ neu entstandenen Endknoten und setze $n_{h+1} = 2(n_h - \bar{n}_h)$.

3. Initialisiere die unter den Punkten $(a) - (e)$ entstandenen Folgeknoten und deren neue Teilmengen mit $t := \{1+ \sum_{l=1}^{h} n_l, ..., n_{h+1}+ \sum_{l=1}^{h} n_l\}$ von links nach rechts mit K_t und setze $h = h + 1$

4. Führe die Schritte 1 - 4 so lange fort, bis in keinem Knoten mehr eine Verbesserung im Sinne des verwendeten Kriteriums erzielbar ist.

Wie man sieht, werden nach diesem Algorithmus sehr große Bäume mit vielen Endknoten entstehen, da so lange fortgefahren wird, bis keine Besserung mehr erreicht werden kann. Das heißt, es wird so lange fortgefahren, bis $Q_{class}(E)$ bzw. $Q_{reg}(E)$ minmal sind. (Vgl: (2.4) und (2.8))

Wenn sich in einem Datensatz die Individuen anhand ihrer Merkmale nach Klassen perfekt trennen lassen, wird ein Entscheidungsbaum ein zufriedenstellendes Ergebnis liefern, wie es z. B. in Abbildung 2.1 auf Seite 6 ersichtlich ist. Dort konnte man schon nach der ersten Entscheidung bezüglich der Blütenlänge die Individuen der Klasse "Setosa" von den restlichen beiden Klassen trennen. Zwei weitere Entscheidungen wurden dann noch benötigt, bis die Verringerung der Unreinheit eine gewisse Schranke unterschritten hat.

Wie man aber feststellen kann, wurde dieser Baum nicht mittels CART-Algorithmus erzeugt, denn man würde noch mindestens drei weitere Entscheidungen benötigen, damit alle Endknoten bezüglich des Unreinheitsmaßes als "rein" gelten.

Wenn sich der Datensatz aber nicht so "schön" in die unterschiedlichen Klassen trennen lässt, kann es sein, dass der CART-Algorithmus große Entscheidungsbäume konstruiert. Wenn ein Baum groß wird, d. h. wenn viele Entscheidungen zu treffen sind, um zum Endknoten zu gelangen, kann das Problem einer Überanpassung auftreten. Denn je mehr Verzweigungen ein Baum aufweist, desto öfter wird auch der Trainingsdatensatz getrennt. Dies hat zur Folge, dass in den Knoten eine immer kleinere Menge an Individuen unterschiedlicher Klassen vorkommt.

Je mehr Entscheidungen nötig sind, die Individuen in Klassen zu trennen, desto spezifischer sind diese dann aber auch. Beispielsweise kann es in der letzten Entscheidung vorkommen, dass diese nur noch zwei Individuen zweier unterschiedlicher Klassen trennt und dass somit der Endknoten aus jeweils einem Datenpunkt besteht. Folglich wird eine geringe Veränderung an den Merkmalen das Ergebnis stark beeinflussen.

Man bezeichnet mit einem instabilen Klassifikator bzw. mit einem schwachen Lerner einen Klassifikator, bei dem kleine Veränderungen der Daten eine große Veränderung auf das Ergebnis bewirken. Diese Klassifikatoren, insbesondere auch Entscheidungsbäume, weisen deswegen auch eine große Varianz auf.

2.2.3 Pruning

Man kann dem Problem der Überanpassung eines Entscheidungsbaumes entgegenwirken, indem man den Baum nicht so lange wachsen lässt bzw. den

Baum stutzt (engl: to prune). Folglich sind in den Endknoten die Individuen nicht mehr nach ihren Klassen perfekt getrennt. Wenn man allerdings einen Baum zu kurz wachsen lässt und dieser nur eine geringe Anzahl an Knoten aufweist, kann es vorkommen, dass wichtige Zusammenhänge des Datensatzes verloren gehen. Hier stellt sich nun die Frage, wie groß man einen Baum wachsen lassen soll, damit dieser ein zufriedenstellendes Ergebnis liefert.

Eine Möglichkeit ist die, dass das Wachstum eines Baumes gestoppt wird, wenn eine gewisse Anzahl an Knoten erreicht wurde. Man kann mit einigen Versuchen und mit jeweils unterschiedlicher Anzahl an Knoten testen, welcher dieser Bäume das beste Ergebnis liefert. Dies ist allerdings mit großem Aufwand verbunden und das Ergebnis ist sehr stark vom Test- bzw. Trainingsdatensatz abhängig.

Eine weitere Methode besteht darin, dass man das Wachstum stoppt, wenn die Verringerung der Unreinheit eines Knotens eine gewisse Schranke C nicht mehr unterschreitet. Wenn gilt: $\Delta\phi(K_t, s) < C$, $\forall s$, wird dieser Knoten K_t somit nicht mehr weiter unterteilt und der Knoten K_t stellt den Endknoten des Baumes dar. Wenn bei einem Knotenpunkt ein Verzweigungspunkt wenig zur Verbesserung der Unreinheit beiträgt, heißt das allerdings nicht, dass es in den Folgeknoten keine Verzweigung gibt, die die Unreinheit nicht deutlich verringert. Somit ist die Idee, eine untere Schranke für die Verringerung der Unreinheit einzuführen, auch nicht ideal.

Cost-Complexity Pruning

Das Cost-Complexity Pruning (vgl:Breiman et al. (1984, S. 66)) ist eine oft angewandte Methode, um Entscheidungsbäume zu kürzen. Man erstellt zuerst einen maximal ausgewachsenen Entscheidungsbaum E_{max}. Danach wird E_{max} schrittweise verkürzt, indem man Äste des Baumes abscheidet. Somit ist der gestutzte Baum E im Ausgangsbaum E_{max} enthalten. Bezeichne mit $E \prec E_{max}$ den Teilbaum vom E_{max}, den man erhält, wenn man Äste von E_{max} abschneidet. Der Baum wird so lange gestutzt, bis er nur noch zwei Äste besitzt. Bezeichne die Anzahl der Endknoten des Baumes E mit $\bar{E} \in \mathbb{N}$ und definiere weiters mit $p(\bar{K}_j)$, $j \in \{1, ..., \bar{E}\}$ die Wahrscheinlichkeit, dass ein Individuum in den Endknoten $\bar{K}_j$ fällt. Das heißt, wenn mit $N_{\bar{K}_j}$die Anzahl der Individuen im Endnoten $\bar{K}_j$ bezeichnet wird, ist p definiert als:

$$p(\bar{K}_j) := \tfrac{1}{N} \cdot N_{\bar{K}_j}$$

Definiere für ein Klassifizierungsproblem folgende Funktion:

$$R(E) := \sum_{j=1}^{\bar{E}} p(\bar{K}_j) \cdot \phi(\bar{K}_j)$$

mit:

E	ein Entscheidungsbaum

ϕ	ein Maß für die Unreinheit, (vgl: Gini, Entropie, Missklassifikationsfehler)
p	Wahrscheinlichkeit, dass ein Element $X \in \mathbb{X}$ in den Endknoten $\bar{K}_j$ fällt
$\bar{E}$	Anzahl der Endknoten des Entscheidungsbaumes E

Definiere für ein Regressionsproblem folgende Funktion:

$$R(E) := \sum_{j=1}^{\bar{E}} N_{\bar{K}_j} \cdot \phi_{reg}(\bar{K}_j)$$

E	ein Entscheidungsbaum
$\bar{E}$	Anzahl der Endknoten des Entscheidungsbaumes E
$\bar{K}_j$	Endknoten des Entscheidungbaumes
ϕ_{reg}	Maß für die Unreinheit, vgl. (2.6)

$R(E)$ stellt die Güte eines Entscheidungsbaumes E dar. Je mehr Endknoten ein Baum hat, desto kleiner wird auch der Wert von $R(E)$. Denn je höher die Anzahl ist, umso kleiner ist auch das Unreinheitsmaß ϕ und auch $p(\bar{K}_j)$ bzw. $N_{\bar{K}_j}$.

Definiere nun das Cost-Complexity Kriterium (vgl: Breiman et al. (1984, S 66):

$$R_\alpha(E) = R(E) + \alpha\bar{E} \tag{2.9}$$

Den Parameter $\alpha \geq 0$ nennt man den Cost-Complexity Parameter. Dieser Parameter ist ein Strafterm für die Anzahl der Endknoten. $R_\alpha(E)$ stellt eine Verbindung der Güte des Entscheidungsbaumes zur Anzahl der Endknoten her. Beispielsweise bei $\alpha = 0$ (d.h. es gibt keinen Strafterm für die Endknoten) ist die Lösung ein ausgewachsener Baum. Bei einem Klassifizierungsproblem sind somit bei einem $\alpha = 0$ nur noch Individuen einer Klasse in den Endknoten eines Entscheidungsbaumes. Je größer α ist, desto weniger Äste wird E haben. Bei festgehaltenem α ist der beste gestutzte Baum $\tilde{E}$ der, für den gilt:

$$R_\alpha(\tilde{E}) = \min_{E \preceq E_{max}} (R_\alpha(E)) \tag{2.10}$$

Um unter allen α den am besten gekürzten Baum zu finden, wird die weakest link pruning Methode angewandt. In der Folge wird die Idee kurz beschrieben.

Bei gegebenem α bezeichne mit $\tilde{E}_\alpha$ jenen Baum, der das Minimierungsproblem (2.10) löst. Zuerst setzt man $\alpha_1 = 0$. Wenn man $\alpha_2 > \alpha_1$ nur geringfügig erhöht, wird $R_{\alpha_2}(\tilde{E}) = R_{\alpha_1}(E_{max})$ sein, weil die Endknoten dadurch auch nur minimal bestraft werden.

Wenn man den Strafterm für die Anzahl der Endknoten weiter erhöht, gibt es ein $\alpha_2 > \alpha_1$ mit $R_{\alpha_2}(\tilde{E}) \neq R_{\alpha_1}(E_{max})$ und $\tilde{E}_{\alpha_2} \prec E_{max}$. Dieses α_2 existiert auf jeden Fall, da man den Strafterm so lange erhöhen kann, bis schlussendlich nur ein Baum mit nur einem Endknoten übrig bleibt (vgl: (2.9)).

Anschließend bestimmt man die weiteren Folgeglieder $\alpha_3, \alpha_4, ..., \alpha_{max}$ inklusive ihrer $\tilde{E}_{\alpha_i}$. Die Anzahl der Folgeglieder ist endlich (vgl. Breiman et al. (1984, S. 70). Nun kann man mittels Kreuzvalidierung testen, welcher dieser Bäume $\tilde{E}_{\alpha_1}, \tilde{E}_{\alpha_2}, ..., \tilde{E}_{\alpha_{max}}$ das beste Ergebnis liefert.

Kapitel 3

Random Forest

Random Forest stellt den Haupteil dieser Arbeit dar. Bevor aber ein Random Forest (Zufallswald) definiert wird, werden noch grundlegende Definitionen angeführt. Anhand dieser wird erklärt, wann und warum es besser ist, anstatt der Verwendung nur eines Baumes gleich einen ganzen Wald für eine Klassifikation bzw. Regression zu nutzen. In der Folge wird auch vermittelt, wofür man überhaupt einen "Zufall" benötigt und es werden auch jene zwei Methoden vorgestellt, die zur Generierung eines "Zufallswaldes" verwendet werden. Nach der Definition des Random Forest sind Vor- und Nachteile dieser Methode angegeben und es werden einige grundlegende Eigenschaften erörtert und beschrieben, welche weiteren Ergebnisse ein Random Forest noch liefern kann.

Am Ende dieses Kapitels wird noch die Güte des Ergebnisses getestet, welches ein Random Forest, der auf einem unbalancierten Datensatz trainiert wurde, liefert. Weiters wird noch erörtert, in welchen Fällen das Maß der Merkmalswichtigkeit eines Random Forest kritisch betrachtet werden soll. Außerdem wird noch gezeigt, welchen Einfluss fehlende Werte eines Datensatzes auf das Ergebnis des Random Forest haben.

3.1 Grundlegende Definitionen

3.1.1 Bagging

Breiman (1996) entwickelte die Methode des Bagging (engl: Bootstrap aggregating). Mit Bagging kann man durch eine Kombination mehrerer schwacher Lerner einen starken Lerner erhalten. Ziel ist es, mit dieser Methode die Vorhersagekraft zu verbessern, indem man die Prognose aller schwachen Lerner kombiniert wie z. B. bei einer Regression das arithmetische Mittel aller Prognosen. Es werden die Vorhersagemodelle bzw. schwachen Lerner jeweils basierend auf einen unterschiedlichen Datensatz konstruiert, sodass diese Modelle unabhängig voneinander sind. Diese Unabhängigkeit stellt einen wichtigen Faktor dar, denn es macht keinen Sinn, wenn man Entscheidungsbäume

mit exakt gleicher Gestalt kombiniert. Der Grund dafür ist der, dass das arithmetische Mittel aller Prognosen das gleiche Ergebnis liefert, wie bei der Verwendung nur eines einzelnen Baumes.

Um unterschiedliche Datensätze zu erlangen, wird die Methode des Bootstrapping angewandt. Das heißt, bevor man ein Vorhersagemodell auf dem Datensatz $\mathbb{X} := \{(X_1^q, y_1), ..., (X_N^q, y_N)\}$ modelliert, werden aus $\mathbb{X}$ zufällig und mit Zurücklegen $n \in \mathbb{N}$ Elemente gezogen. Das Modell wird dann anhand dieser n-elementigen Menge konstruiert. Diese Menge wird auch als der Trainingsdatensatz bzw. als die Lernmenge des Klassifikators bezeichnet. Die Menge, die nicht für die Konstruktion des Klassifikators verwendet wird, bezeichnet man als den Testdatensatz. Bezeichne im Folgenden mit $\mathcal{L}_i \subset \mathbb{X}$ die Menge der Datenpunkte, die für das i-te Vorhersagemodell gezogen worden sind.

Der Bagging-Algorithmus:

- Zuerst werden B Stichproben $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_B$ jeweils mit Größe n mittels Ziehen **mit Zurücklegen** aus dem Datensatz $\mathbb{X}$ konstruiert.
- Auf jede dieser Stichproben modelliert man ein Vorhersagemodell f_b mit $b \in \{1, ...B\}$.
- Füge alle Ergebnisse der B Vorhersagemodelle f_b zusammen.

Bei einer Klassifikation wird ein X_i nach der Mehrheit der Vorhersagewerte aller f_b, $b \in \{1, ...B\}$ prognostiziert:

$$f_{bag}(X_i) := arg \max_{j \in \{1,...M\}} \sum_{b=1}^{B} I\left(f_b(X_i) = c_j\right)$$

mit:

c_j	Klasse j, $j \in \{1, ...M\}$
I	Indikatorfunktion

Bei einer Regression wird schlussendlich folgendermaßen prognostiziert:

$$f_{bag}(X_i) := \frac{1}{B} \sum_{b=1}^{B} f_b(X_i)$$

Bagging muss nicht immer zu einer Besserung der Vorhersagekraft führen. Angenommen es gibt zwei Klassen zu prognostizieren und jeder der unabhängigen Klassifikatoren weist die selbe Treffergenauigkeit p auf. In diesem Fall liefert die Bagging Methode ein richtiges Ergebnis, wenn sich mehr als die Hälfte der Klassifikatoren für die richtige Klasse entscheiden. Somit ist die Wahrscheinlichkeit für eine richtige Klassifikation binomialverteilt:

$$\mathbb{E}\left(f_{bag}(X_i) = y_i\right) = \sum_{i=\lceil B/2 \rfloor + 1}^{B} \binom{B}{i} p^i \left(1-p\right)^{B-i}$$

mit:

B	Anzahl an Klassifikatoren
y_i	die wahre Klasse
p	die Wahrscheinlichkeit, dass ein Klassifikator f_i richtig klassifiziert

Hastie et al. (2009, S. 286) führte z. B. folgendes 2-Klassen-Beispiel an, bei dem ein einziger Klassifikator eine bessere Treffergenauigkeit liefert als die Bagging-Methode. Wenn die Treffergenauigkeit aller Klassifikatoren bei 40% liegt, wird die Mehrheit falsch prognostizieren und folglich wird die Bagging-Methode die falsche Klasse wählen. Allerdings, wenn man nur einen Klassifikator prognostizieren lässt, wird dieser nur zu 60% der Fälle falsch liegen.

Zum Abschluss wird an dieser Stelle noch eine weitere Methode erwähnt: das Subbagging. Beim Subbagging wird größtenteils analog vorgegangen wie beim Bagging-Algorithmus 3.1.1, allerdings werden unter dem ersten Punkt die Elemente $X_i \in \mathbb{X}$ für die Stichproben $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_B$ **ohne Zurücklegen** gezogen.

3.1.2 Varianz und Bias

Mit dem Bias des Lernalgorithmus f_{bag} bezeichnet man den erwarteten Fehler, der begangen wird, wenn man den Lernalgorithmus auf eine Folge von Trainingsmengen $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_B$ trainiert. Im Folgenden konstruiert man mittels Bagging eine Folge von Stichproben $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_B$ und eine Folge von Regressionsbäumen $f_1, f_2, ..., f_B$.

Der Erwartungswert einer fixen Beobachtung $X_i^q \in \mathbb{X}$ ist definiert als der Limes vom Mittelwert der Summe aller Ergebnisse der Regressionsbäume (vgl. Dietterich and Kong (1995)):

$$\bar{f}(X_i^q) := \lim_{B \to \infty} \frac{1}{B} \sum_{b=1}^{B} f_b(X_i^q)$$

Der statistische Bias von f_{bag} bei X_i^q ist definiert als die Abweichung von $\bar{f}$ vom wahren Wert y_i:

$$Bias\ \left(f_{bag}(X_i^q)\right) := \bar{f}(X_i^q) - y_i \tag{3.1}$$

Die Varianz von f_{bag} ist weiters definiert als der Erwartungswert für die quadratische Abweichung einer bestimmten Hypothese $f_j(X_i^q)$ und der gemittelten Hypothese $\bar{f}(X_i^q)$:

$$Var\left(f_{bag}(X_i^q)\right) := \mathbb{E}\left[(f_b(X_i^q) - \overline{f}(X_i^q))^2\right] \tag{3.2}$$

Geman et al. (1992), Bienenstock und Doursat haben gezeigt, dass sich der durchschnittliche Fehler eines Lernalgorithmus an einem Punkt X_i^q als die Summe der Varianz und des Bias zusammensetzt:

$$Err(f_{bag}(X_i^q) = Bias\left(f_{bag}(X_i^q)\right)^2 + Var\left(f_{bag}(X_i^q)\right) \tag{3.3}$$

Bei einer Klassifikation kann man obige Definitionen nicht anwenden, weil die Zielfunktion und die Zielvariablen keine messbaren Werte mehr sind, sondern Klassen darstellen. Man umgeht dieses Problem, indem man die Wahrscheinlichkeit betrachtet, dass die Klassifikatoren die Individuen der richtigen Klasse zuweisen.

Man konstruiert mittels Bagging eine Folge von Stichproben $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_B$ und eine Folge von Klassifikatoren $f_1, f_2, ..., f_3$. Bezeichne mit $\hat{p}_b$, $b \in \{1, ..., B\}$, folgende Indikatorfunktion:

$$\hat{p}_b(X_i^q) := \begin{cases} 1 & wenn\ f_b(X_i) \neq y_i \\ 0 & wenn\ f_b(X_i) = y_i \end{cases}$$

Den Fehler der kombinierten Klassifikatoren erhält man, wenn man das arithmetische Mittel aller Indikatorfunktionen $\hat{p}_b$ heranzieht. Das heißt:

$$Err\left(f_{bag}(X_i^q)\right) := \lim_{B\to\infty} \tfrac{1}{B} \sum_{b=1}^{B} \hat{p}_b(X_i^q)$$

Diese Funktion Err gibt den zu erwartenden Fehler an, der begangen wird, wenn man ein Individuum anhand einer Folge von Klassifikatoren f_i klassifiziert. Wenn gilt, $Err(f_{bag}(X_i^q)) > 0.5$ für einen Datenpunkt $X_i^q \in \mathbb{X}$, wird X_i^q in den meisten Fällen falsch klassifiziert. Somit gibt $Err(f_{bag}(X_i^q))$ an, wie oft im Mittel falsch klassifiziert wurde.

Der Bias wird nun folgendermaßen definiert:

$$Bias\left(f_{bag}(X_i^q)\right) := \begin{cases} 0 & wenn\ Err\left(f_{bag}(X_i^q)\right) \leq 0,5 \\ 1 & wenn\ Err\left(f_{bag}(X_i^q)\right) > 0,5 \end{cases} \tag{3.4}$$

Und die Varianz vom Datenpunkt X_i^q ist definiert als die Differenz aus dem Fehler und dem Bias:

$$Var\left(f_{bag}(X_i^q)\right) := \begin{cases} Err(f_{bag}(X_i^q)) & wenn\ Err\left(f_{bag}(X_i^q)\right) \leq 0,5 \\ Err(f_{bag}(X_i^q)) - 1 & wenn\ Err\left(f_{bag}(X_i^q)\right) > 0,5 \end{cases} \tag{3.5}$$

Die Varianz bei X_i^q gibt die Erhöhung des Fehlers relativ zum Bias an.

3.1.3 Random Features Selection

Jeder Baum wird beim Bagging auf einem unterschiedlichen Datensatz konstruiert und hat somit eine eigene Gestalt. Wenn man einen Baum, wie im CART-Algorithmus, so weit wie möglich wachsen lässt, hat dies zur Folge, dass in den Endknoten die Individuen nach ihren Klassen perfekt getrennt sind. Wenn man nun ein $X_i^q \in \mathbb{X}$ durch die Bäume laufen lässt, wird die Prognose in fast allen Fällen richtig liegen und dadurch wird auch der Bias klein sein. Vergleiche (3.1) und (3.4).

Hat man nun einen ganzen Wald aus "großen" Bäumen mit niedrigem Bias, wird auch der Wald einen niedrigen Bias aufweisen. Das Ziel ist es nun, die Varianz zu verringern, um die Güte des Klassifikators zu verbessern (vergleiche 3.3). Wie bereits erwähnt, sollten sich die Bäume so weit wie möglich unterscheiden, um eine gute Vorhersagekraft zu bewirken. Um diese These mathematisch zu belegen, sei im Folgenden $\{E_1, E_2, ..., E_B\}$ eine Menge von binären Entscheidungsbäumen, die mittels CART-Algorithmus erzeugt wurden. Diese Menge wird nun als Wald bezeichnet. Der Wald besteht somit aus B Bäumen. Weiters bezeichnet $\rho \geq 0$ die Korrelation zwischen den Entscheidungsbäumen und σ^2 bezeichnet die Varianz eines Baumes.

Das arithmetische Mittel von B unabhängigen und identisch verteilten Zufallsvariablen, jede mit einer Varianz σ^2, hat eine Varianz von $\frac{1}{B}\sigma^2$ (vgl. Hastie et al. (2009, S 588)).

Es gilt:

$$\begin{aligned} var\left[\sum_{b=1}^{B} E_b\right] &= cov\left(\sum_{b=1}^{B} E_b, \sum_{b=1}^{B} E_b\right) \\ &= \sum_{b=1}^{B}\sum_{h=1}^{B} cov\left(E_b, E_h\right). \end{aligned}$$

Für $b = h$ ist $\sum_{b=1}^{B}\sum_{h=1}^{B} cov\left(E_b, E_h\right) = \sum_{b=1}^{B} var\left[E_b\right]$ und die restlichen $B\left(B-1\right)$ Terme sind die Kovarianzen, die aber unabhängig vom Index und somit gleich sind. Daher folgt weiters:

$$\begin{aligned} var\left[\sum_{b=1}^{B} E_b\right] &= \sum_{b=1}^{B} var\left[E_b\right] + B\left(B-1\right) cov\left(E_1, E_2\right) \\ &= \sum_{b=1}^{B} \sigma^2 + B\left(B-1\right)\rho{\cdot}\sigma^2 \\ &= B\sigma^2 + B\left(B-1\right)\rho{\cdot}\sigma^2 \end{aligned}$$

Nun ist die Varianz des Waldes das arithmetische Mittel der Varianz aller B Entscheidungsbäume:

$$
\begin{aligned}
var\left[\frac{1}{B}\sum_{b=1}^{B} E_b\right] &= \frac{B\sigma^2}{B^2} + \frac{B\left(B-1\right)\rho{\cdot}\sigma^2}{B^2} \\
&= \frac{\sigma^2}{B} + \rho{\cdot}\sigma^2 - \frac{\rho{\cdot}\sigma^2}{B} \\
&= \rho{\cdot}\sigma^2 + \frac{\sigma^2}{B}\left(1-\rho\right) \qquad (3.6)
\end{aligned}
$$

Anhand der letzten Gleichung lässt sich nun sehr schön erkennen, welche Parameter die Varianz des Random Forest verkleinern lassen. Wenn man B groß wählt und somit eine große Anzahl an Bäumen hat, wird der letzte Term $\frac{\sigma^2}{B}\left(1-\rho\right)$ der Gleichung (3.6) nahe bei 0 liegen und somit $\rho{\cdot}\sigma^2$ übrig bleiben. Genau deswegen ist es wichtig, dass die Korrelation ρ zwischen den Bäumen klein ist, um eine gute Vorhersagekraft eines Random Forest zu erhalten.

Breiman‘s Ansatz, die Korrelation unter den Bäumen weiter zu verringern, war die der Random Features Selection. Als Grundgedanke dahinter gilt, dass man den Merkmalsraum zur Entscheidungsfindung an jedem Knotenpunkt des Baumes reduziert. Random Features Selection bedeutet, dass der CART-Algorithmus bei der Konstruktion eines Baumes an jedem Knotenpunkt K_t zufällig $m < q$ Variablen zieht und anschließend auf einer dieser Variablen den Verzweigungspunkt setzt. Die Größe m wird dabei als fix angenommen. Alle Bäume werden somit eine andere Gestalt aufweisen, weil an jedem Knotenpunkt andere Merkmale für den Verzweigungspunkt verwendet wurden.

Je größer man m wählt, umso ähnlicher wird auch die Gestalt der Bäume sein und umso größer wird auch die Korrelation. Dies ist durch folgende Tatsache begründet: Je größer man m wählt, desto öfter hat der CART-Algorithmus auch ein fixes Merkmal für die Entscheidungsfindung zur Verfügung. Wenn dieses fixe Merkmal die Individuen nach ihren Klassen gut trennen kann, wird es auch oft für einen Verzweigungspunkt verwendet werden.

Neben der Random Features Selection gibt es noch eine weitere Variante, um den Zufall zu simulieren: die der Random Split Selection.

Zur Vervollständigung wird diese hier nur kurz erklärt. Es wird bei der Random Features Selection an jedem Knotenpunkt eine bestimmte Anzahl $m < q$ an Merkmalen bestimmt, die die besten Verzweigungspunkte aufweisen können. Aus diesen Merkmalen wird anschließend zufällig eines gezogen, das dann für den Verzweigungspunkt verwendet wird.

3.2 Definition

Im Folgenden wird ein Zufallsvektor ϕ_b generiert, bevor man den b-ten Entscheidungsbaum E_b konstruiert. Dieser Zufallsvektor ϕ_b ist unabhängig von

den zuvor erzeugten Zufallsvektoren $\phi_1, \phi_2, ...\phi_{b-1}$, aber er hat die gleiche Verteilung wie diese. Anhand dieses Zufallsvektors wird der Entscheidungsbaum $E_b(\cdot\,, \phi_b)$ konstruiert. Im Fall eines Random Forest stellen die ϕ_b jeweils die zufällige Ziehung der n Elemente der Bagging Methode und die zufällige Auswahl an m Merkmalen pro Entscheidung der Random Features Selection dar. Bezeichne mit $\Phi_B := \{\phi_1, \phi_2, ..., \phi_B\}$ die Menge der Zufallsvektoren.

Definition: Ein Random Forest ist ein Klassifikator $h(X, \Phi_B) := \{E_b(X, \phi_b) : b \in \{1, ..., B\} \subseteq \mathbb{N}\}$, der sich aus einer Vielzahl aus baumbasierten Klassifikatoren $E_b(X, \phi_b)$ zusammensetzt, wobei ϕ_b unabhängige und identisch verteilte Zufallsvektoren sind und der Random Forest auf den Eingabedaten X nach dem "the winner takes it all"- Prinzip entscheidet. (Breiman (2001))

Breiman entwickelte in seiner Arbeit im Jahre 1996 das "Bagging" (Bootstrap aggregating). Die Idee, wie bereits erläutert, ist die, dass man viele einzelne Entscheidungsbäume oder Klassifikatoren zu einem Wald zusammenfügt und somit die Vorhersagekraft steigert, indem man dann nach der Mehrheit der Entscheidungen beurteilt. Die große Anzahl der Bäume erhöht die Vorhersagegenauigkeit, da dadurch den Fehlentscheidungen und Instabilitäten eines einzelnen Baumes weniger Bedeutung beigemessen wird. In Abbildung 3.2 auf Seite 41 ist ersichtlich, dass eine Erhöhung der Baumanzahl beim Random Forest die Vorhersagegenauigkeit verbessert bzw. die Missklassifikationsrate verringert. Wie man anhand der schwarzen Kurve erkennen kann, bewirkt eine größere Anzahl der Bäume eine bessere Prognose.

Allerdings hat dies auch zur Folge, dass der Wald sehr unübersichtlich wird, weil es durch die große Anzahl der Bäume schwierig ist, deren einzelne Entscheidungen zu analysieren. Man befindet sich schlussendlich vor einer Art "Black Box", aus der man nur sehr schwer Rückschlüsse ziehen kann, wie das Ergebnis zustande gekommen ist.

Ein großer Vorteil des Random Forest ist der, dass dieser trotz großer Datenmengen sehr effizient ist und nur eine geringe Laufzeit aufweist. Folglich stellt auch ein großer Merkmalsraum kein Problem dar.

Mit einer einfachen und effizienten Methode kann man aus einem Random Forest Maße für die Wichtigkeit der Variablen erhalten. Und mit der Konstruktion des Random Forest wird auch ein erwartungstreuer Schätzer des Generalisierungsfehlers generiert. Dieser Generalisierungsfehler gibt die Wahrscheinlichkeit einer fehlerhaften Klassifizierung eines Individuums an.

3.2.1 Der Random-Forest-Algorithmus

Der Random Forest verwendet bei der Konstruktion der Bäume die Methode der Random Features Selection. An jedem Knoten wird nun der beste Splittingpunkt aus den m zufällig ausgewählten Variablen gesucht.

1. Generiere B unterschiedliche Trainingsdatensätze $\{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_B\}$, sodass jeder Baum auf einem eigenen Trainingsdatensatz basiert.
2. Für jeden dieser Datensätze erzeuge mittels CART-Algorithmus einen ungekürzten Entscheidungsbaum, der an jedem Knoten nur eine zufällige aber fixe Auswahl m aller q Merkmale hat.
3. Der Random Forest entscheidet nach dem Prinzip "The winner takes it all".

Das heißt, es wird anhand der Mehrheitsentscheidung der einzelnen Bäume klassifiziert. Ist jeder dieser Bäume generiert, wird bei einer Klassifikation so entschieden, dass für jene Klasse gestimmt wird, für welche die meisten Bäume gestimmt haben:

$$h_{class}(X, \Phi_B) = arg \max_{j \in \{1,...M\}} \sum_{b=1}^{B} I\left(E_b(X, \phi_b) = c_j\right) \tag{3.7}$$

mit:

c_j	Klasse j, $j \in \{1, ...M\}$
E_b	Folge von Entscheidungsbäumen
I	Indikatorfunktion

Bei Regressionsbäumen kann man nicht nach der Mehrheit der einzelnen Bäume entscheiden, weil die Ergebnisse beispielsweise Werte aus den reellen Zahlen sind. Aus diesem Grund wird das arithmetische Mittel der Ergebnisse aller Entscheidungsbäume herangezogen. Man erhält bei einer Regression somit:

$$h_{reg}(X, \Phi_B) = \frac{1}{B} \sum_{b=1}^{B} E_b(X, \phi_b) \tag{3.8}$$

Die im R-Package "RandomForest" vorhandene Implementierung eines Random Forest verwendet 500 Bäume als Default Wert. Weiters wird bei einer Klassifizierung $m = \sqrt{q}$ und bei einer Regression $m = {}^{3}/_{q}$ als Default Wert für die Random Features Selection gewählt.

3.3 Konvergenz

Bei Klassifizierungs- bzw. Regressionsmodellen hat man oft das Problem, dass diese Modelle an die Trainingsdaten überangepasst sind. Das bedeutet, dass die Modelle auf der Trainingsmenge zwar gut funktionieren, aber bei neuen Beobachtungen schlechte Ergebnisse liefern. Der Grund dafür ist der, dass das Modell zu sehr an den Trainingsdatensatz gebunden ist. Man kann beispielsweise einen Baum so lange verzweigen, bis in den Endknoten nur noch Individuen einer Klasse vorhanden sind. Somit sind aber die später zu treffenden Entscheidungen an eine kleine Gruppe von Individuen angepasst und deswegen werden sie auch immer spezifischer.

Nun stellt sich die Frage, ob sich ein Random Forest an die Trainingsdaten überanpasst, wenn man die Anzahl der Bäume erhöht. Die Antwort darauf ist "Nein", eine hohe Anzahl an Bäumen verursacht keine Überanpassung (vgl. Breiman (2001)).

Um dies zu begründen, benötigen wir im Folgenden die Marge-Funktion:

$$mg(X,y) := \frac{1}{B}\sum_{b=1}^{B} I\left[f_b(X) = y\right] - \max_{y_j \neq y}\left(\frac{1}{B}\sum_{b=1}^{B} I\left[f_b(X) = y_j\right]\right) \tag{3.9}$$

mit:

I	Indikatorfunktion
(X,y)	ein Element aus $\mathbb{X}$, welches zufällig gezogen wurde
f_b	Folge von Klassifikatoren
B	die Anzahl an Klassifikatoren

Es ist zu beachten, dass die Marge-Funktion für eine Folge von beliebigen Klassifikatoren definiert ist. Die Marge-Funktion misst den relativen Anteil der Klassifikatoren, die korrekt klassifiziert haben. Davon wird der relative Anteil jener Klassifikatoren abgezogen, die am häufigsten für ein und dieselbe falsche Klasse gestimmt haben. Diese Funktion nimmt für ein $X_i \in \mathbb{X}$ positive Werte an, wenn die Mehrheit der Klassifikatoren X_i der richtigen Klasse zuweisen. Hingegen ist die Funktion für ein $X_i \in \mathbb{X}$ negativ, wenn es eine Mehrheit an Klassifikatoren gibt, die sich für die selbe falsche Klasse entscheiden.

Der Wert einer Marge-Funktion gibt Auskunft über die Güte eines Random Forest. Werte nahe bei 0 bedeuten, dass sich etwa gleich viele Bäume für eine falsche wie auch für die richtige Klasse entschieden haben. Aus diesem Grund weisen Werte um 0 auch auf einen unsicheren Random Forest hin. Hingegen zeigen Werte nahe bei 1, dass die Bäume in den meisten Fällen X_i richtig

klassifiziert haben und folglich der Random Forest zuverlässig ist. Es gilt: je größer der Wert, desto zuverlässiger ist die Klassifikation.

Der Generalisierungsfehler ist nun definiert als die Wahrscheinlichkeit einer negativen Marge-Funktion:

$$\overset{*}{Err} := \mathbb{P}_{(X,y)}[mg(X,y) < 0]$$

Der Index (X, y) gibt den zu betrachtenden Wahrscheinlichkeitsraum an.

Der Random Forest $h(X, \Phi_B) := \{E_b(X, \phi_b) : b \in \{1, ..., B\}\}$ besteht bekanntlich aus einer Folge von Entscheidungsbäumen. Zu jedem Baum E_b existiert ein dazugehöriger Zufallsvektor ϕ_b. Anhand des Gesetzes der großen Zahlen kann gezeigt werden, dass mit steigender Baumanzahl Err für fast alle Folgen $\phi_1, \phi_2, ..., \phi_B \sim P(\theta)$ konvergiert und zwar gegen:

$$\overset{*}{Err} \underset{B\to\infty}{\longrightarrow} \mathbb{P}_{(X,y)}\left[\mathbb{P}_{\Phi_B}(h(X,\Phi_B) = y) - \underset{y_j \neq y}{max}\mathbb{P}_{\Phi_B}(h(X,\Phi_B) = y_j) < 0\right]$$

Für den Beweis siehe: Breiman (2001, S. 30).

Dies bedeutet nun, dass sich ein Random Forest nicht an die Lerndaten überanpasst, auch wenn man die Anzahl der Bäume erhöht. Eine große Baumanzahl bewirkt sogar, dass der Generalisierungsfehler gegen den wahren Wert konvergiert. Somit sollte man sogar eine große Anzahl an Entscheidungsbäumen bevorzugen, weil dadurch auch die Varianz des Random Forest verringert wird. (vgl: (3.6))

Es kann nur dann eine Überanpassung des Random Forest auftreten, wenn alle Bäume an die Trainingsdaten überangepasst sind. Allerdings ist dann für die Überanpassung des Random Forest nicht die Anzahl, sondern nur die Größe der Entscheidungsbäume relevant. Dem Problem der Überanpassung kann man dadurch entgegenwirken, indem man die Bäume stutzt (vgl. Abschnitt 2.2.3).

Mit der Tschebyscheff-Ungleichung kann man für den Generalisierungsfehler eines Random Forest eine obere Grenze abschätzen. Dazu ist die Marge-Funktion eines Random Forest folgendermaßen definiert:

$$mg_{RF}(X,y) := \mathbb{P}_{\phi}\left(h(X,\Phi_B) = y\right) - \underset{y_j \neq y}{max}\mathbb{P}_{\phi}\left(h(X,\Phi_B) = y_j\right)$$

Die Güte der Vorhersage eines Random Forest kann man ausdrücken als den Erwartungswert der Marge-Funktion (vgl. Breiman (2001)):

$$s := \mathbb{E}_{(X,y)}\left(mg_{RF}(X,y)\right)$$

Und mit der Tschebyscheff-Ungleichung erhält man für ein $s > 0$ folgende Abschätzung des Generalisierungsfehlers:

$$\begin{aligned} \overset{*}{Err} &:= \mathbb{P}_{(X,y)}\left(mg_{RF}(X,y) < 0\right) \\ &\leq \mathbb{P}_{(X,y)}\left(|mg_{RF}(X,y) - s| \geq s\right) \\ &\leq \frac{Var(mg_{RF})}{s^2} \end{aligned}$$

Breiman (2001, S. 8) leitete in seiner Arbeit noch folgende Abschätzung für die Varianz ab: $Var(mg_{RF}) \leq \bar{\rho}(1 - s^2)$. Wobei $\bar{\rho}$ das arithmetische Mittel aller Korrelationen der Bäume darstellt. Somit kann man eine obere Schranke des Generalisierungsfehlers auch folgendermaßen angeben:

$$\overset{*}{Err} \leq \frac{Var(mg_{RF})}{s^2} \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{3.10}$$

Diese Abschätzung rechtfertigt ein weiteres Mal die Verwendung der Bagging- bzw. die der Random-Features-Selection-Methode. Wie bei der Abschätzung (3.10) ersichtlich wird, bewirkt ein kleines $\bar{\rho}$ und somit viele differente Bäume im Wald eine Verringerung des Generalisierungsfehlers.

Im Großen und Ganzen gilt: Je mehr Klassen vorhanden sind und je größer die Anzahl der Merkmale ist, desto mehr Bäume sollte man für die Konstruktion des Waldes verwenden. Damit wird auch gewährleistet, dass jedes Merkmal so oft wie möglich als Verzweigungspunkt eines Baumes verwendet wird. Im R Package "RandomForest" ist der Default Wert für die Anzahl der Bäume 500.

3.4 "Out of Bag" Daten

Bezeichne mit $\mathcal{L}^b$ die Lernmenge, die für die Konstruktion des Baumes E_b verwendet wurde. Die Elemente $X_i \in \mathcal{L}^b$ nennt man die "In-of-Bag" Daten ("IoB"-Daten) eines Klassifikators. Hingegen bezeichnet man mit den "Out-of-Bag"-Daten ("OoB"-Daten) jene Daten, die nicht für die Konstruktion verwendet wurden. Die Menge der "Out of Bag"-Daten des Baumes E_b lassen sich somit folgendermaßen definieren:

$$\mathbb{X}_b^{OoB} := \{(X_1^q, y_1), ..., (X_{N_b}^q, y_{N_b})\} = \mathbb{X} \backslash \mathcal{L}_b$$

Die Wahrscheinlichkeit ist $1/N$, dass mit der Bagging Methode bei einer Ziehung ein bestimmtes Individuum $(X_i, y_i) \in \mathbb{X}, i \in \{1, ..., N\}$ aus dem Datensatz $\mathbb{X}$ gezogen wird. Im Folgenden ist $i \in \{1, ..., N\}$ und $b \in \{1, ..., B\}$

festgehalten. Da man n mal mit Zurücklegen zieht, ist die Wahrscheinlichkeit $(\frac{1}{N})^n$, dass X_i für die Erstellung des Baumes E_b verwendet wird. Die Gegenwahrscheinlichkeit dazu, also $X_i \notin \mathcal{L}_b$, ist: $1 - (\frac{1}{N})^n$.

Für großes n gilt:

$$\lim_{n\to\infty} \left[1 - (\tfrac{1}{N})^n\right] = \tfrac{1}{e} \approx 37\%.$$

Somit wird für die Erstellung eines Baumes mit einer Wahrscheinlichkeit von 37% ein fix gewähltes Individuum nicht gezogen. Folglich sind nun im Durchschnitt 37% der Trainingsdaten eines Baumes Elemente der "OoB"-Daten und im Durchschnitt 63% der Trainingsdaten eines Baumes Elemente der "IoB"-Daten.

Unter Zuhilfenahme der "Out-of-Bag"-Daten lassen sich einige Eigenschaften des Random Forest validieren. Man erhält beispielsweise Aussagen über die Güte des Klassifikators, ohne dass man dazu einen zusätzlichen Testdatensatz benötigt. Mehr dazu wird in den nachfolgenden Abschnitten erörtert.

3.4.1 Missklassifikationsfehler

Mit einer einfachen Methode kann man mit Hilfe der "Out-of-Bag"-Daten einen erwartungstreuen Schätzer für die Missklassifikationsrate erhalten. Hat man so einen Schätzer ermittelt, besteht der Vorteil darin, dass weitere Verfahren, wie beispielsweise das Kreuzvalidierungsverfahren, für die Schätzung der Güte eines Klassifikators überflüssig werden.

Klassifikation

In diesem Abschnitt wird nur ein Klassifikationsproblem behandelt. Um einen Schätzer für die Missklassifikationsrate zu erhalten, besteht der Leitgedanke darin, dass man durch jeden Baum nur seine "Out-of-Bag"-Daten schickt, von welchen ja auch das Ergebnis y bekannt ist. Daraus lässt sich ermitteln, wie oft dieser Baum die "Out-of-Bag"-Daten richtig klassifiziert. Der Missklassifikationsfehler des Baumes errechnet sich dann als die Summe der falsch klassifizierten Individuen durch die Mächtigkeit der "OoB"-Daten des Baumes.

Bezeichne mit $N_b :=\mid \mathbb{X}_b^{OoB} \mid$ die Mächtigkeit der "Out-of-Bag"-Daten des Baumes E_b und I ist anschließend wieder die Indikatorfunktion. Der Missklassifikationsfehler des Baumes E_b ist bestimmt als:

$$Err_{_{OoB}}(E_b) := \frac{1}{N_b} \sum_{X_i \in \mathbb{X}_b^{OoB}} I\left[E_b(X_i, \phi_b) \neq y_i\right] \tag{3.11}$$

mit:

E_b	ein Entscheidungsbaum
X_i	ein Individuum der "OoB"-Daten.
y_i	die wahre Klasse von X_i
N_b	Mächtigkeit der "OoB"-Daten des Baumes E_b
ϕ_b	Zufallsvektoren (vgl. Abschnitt 3.2)

Der Missklassifikationsfehler der "OoB"-Daten des Random Forest $h(\cdot, \Phi_B)$ ist nun das arithmetische Mittel der Summe aller Missklassifikationsfehler der Bäume, das heißt:

$$Err_{_{OoB}}(h(\mathbb{X}^{OoB}, \Phi_B)) := \frac{1}{B} \sum_{b=1}^{B} Err_{_{OoB}}(E_b) \tag{3.12}$$

mit:

B	die Anzahl der Bäume im Wald

Dieser Fehler gibt die Wahrscheinlichkeit für eine inkorrekte Klassifizierung an und ist deswegen auch ein Maß für die Güte der Klassifikation eines Random Forest.

Regression

Bei einem Regressionsproblem ist die zuvor erwähnte Methode zur Bestimmung des Missklassifikationsfehlers nicht anwendbar, da die Zielvariablen numerische Werte sind und keine Klassen mehr darstellen. Deswegen betrachtet man in diesem Fall die mittlere quadratische Abweichung (engl: mean squared Error "MSE") des Zielwertes vom geschätzten Wert.

Bezeichne mit $N_{X_i} := \#\{X_i \in \mathbb{X}_b^{OoB}\}$, $b \in \{1, ..., B\}$, die Anzahl der Bäume, in denen X_i ein Element der "Out-of-Bag"-Daten ist. I^{OoB} stellt eine Indikatorfunktion dar, die genau dann das geschätzte Ergebnis $\widetilde{y}_i := E_b(X_i, \phi_b)$ des Entscheidungsbaumes liefert, wenn X_i ein Element der "Out of Bag"-Daten von E_b ist:

$$I^{OoB}\left[E_b(X_i, \phi_b)\right] := \begin{cases} \widetilde{y}_i & wenn\ X_i \in \mathbb{X}_b^{OoB} \\ 0 & sonst \end{cases}$$

Der aus den "OoB" geschätzte Zielwert y_i^{OoB} von X_i ist das arithmetische Mittel der Ergebnisse der Bäume, in denen X_i ein Element der "OoB"-Daten ist:

$$y_i^{OoB} := \frac{1}{N_{X_i}} \sum_{b=1}^{B} I^{OoB} \left[E_b(X_i, \phi_b) \right] \tag{3.13}$$

mit:

E_b	ein Entscheidungsbaum
N_{X_i}	Anzahl der Bäume, in denen $X_i \in \mathbb{X}_b^{OoB}$ ist

Die mittlere quadratische Abweichung der "OoB"-Daten eines Random Forest $h(\cdot\,, \Phi_B)$ beträgt somit:

$$MSE_{OoB}(h(\cdot\,, \Phi_B)) := \frac{1}{N} \sum_{i=1}^{N} (y_i - y_i^{OoB})^2 \tag{3.14}$$

mit:

N	Anzahl der Individuen des Datensatzes $\mathbb{X}$

Je kleiner der Wert dieser Funktion ist, desto näher ist der geschätzte am tatsächlichen Wert. Somit gilt auch in diesem Fall: Je kleiner der Funktionswert, desto besser ist die Regression.

3.4.2 Permutierte Merkmalswichtigkeit

Eine weitere und ebenfalls sehr nützliche Information, welche man aus den "Out-of-Bag"-Daten gewinnen kann, ist die der Variablen- oder Merkmalswichtigkeit. Wenn man mit einem vorhandenen Datensatz ein Vorhersagemodell konstruieren will, taucht oft folgende Problemstellung auf: Welche der unabhängigen Variablen korrelieren mit der abhängigen Zielvariablen Y? Anders augedrückt: Welche Merkmale eines Individuums geben Aufschluss über deren Klassenzugehörigkeit?

Man kann sich einiger Verfahren bedienen, um herauszufinden, welche Merkmale wichtig für die Vorhersage von Y sind. Man kann etwa eine Faktorenanalyse durchführen oder mit mehreren Regressionsmodellen testen, welche Variablen Y am besten beschreiben. In einem Random Forest benötigt man diese Verfahren nicht, man kann ausschließlich aus diesem Wald die dafür benötigte Information gewinnen.

Die Absicht ist die, dass man die Auswirkungen auf die Vorhersage von Y betrachtet, wenn man alle Werte eines Merkmales vertauscht. Man simuliert dabei, welchen Einfluss der Informationsverlust eines Merkmales auf die Vorhersage des Random Forest hat. Wenn ein Merkmal mit Y korreliert, sollte

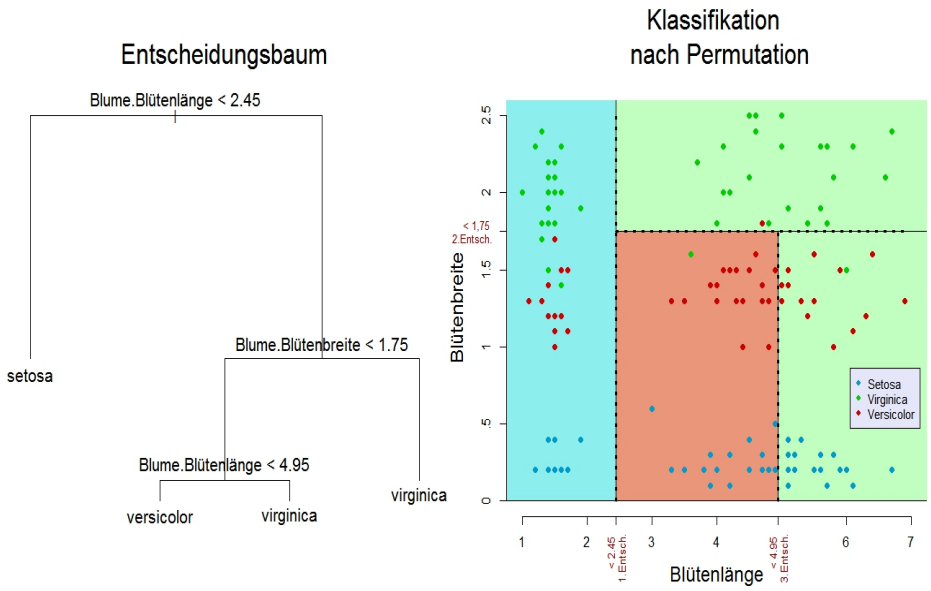

Abbildung 3.1: Wenn man die Werte des Merkmals "Blütenlänge" permutiert, entsteht ein komplett anderes Bild. Deswegen wird eine Permutation auf diesem Merkmal eine hohe Missklassifikationsrate nach sich ziehen. Die hinterlegte Farbe zeigt an, wie der Entscheidungsbaum die Individuen mit den permutierten Merkmalen klassifizieren würde.(vgl. dazu auch Abbildung 2.1)

nach einer Permutation dieser Werte der Random Forest ein schlechteres Ergebnis liefern. Somit erwartet man: Je "wichtiger" ein Merkmal ist, umso größer ist auch der Missklassifikationsfehler nach einer Permutation dieser Werte. Wenn man mit $Err^{\pi_j}_{OoB}$ den Fehler mit permutierten Werten des j-ten Merkmales bezeichnet, sollte bei einem "wichtigen" Merkmal x^j gelten:

$$Err_{OoB} < Err^{\pi_j}_{OoB},\ j \in \{1, ..., q\}$$

In Abbildung 3.1 wird die Permutation eines Merkmales dargestellt. Hier wurden die Werte des Merkmals "Blütenlänge" permutiert und man sieht nun, dass der Entscheidungsbaum nach der Permutation in vielen Fällen falsch klassifiziert. Somit ist das Merkmal "Blütenlänge" ein wichtiges.

Klassifikation

Bezeichne wieder mit $N_b :=\mid \mathbb{X}^{OoB}_b \mid$ die Mächtigkeit der "Out-of-Bag"-Daten des Baumes E_b. Der Random Forest besteht aus B Bäumen, die mittels CART-Algorithmus erzeugt wurden und π_j stellt die Permutation von $\{1, ..., N_b\}$ des $j-ten$ Merkmales x^j von $\mathbb{X}$ dar:

$$X^{\pi_j}_i := (x^1_i, ..., x^{j-1}_i, x^{\pi_j}_i, x^{j+1}_i, ..., x^q)$$

Und die permutierten "Out of Bag"-Daten des Baumes E_b sind:

$$\mathbb{X}_b^{OoB_{\pi_j}} := \{X_1^{\pi_j}, ..., X_{N_b}^{\pi_j}\}$$

Um die weiteren Definitionen übersichtlicher zu machen, bezeichne mit $E_b(\cdot) := E_b(\cdot, \phi_b)$ und definiere folgende Indikator-Funktion:

$$\overset{*}{I}(E_b, X_i) := \begin{cases} 1 & wenn\ E_b(X_i) = y_i \\ 0 & wenn\ E_b(X_i) \neq y_i \end{cases}$$

mit:

E_b	ein Entscheidungsbaum
X_i	ein Individuum der "OoB"-Daten, also $X_i \in \mathbb{X}_b^{OoB}$
y_i	die wahre Klasse von X_i

Betrachte zuerst nur einen Baum mit seinen "Out of Bag"-Daten, dann ist die Variablenwichtigkeit des Merkmales x^j gegeben als:

$$imp(E_b, \pi_j) := \frac{1}{N_b} \sum_{i=1}^{N_b} \left[\overset{*}{I}(E_b, X_i) - \overset{*}{I}(E_b, X_i^{\pi_j}) \right] \tag{3.15}$$

mit:

X_i	Ein Individuum $X_i \in \mathbb{X}_b^{OoB_j}$
N_b	Mächtigkeit der "OoB"-Daten des Baumes E_b

Und die vom Random Forest $h(\cdot, \Phi_B)$ erzeugte Variablenwichtigkeit des Merkmales x^j ist nun das arithmetische Mittel der Summe der Variablenwichtigkeit aller Bäume:

$$imp_{class}\left(h(\mathbb{X}^{OoB}, \Phi_B), \pi^j\right) := \frac{1}{B} \sum_{b=1}^{B} imp(E_b, \pi^j) \tag{3.16}$$

mit:

B	die Anzahl der Bäume im Wald

Wenn nun eine Permutation des Merkmales x^j eine große Veränderung der Antwortvariablen bewirkt, in dem Sinne, dass dadurch die Bäume oft falsch klassifizieren, wird $imp\,(h(\cdot, \Phi_B), x^j)$ einen Wert nahe 1 haben. Es gilt: Je "wichtiger" ein Merkmal x^j für die Vorhersage ist, desto höher wird der Wert der Funktion $imp\,(h(\cdot, \Phi_B), x^j)$ sein.

Regression

In Falle eines Regressionsproblems betrachtet man, wie sich nach einer Permutation die Summe der Fehlerquadrate in den Endknoten der Bäume verändern (vgl. 2.8). Je wichtiger ein Merkmal ist, desto größer wird auch die Summe der Fehlerquadrate in den Endknoten sein.

Dazu sei der geschätzte Zielwert mit permutierten Werten folgendermaßen definiert (vgl. 3.13):

$$y_i^{OoB}(\pi_j) := \frac{1}{N_{X_i}} \sum_{b=1}^{B} I^{OoB} \left[E_b(X_i^{\pi_j}, \phi_b) \right]$$

Die mittlere quadratische Abweichung von den permutierten "OoB"-Daten eines Random Forest $h(\cdot\,, \Phi_B)$ beträgt somit:

$$MSE_{OoB}(h(\cdot\,, \Phi_B), \pi_j) := \frac{1}{N} \sum_{i=1}^{N} (y_i - y_i^{OoB}(\pi_j))^2$$

Um die Notationen einfacher zu gestalten, bezeichne mit $MSE_{OoB}(\pi_j) := MSE_{OoB_{\pi_j}}(h(\cdot\,, \Phi_B), \pi_j)$ und mit $MSE_{OoB} := MSE_{OoB}(h(\cdot\,, \Phi_B))$. Dann ist die permutierte Merkmalswichtigkeit gegeben als (vgl. Liaw and Wiener (2002, Seite 20)):

$$imp_{reg}(h(\mathbb{X}^{OoB}, \Phi_B), \pi^j) := \frac{1}{B} \sum_{b=1}^{B} (MSE_{OoB} - MSE_{OoB}(\pi_j)) \qquad (3.17)$$

Wenn nun ein Merkmal wenig Einfluss auf die Zielvariable Y hat, werden sich weder MSE_{OoB} noch $MSE_{OoB}(\pi_j)$ unterscheiden, deswegen wird auch imp_{reg} nahe bei 0 sein. Je größer der Einfluss eines Merkmals auf die abhängige Variable Y ist, desto größer ist auch der Unterschied von MSE_{OoB} und $MSE_{OoB}(\pi_j)$.

3.4.3 Weitere Maße für Merkmalswichtigkeiten

Es gibt noch weitere Methoden, ein Maß für die Merkmalswichtigkeit zu bestimmen. Der CART-Algorithmus setzt bekanntermaßen die Verzweigungspunkte auf jene Merkmale, welche die Individuen nach den Klassen am besten trennen. Wird ein Merkmal markant öfter als andere Merkmale für einen Verzweigungspunkt verwendet, heißt das, dass ein Verzweigungspunkt auf diesem Merkmal eine höhere Trennschärfe bewirkt. Wenn man nun das arithmetische Mittel betrachtet, wie oft ein Merkmal für eine Entscheidungsfindung der Bäume verwendet wurde, dann erhält man ein weiteres Merkmalswichtigkeits-Maß. Bezeichne dieses Maß im Folgenden als die Verwendungswichtigkeit. Dieses Maß lässt sich somit unabhängig der "Out of Bag"-Daten bestimmen.

Mit Hilfe der Gini-Wichtigkeit kann man ein weiteres Merkmalswichtigkeits-Maß bestimmen. Dieses lässt sich ebenfalls ohne "OoB"-Daten berechnen.

Man betrachtet aber nicht, wie bei der Verwendungswichtigkeit, wie oft ein Merkmal für eine Entscheidungsfindung verwendet wurde, sondern man betrachtet die Verbesserung, die diese Entscheidung auf dem Merkmal nach sich zieht. Bei jedem Verzweigungspunkt, an dem ein Merkmal verwendet wurde, wird die Verbesserung der Gini Reinheit (vgl. 2.3) betrachtet. Die Gini-Wichtigkeit des Merkmals x^i ist das arithmetische Mittel der Gini-Reinheiten an den Knotenpunkten, an denen das Merkmal x^i als Verzweigungskriterium verwendet wurde.

Bei einer Regression kann man bekanntlich die Gini-Wichtigkeit nicht anwenden. Man verwendet in diesem Fall erneut die mittlere quadratische Abweichung (MSE) (vgl. 2.7). Die aus diesem Wert ermittelte Wichtigkeit des Merkmals x^i ist das arithmetische Mittel der $\Delta\phi(K_t, s)$ von allen Knotenpunkten, an denen das Merkmal x^i als Verzweigungskriterium verwendet wurde.

3.5 Distanzmaß

Ziel dieses Abschnittes ist es, durch einen Random Forest ein Maß für Gemeinsamkeiten und daraus ein Distanzmaß für zwei Individuen abzuleiten. Durch einen Random Forest wird per se kein Distanzmaß bestimmt. Deswegen muss man zuerst definieren, was es bei einem Random Forest bedeuten soll, wenn zwei Individuen ähnlich positioniert sind. In einem Random Forest sollen zwei Individuen umso mehr Gemeinsamkeiten aufweisen, je öfter sie in den gleichen Endknoten eines Entscheidungbaumes fallen.

Dazu bezeichne nun im Folgenden wieder mit $\bar{E}_b \in \mathbb{N}$ die Anzahl der Endknoten des Baumes E_b und mit $\overline{K}_b^l$, $l \in \{1, ..., \bar{E}_b\}$ den l-ten Endknoten des Baumes E_b. Definiere weiters die Indikatorfunktion, die nur dann 1 ist, wenn zwei Individuen in den selben Endknoten fallen:

$$I_b^l : (X \times X) \longrightarrow \{0, 1\} := \begin{cases} 1 & wenn\ X_i \wedge X_j \in \overline{K}_b^l \\ 0 & sonst \end{cases}$$

Nun ist ein Maß für die Gemeinsamkeit bzw. Proximities zweier Elemente definiert als:

$$prox(X_i, X_j) := (X \times X) \longrightarrow (0, 1) : \frac{1}{B} \sum_{b=1}^{B} \left[\sum_{l=1}^{\bar{E}_b} I_b^l(X_i, X_j) \right] \tag{3.18}$$

mit:

X_i Ein Individuum $X_i \in \mathbb{X}$

B	Anzahl der Bäume im Wald.
$\bar{E}_b$	Anzahl der Endknoten des Entscheidungsbaumes E_b

Je größer der Wert von $prox(X_i, X_j)$ ist, desto öfter fallen die beiden Individuen in den selben Endknoten der Bäume. Deswegen sind zwei Elemente $X_i \in \mathbb{X}$ und $X_j \in \mathbb{X}$ mit $prox(X_i, X_j) \approx 1$ für einen Random Forest kaum unterscheidbar. Klarerweise gilt $prox(X_i, X_i) = 1$ und $0 \leq prox(X_i, X_j) \leq 1$, $\forall i, j \in \{1, ..., N\}$.

Bezeichne im Folgenden mit $prox_{i,j} := prox(X_i, X_j)$. Wenn man nun dieses Maß für jedes Datenpaar $(X_{i,}X_j)$ berechnet, erhält man die $N \times N$ Proximity Matrix mit $diag(Prox) = 1$

$$Prox := \begin{pmatrix} prox_{1,1} & prox_{1,2} & \cdots & prox_{1,N} \\ prox_{2,1} & prox_{2,2} & & \vdots \\ \vdots & & \ddots & \vdots \\ prox_{N,1} & prox_{N,2,} & \cdots & prox_{N,N} \end{pmatrix}$$

Diese Matrix ist symmetrisch, positiv definit und für alle Einträge gilt: $0 \leq prox_{i,j} \leq 1$ $\forall i, j \in \{1, ..., N\}$.

Mit den Proximities ist es möglich, einen durch den Random Forest erzeugten Distanzbegriff zu definieren:

$$d(X_i, X_j) := 1 - prox(X_i, X_j)$$

Diese Funktion ist allerdings keine Metrik: Es gilt weder die Dreiecksungleichung noch ist die erste Bedingung einer Metrik erfüllt, das heißt, dass eine Funktion $d(X_i, X_j)$ nur dann 0 ist, wenn gilt $X_i = X_j$. Denn im Fall eines Random Forest kann es unterschiedliche Individuen geben, die immer in die gleichen Endknoten der Bäume fallen.

Mit Hilfe dieses Maßes ist es möglich, Ausreißer aufzuspüren. Hat man einen Datensatz, bei dem einige Individuen fehlende Merkmale haben, wird ebenfalls die *prox* in Betracht gezogen, um diese fehlenden Werte zu ersetzen.

3.5.1 Ausreißer

In einem Random Forest zählen zu den Ausreißern einer Klasse jene Individuen, die zu den anderen Individuen der Klasse nur geringe Gemeinsamkeiten aufweisen. Mit den Proximities hat man ein Werkzeug, um diese Gemeinsamkeiten messen zu können.

Definiere für ein festgehaltenes $X_i \in y_l$ eine Funktion (vgl. Yang et al. (2009)):

$$\overline{prox(X_i)} = \sum_{X_j \in y_l} [prox(X_i, X_j)]^2$$

mit:

X_i, X_j zur Klasse y_l gehörige Individuen

$\overline{prox(X_i)}$ gibt einen Wert an, wie sich das Element X_i der Klasse y_l zu den restlichen Elementen derselben Klasse unterscheidet. Je größer der Wert ist, desto ähnlicher ist X_i zu den restlichen Individuen der Klasse y_l.

Bezeichne nun mit $n_l := |y_l|$, $l \in \{1, ..., M\}$, die Mächtigkeit der Klasse l und definiere weiters folgendes Maß:

$$out_{y_m}(X_i) = \frac{n_l}{\overline{prox(X_i)}}.$$

Dieses Maß wird kleiner, je geringer die Unterschiede der Elemente einer Klasse sind (großes $\overline{prox(X_i)}$), und größer, je mehr sich die Elemente unterscheiden (kleines $\overline{prox(X_i)}$). Von jeder Klasse y_m bezeichne mit $\overline{out}_{y_m}$ den Median der $out_{y_m}(X_i)$, und bezeichne mit σ die absolute Abweichung zum Median.

Das Ausreißermaß ist dann folgendermaßen definiert:

$$out(X_i) = \frac{out_{y_m}(X_i) - \overline{out}_{y_m}}{\sigma}$$

Je größer dieser Wert ist, desto eher wird das Individuum X_i als Ausreißer deklariert.

3.5.2 Fehlende Werte

Oft taucht das Problem auf, dass man einen Datensatz $\mathbb{X}$ mit fehlenden Einträgen hat. Die im R-Package implementierte Funktion "rfImpute" verwendet die $\overline{prox(X_i)}$, um fehlende Werte zu ersetzen. Um den folgenden Algorithmus übersichtlicher zu machen, teilt man den Datensatz $\mathbb{X}$ in zwei Teildatensätze, sodass gilt: $\mathbb{X} = \mathbb{X}_{fehl} \bigcup \mathbb{X}_{voll}$. Dabei bezeichnet man mit $\mathbb{X}_{fehl}$ den Datensatz, der aus jenen Individuen besteht, die fehlende Werte haben. $\mathbb{X}_{voll}$ stellt den Datensatz dar, der die Individuen enthält, deren Einträge vollständig sind. Weiters besteht der Datensatz $\mathbb{X}_{voll}$ aus N_{voll} Individuen.

Es wird bei einem Datensatz mit fehlenden Werten mit folgendem Algorithmus vorgegangen:

1. Wenn bei einem Individuum $X_i \in \mathbb{X}_{fehl}$ ein Eintrag eines kategoriellen Merkmals $x_i^j \in \{c_1, ..., c_K\}$ mit K unterschiedlichen Kategorien fehlt, so setzt man jene Kategorie ein, für die gilt:

$$x_i^j = c_k, \; mit \; c_k = \underset{k \in \{1,...,K\}}{arg\,max} \left[\sum_{X_l \in \mathbb{X}_{voll}} I(x_l^j = c_k) \right]$$

Wenn ein Eintrag eines numerischen Merkmals fehlt, so setzt man folgenden Wert ein:

$$x_i^j := \frac{1}{N_{Voll}} \left[\sum_{X_l \in \mathbb{X}_{voll}} x_l^j \right]$$

2. Auf diesem Datensatz wird ein Random Forest samt seiner Proximity Matrix erstellt.

3. War der fehlende Wert von X_i bei einem numerischen Merkmal x^j, so nimmt man das gewichtete Mittel der restlichen Werte des Merkmales. Die Gewichte sind in diesem Fall die $prox$, sodass die Gewichte zu ähnlicheren Objekten größer werden als die zu unähnlicheren. Der einzusetzende Wert berechnet sich folgendermaßen:

$$\hat{x}_l^j := \frac{1}{N_{Voll}} \left[\sum_{X_l \in \mathbb{X}_{voll}} x_i^j \cdot prox(X_l, X_i) \right]$$

Ist der fehlende Wert von X_i bei einem kategoriellen Merkmal x_i^j mit K unterschiedlichen Kategorien, so wählt man diejenige Kategorie, die am häufigsten vorkommt. Allerdings werden die Häufigkeiten der Kategorien ebenfalls mit den $prox$ gewichtet. Die einzusetzende Kategorie ist diejenige, die folgenden Wert maximiert.

$$\hat{x}_i^j = c_k, \; mit \; c_k = \underset{k \in \{1,\ldots,K\}}{arg\,max} \left[|c_k| \cdot \sum_{\substack{X_l \in \mathbb{X}_{voll} \\ x_l^j = c_k}} prox(X_i, X_l) \right]$$

mit:

K	Anzahl der Kategorien des Merkmals x^j
$\lvert c_k \rvert$	Anzahl der Individuen $X_l \in \mathbb{X}_{voll}$ mit dem Merkmal $x_l^j = c_k$
X_l	Individuum mit Merkmal $x_l^j = c_k$

Wiederhole Schritte 2 und 3 so lange, bis die maximale Anzahl an Iterationsschritten erreicht wurde. Laut Breiman sollten vier bis sechs Iterationsschritte ausreichen, um ein zufriedenstellendes Ergebnis zu erhalten.

Diese Vorgehensweise ist sehr rechenintensiv, da man jedesmal einen ganzen Wald samt seiner Proximity erstellen muss. Eine weitere und viel weniger rechenintensive Möglichkeit, trotz fehlender Werte einen Random Forest aufzubauen, ist die der Durchführung nur des ersten Schrittes des Algorithmus.

Dadurch erspart man sich die mehrfache Konstruktion eines Random Forest mit seiner Proximity Matrix.

Im R-Package "Random Forest" ersetzt der Befehl "na.roughfix()" die fehlenden Werte mit der Vorgehensweise, wie unter Punkt 1 des Algorithmus beschrieben wurde. Die Funktion "rfImpute()" konstruiert anhand der Proximity die neuen Werte, wie in den Punkten 2 und 3 erläutert.

3.6 Evaluierung

3.6.1 Güte bei unbalancierten Daten

In vielen Datensätzen ist die Klassengröße der Zielvariablen ungleichmäßig verteilt und das stellt für viele Lernalgorithmen ein Problem dar. Wenn man wieder mit $n_j := \#\{X_i \in n_j\}$ die Anzahl der Individuen bezeichnet, die der Klasse n_j angehören, dann ist die Klassengröße ($j = 2$) ungleichmäßig verteilt, wenn n_1 viel größer als n_2 ist, bzw. umgekehrt. Man spricht in diesen Fällen von einem unbalancierten Datensatz (engl: imbalanced data). Modelle, die auf einem solchen Datensatz trainiert werden, weisen das Problem auf, dass diese für die Klassifizierung jene Individuen bevorzugen, die der größeren Klasse angehören.

Entscheidungsbäume sind beispielsweise von diesem Problem betroffen, wenn diese auf einem unbalancierten Datensatz konstruiert werden: Angenommen, man hat einen Datensatz mit 100 Individuen, die jeweils zwei Klassen ($\mathbb{Y} := \{A, B\}$) angehören. Nur ein Individuum repräsentiert die Klasse A und die restlichen 99 gehören zur Klasse B. Somit ist dieser Datensatz sehr ungleichmäßig verteilt. Bildet man mittels CART-Algorithmus einen Entscheidungsbaum, wird der Baum so lange wachsen, bis das eine Individuum von den restlichen 99 Individuen getrennt ist. Folglich gibt es in diesem Fall genau einen Endknoten, der für die Klasse A stimmt, die restlichen Endknoten stimmen für die Klasse B. Dieser Baum weist auch eine eigentümliche Gestalt auf, da er nur einseitig wächst. Das heißt, jeder Knoten besitzt einen Endknoten als Folgeknoten, der für die Klasse B stimmt. Die zu treffenden Entscheidungen sind somit den Individuen der Klasse B angepasst. Daraus resultiert nun, dass die Klassifizierung von Individuen der kleineren Klasse bei einem Entscheidungsbaum oft fehlerhaft ist.

Bildet man nun auf einem unbalancierten Datensatz einen Random Forest, folgt aus obiger Überlegung, dass auch hier dieses Problem auftritt. Der folgende Versuch soll diese These mit einem unbalancierten Datensatz verdeutlichen. Es wird mit Hilfe der "randomForest" Funktion ein Random Forest Modell auf einem unbalancierten Datensatz erzeugt.

Der dafür verwendete Datensatz $\mathbb{X}$ stellt eine von der Bank telefonisch durchgeführte Umfrage dar. Die Bank möchte wissen, ob ein Kunde Interesse hat (Antwortvariable $\mathbb{Y} := \{Yes, No\}$), eine Termineinlage bei ihr zu tätigen.

Der Datensatz besteht aus 45.211 Individuen (Kunden) mit jeweils 16 Merkmalen. Diese Merkmale geben unter anderem Auskunft über das Alter, den Familienstand, den Job, die jährliche Bilanz, die Ausbildung, usw. Im Datensatz befinden sich nur 5.289 Kunden, die mit einer Termineinlage einverstanden wären ("Yes"-Klasse). Somit hat die "Yes"-Klasse einen Anteil von etwa 11% des Datensatzes. Für weitere detaillierte Informationen siehe Moro et al. (2014).

In den folgenden Modellen wird der Datensatz in eine Trainings- ($\mathbb{X}_{Train}$) und eine Testmenge ($\mathbb{X}_{Test}$) geteilt (Verhältnis: 2/3 Trainingsmenge, 1/3 Testmenge). Um das Verhältnis der Klassenzugehörigkeiten nicht zu ändern, wird bei der Konstruktion des Trainingsdatensatzes darauf geachtet, dass man jeweils 2/3 der Individuen von der "No"-Klasse bzw. der "Yes"-Klasse ohne Zurücklegen zieht. Mit n_{yes} bzw. n_{no} bezeichnet man im Folgenden die Anzahl der Individuen der "Yes"- bzw. der "No"-Klasse des Trainingsdatensatzes.

Um die Ergebnisse anschaulich darzustellen, werden sie in folgender Kontingenztabelle wiedergegeben:

		Wahre Klasse	
	$Klassen$	"No"	"Yes"
Ergebnis	"No"	$\#RN$	$\#FN$
	"Yes"	$\#FY$	$\#RY$

Die Zeilen stellen jeweils das Ergebnis der Klassifikation dar und die Spalten die wahre Klasse. $\#RN$ bzw. $\#RY$ geben die Anzahl der Individuen der "No"-Klasse bzw. "Yes"-Klasse an, die richtig klassifiziert wurden. Hingegen geben $\#FN$ bzw. $\#FY$ die Anzahl der Individuen an, die falsch klassifiziert wurden. Aus einer Kontingenztabelle lässt sich der Missklassifikationsfehler Err und die relative Treffergenauigkeit $TQ(Yes)$ der "Yes"-Klassen folgendermaßen berechnen:

$$\begin{aligned} Err &:= 1 - \left(\frac{\#RN + \#RY}{\#RN + \#FN + \#FY + \#RY} \right) \\ TQ(Yes) &:= \frac{\#RY}{\#RY + \#FY} \end{aligned}$$

Wenn man einen Random Forest auf dem Trainingsdatensatz trainiert und mittels Testdatensatz testet, erhält man folgende Kontingenztabelle als Ergebnis:

1.Versuch		Wahre Klasse	
	$Klassen$	"No"	"Yes"
Ergebnis	"No"	12.847	461
	"Yes"	947	816

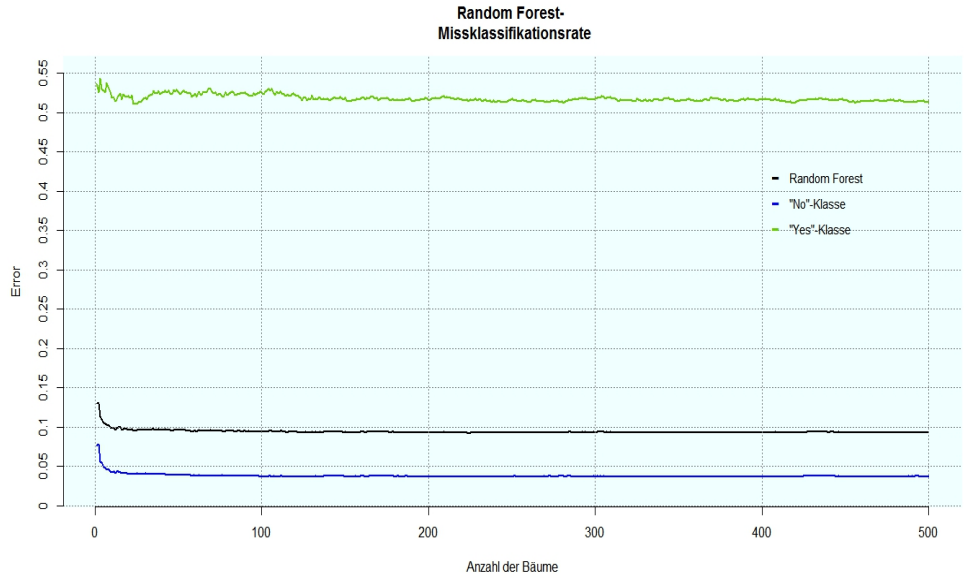

Abbildung 3.2: Reduzierung des Missklassifikationsfehlers durch Erhöhung der Baumanzahl des Random Forest

Der Missklassifikationsfehler beträgt in etwa $\approx 9\%$. Obwohl der Missklassifikationsfehler klein ist, ist die Treffergenauigkeit für die Individuen der "Yes"-Klasse in etwa $\approx 46\%$. Dies ist jedoch nicht im Interesse der Bank, da diese auf keinen Fall potenzielle Kunden verlieren möchte.

Die Abbildung 3.2 zeigt den Missklassifikationsfehler Err und den Missklassifikationsfehler der beiden Klassen $(1 - TQ(Yes)$ bzw. $1 - TQ(No))$ in Abhängigkeit von der Anzahl der Bäume des Random Forest. Wie in dieser Abbildung ersichtlich ist, bewirkt der geringe Missklassifikationsfehler der "No"-Klasse, dass auch der Random Forest eine geringe Missklassifikationsrate aufweist, obwohl dieser die "Yes"-Klasse in den meisten Fällen ($\approx 54\%$) falsch klassifiziert. Daraus lässt schließen, dass eine Erhöhung der Baumanzahl auch keine Verbesserung der Treffergenauigkeit für die "Yes"-Klasse bewirkt.

Aus diesem Grund muss man sich Methoden überlegen, bei denen der Random Forest bei der Klassifizierung nicht die größeren Klassen bevorzugt. Dazu werden die folgenden drei Methoden vorgestellt (vgl: Chen et al. (2004)):

1. Die erste Methode ist eine naive aber mit wenig Aufwand verbundene Vorgehensweise. Bevor man einen Random Forest auf einem Trainingsdatensatz trainiert, wird der Trainingsdatensatz noch folgendermaßen geändert: Man stellt ein gleichmäßiges Klassenverhältnis her, indem man aus jeder Klasse n_{yes} Individuen ohne Zurücklegen zieht. Das heißt, man erhält eine Teilmenge $\mathbb{X}_{neu} \subset \mathbb{X}_{Train}$, und auf dieser wird

dann ein Random Forest trainiert. Ein Problem an dieser Methode besteht darin, dass man einen Informationsverlust in Kauf nimmt, wenn wichtige Repräsentanten der Klasse nicht gezogen werden.

2. Eine weitere Methode ist die des "Under Sample". Es wird der Bagging-Algorithmus (vgl. Abschnitt 3.1.1) dahingehend geändert, dass man aus jeder Klasse $n = n_{yes}$ Individuen $X_i \in \mathbb{X}_{train}$ mit Zurücklegen zieht. Dadurch erhält jeder Entscheidungsbaum einen individuellen Trainingsdatensatz mit gleich vielen Repräsentanten pro Klasse. Somit wird gewährleistet, dass jeder Baum auch andere Individuen zur Verfügung hat. Der Vorteil dieses Ansatzes ist der, dass jedes Individuum gezogen werden kann und so das Problem des Informationsverlustes umgangen wird.

3. Die Methode des "Over Sample" bedeutet hingegen: Man zieht $n = n_{no}$ Individuen mit Zurücklegen aus den kleineren Klassen, bis alle Klassen die Anzahl der Individuen der "No"-Klasse erreicht haben. Diese Methode ist allerdings sehr rechenintensiv, weil man vor jeder Konstruktion eines Entscheidungsbaumes eine große Menge an Individuen zu jeder Klasse ziehen muss.

Der Unterschied zwischen der ersten Methode und der des "Under Sample" besteht darin, dass beim "Under Sample" für jeden Entscheidungsbaum eine Bootstrap Stichprobe aus dem gesamten Trainingsdatensatz gezogen wird. Hingegen wird bei der ersten (naiven) Methode zuerst der Trainingsdatensatz verkleinert, sodass ein gleiches Klassenverhältnis der Individuen besteht und auf dieser Teilmenge wird ein Random Forest trainiert. Dadurch werden schon im vorhinein Individuen ausgeschlossen und diese auch nicht für die Konstruktion der Bäume verwendet. (vgl. Abschnitt 3.2.1).

An den folgenden drei Random-Forest-Modellen werden die drei Methoden getestet. Auch in diesen Modellen werden zwei Drittel des Datensatzes für die Trainingsmenge herangezogen und auf der restlichen Menge werden die Modelle getestet. Um die Ergebnisse vergleichbar zu machen, werden die Modelle auf der identen Trainingsmenge trainiert, sodass diese auch auf der identen Testmenge getestet werden können. Die Trainingsmenge hat 31.903 Individuen, bevor man eine der Methoden darauf anwendet. Davon gehören 3.526 Individuen der "Yes"-Klasse an ($n_{yes} = 3.526$). Es wird für die in R implementierte "set.seed" Funktion "10101" als Parameter gewählt.

- Im Modell 1 wird die erste (naive) Methode angewandt. Die Trainingsmenge, auf der der Random Forest aufgebaut wird, hat alle 3.526 Individuen der "Yes"-Klasse zur Verfügung und 3.526 Individuen der "No"-Klasse werden ohne Zurücklegen gezogen.

- Im Modell "Under Sample" werden für jeden Baum $n = n_{yes}(= 3.526)$ Individuen der "Yes"-Klasse bzw. der "No"-Klasse mit Zurücklegen gezogen. Das heißt, es wird so oft ein Individuum jeder Klasse mit

Zurücklegen gezogen, wie die Anzahl der Individuen der "Yes"-Klasse ursprünglich betrug.

- Bei dem Modell "Over Sample" werden jeweils $n = n_{no}(= 26.614)$ mal Individuen der kleineren Klasse mit Zurücklegen gezogen. In diesem Fall wird so oft ein Individuum jeder Klasse mit Zurücklegen gezogen, wie die "No"-Klasse ursprünglich hatte.

Jedes dieser Random Forest Modelle besteht aus 500 Entscheidungsbäumen. In der unten angeführten Tabelle sind die Ergebnisse der Modelle ersichtlich, wenn man diese mit dem jeweils identen Testdatensatz testet:

Methode	Modell 1		"Under Sample (mZ)"		"Over Sample"	
Kontingenz	N	Y	N	Y	N	Y
N	10.945	2.363	11.402	1.906	11.386	1.922
Y	152	1.611	222	1.541	213	1.550
Error	$\approx 16,6\%$		$\approx 14,1\%$		$\approx 14.1\%$	
TQ(Yes)"	$\approx 91,3\%$		$\approx 87,4\%$		$\approx 87,9\%$	
Rechenzeit	$37\,sec$		$1\,min\ 38\,sec$		$8\,min\ 9\,sec$	

Die Rechenzeit wurde auf einem Intel Core i3 mit 2,4 GHz bemessen und gibt die benötigte Zeit an, um den Random Forest zu erstellen. Der Missklassifikationsfehler ist in allen drei Modellen größer als im allerersten Versuch, bei dem er nur $\approx 9\%$ betrug. Allerdings haben diese Modelle allesamt eine viel bessere Treffergenauigkeit für die Individuen der "Yes"-Klasse. Somit erhält die Bank aus den Ergebnissen aller drei Modelle eine aussagekräftigere Erkenntnis als beim ersten Versuch, wo die Trefferquote für die "Yes"-Klasse nur bei $\approx 46\%$ lag. An diesem Ergebnis ist allerdings verwunderlich, dass die erste (naive) Methode das beste Ergebnis für die Treffergenauigkeit der "Yes"-Klasse liefert.

In den nächsten Modellen werden die Individuen im Gegensatz zu den vorangegangenen Modellen "Under Sample" und "Over Sample" ohne Zurücklegen gezogen. Bei der Methode des "Under Sample oZ" werden wieder $n = n_{yes}(= 3.526)$ Individuen pro Klasse verwendet. Das heißt, jeder Entscheidungsbaum hat die gesamte Information der "Yes"-Klasse zur Verfügung. Allerdings nimmt man in diesem Fall eine höhere Missklassifikationsrate in Kauf. Die Begründung dafür ist, dass die Trainingsmenge jedes Baumes aus den selben Individuen der "Yes"-Klasse besteht. Folglich ist die Varianz zwischen den Bäumen höher (vgl. Abschnitt 3.1.1 und Gleichung 3.6 auf Seite 23), was zu einer höheren Missklassifikationsrate des Random Forest führen kann (vgl: 3.3 auf Seite 21).

Bei der Methode des "Over Sample 2" werden die Individuen der "Yes"-Klasse mit und die der "No"-Klasse $n = n_{no}$ ohne Zurücklegen gezogen. Die folgende Tabelle zeigt wieder die aus dem Testdatensatz erhaltenen Ergebnisse:

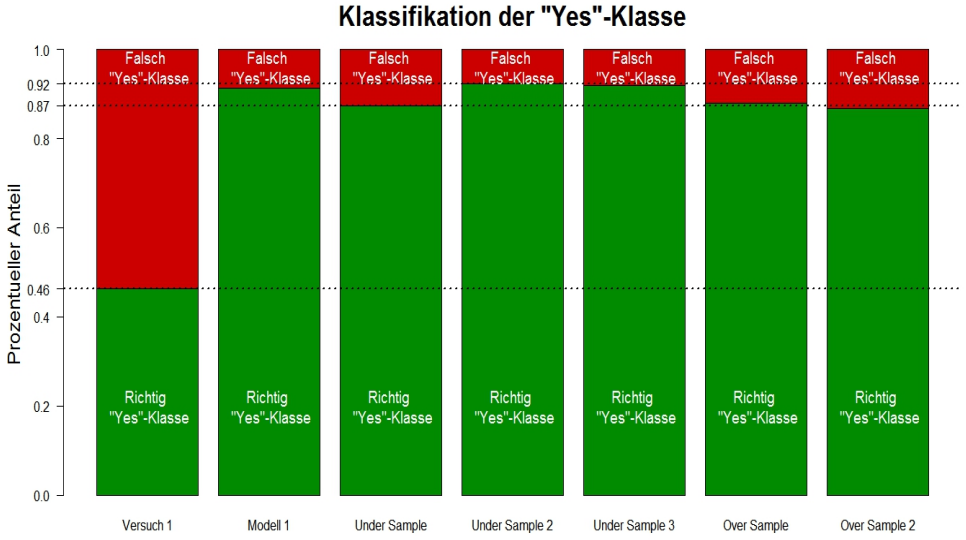

Abbildung 3.3: Zeigt den prozentuellen Anteil, wie viele Individuen der "Yes"-Klasse richtig bzw. falsch klassifiziert wurden

Methode	"Under Sample oZ"			"Over Sample 2"		
Kontingenz		N	Y		N	Y
	N	10.935	2.373	N	11.465	1.843
	Y	136	1.627	Y	233	1.530
Error	$\approx 16,6\%$			$\approx 13,7\%$		
TG "Yes"	$\approx 92,3\%$			$\approx 86,8\%$		
Rechenzeit	$1\,min\ 53\,sec$			$7\,min\ 41\,sec$		

Die Methode des "Under Sample oZ" weist die beste Treffergenauigkeit auf. Auch der Missklassifikationsfehler, mit Ausnahme des ersten Versuchs ($\approx$ 9%), ist nicht wesentlich höher als bei den anderen Modellen. Verglichen mit dem Modell 1 benötigt die Methode des "Under Sample oZ" aber mehr Rechenaufwand. Wenn man die beiden "Under Sample" und die beiden "Over Sample" Methoden vergleicht, fällt unschwer auf, dass "Over Sample" eine viel höhere Rechenleistung benötigt und die Treffergenauigkeit für die Individuen der "Yes"-Klasse niedriger ist als beim "Under Sample".

Der letzte Versuch baut auf der vorher dokumentierten Methode des "Under Sample oZ" auf. Um den Missklassifikationsfehler und den Rechenaufwand zu verringern, besteht die Idee, dass die Bäume für die Konstruktion nur mehr 50% vom vorangegangenen Trainingsdatensatz zur Verfügung haben. Bevor jeder einzelne Baum erstellt wird, werden zufällig und ohne Zurücklegen $n = 1.763 \approx {}^{n_{yes}}/_{2}$ Individuen der "Yes"- und 1.763 Individuen der "No"-Klasse gezogen. Diese Methode liefert folgendes Ergebnis:

Methode	"Under Sample oZ" (50%)		
Kontingenz		N	Y
	N	10.880	2.428
	Y	143	1.620
Error	$\approx 17,1\%$		
TG "Yes"	$\approx 91,9\%$		
Rechenzeit	$38\,sec$		

Die erhoffte Verringerung des Missklassifaktionsfehlers ist nicht eingetreten. Bei nun etwas schlechterem Ergebnis hat sich allerdings der Rechenaufwand doch markant verringert.

In Abbildung 3.3 sind die Ergebnisse der "Yes"-Klasse für alle Modelle wiedergegeben. Aus dieser Grafik ist klar erkennbar, dass die Bank die beste Treffergenauigkeit für die Individuen der "Yes"-Klasse erhält, wenn sie die Methode des "Under Sample oZ" auf dem Random Forest anwendet.

Generell gilt: Wenn man einen Random Forest auf einem unbalancierten Datensatz trainieren möchte und vor allem an Individuen der kleineren Klasse interessiert ist, ist es sinnvoll, eine der zuvor erwähnten Methoden anzuwenden, um auch ein zufriedenstellendes Ergebnis zu erhalten.

3.6.2 Güte bei fehlenden Werten

In diesem Abschnitt wird getestet, welche Auswirkungen fehlende Werte auf das Ergebnis haben. Für Klassifizierungsmethoden sind fehlende Werte oft ein Problem. Wenn man eine neue Beobachtung mit Hilfe eines Entscheidungsbaumes klassifizieren möchte, jedoch bei einer zu treffenden Entscheidung ein Merkmalswert fehlt, ist eine Klassifizierung nicht möglich. Man kann beim Verzweigungspunkt nicht eruieren, zu welchem der beiden Folgeknoten man gelangt.

Der hier verwendete Datensatz Moro et al. (2014) ist ident mit dem im vorhergehenden Abschnitt. Eine Bank will wissen, ob ein Kunde Interesse hat, eine Termineinlage bei ihr zu tätigen. Die Vorgangsweise ist nun die, dass Schritt für Schritt und zufällig Einträge des Datensatzes gelöscht werden und auf diesem Datensatz die im Abschnitt 3.5.2 vorgestellten Methoden angewendet werden. Um die Ergebnisse zu vergleichen, werden beide Algorithmen ("$na.roughfix$","$rfImpute$") jeweils auf dem selben Trainingsdatensatz angewand und auf dem selben Testdatensatz getestet. Weiters wird, um Rechenzeit einzusparen, die Trainingsmenge wie im Abschnitt 3.6.1 unter der "naiven" Methode verändert. Das heißt, es werden für den Trainingsdatensatz zufällig und für jede Klasse gleich viele Beobachtungen ohne Zurücklegen gezogen.

Beim ersten Schritt werden für jedes Merkmal 1% der Einträge (=53) gelöscht, beim nächsten 5%, dann 10%, 15%, 20%, ..., 50%. Es werden bei jedem Merkmal gleich viele Einträge gelöscht. Damit vesucht man zu verhindern, dass das Ergebnis bzw. der Missklassifikationsfehler verfälscht wird, wenn zum

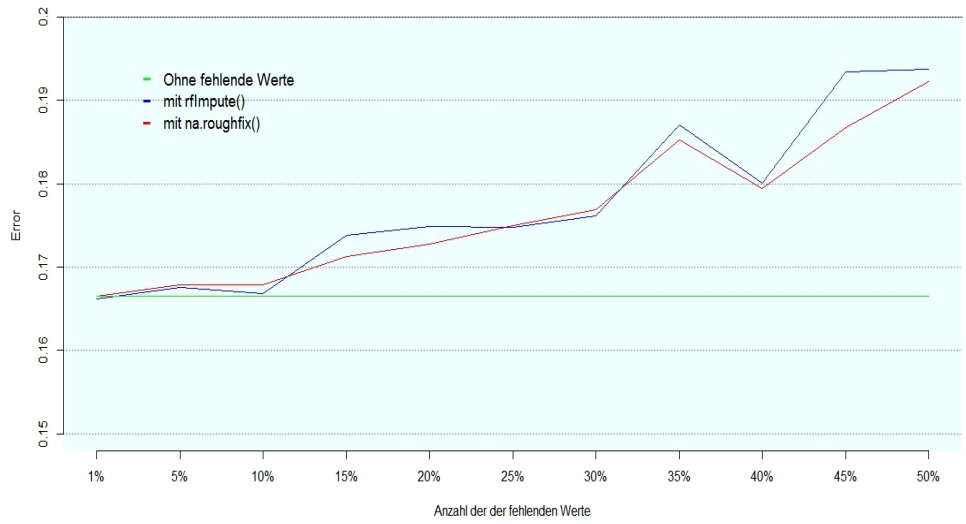

Abbildung 3.4: Güte bei fehlenden Werten. In der X-Achse ist die Anzahl der fehlenden Werte eingetragen und in der Y-Achse der Missklassifikationsfehler (Error) für die unterschiedlichen Funktionen.

Beispiel bei einem wichtigen Merkmal mehrere Einträge fehlen. Dazu werden 11 Random Forests aufgebaut und diese auf dem selben Testdatensatz angewandt. Es wird der im Abschnitt 3.5.2 vorgestellte Algorithmus "$rfImpute$" mit jeweils 5 Iterationsschritten getestet. Um die Rechenleistung nochmals zu verringern, werden die unter Iterationsschritt 2 des Algorithmus neu aufgebauten Random Forests jeweils aus nur 300 Bäumen bestehen. In Abbildung 3.4 sind nun die Missklassifikationsfehler bei unterschiedlicher Anzahl von fehlenden Werten dargestellt.

Man kann erkennen, dass die Missklassifikationsfehler für die beiden Funktionen "$rfImpute$" und "$na.roughfix$" erst bei etwa 20% fehlender Werte zunehmen. Auch ist in der Abbildung gut ersichtlich, dass bei einem Datensatz von 15%, 20% und 40% fehlender Werte der Missklassifikationsfehler (Error) sogar mit der "$na.roughfix$" etwas niedriger liegt, als bei der "$rfImpute$" Funktion. An dieser Stelle ist auch noch anzumerken, dass die Rechenleistung für die "$rfImpute$" Funktion eindeutig höher liegt. Dies ist aber auch wenig verwunderlich, da man für jeden Iterationsschritt einen ganzen Wald samt seiner Proximitywerte erstellen muss. (vgl. Abschnitt 3.5.2)

Zum Vergleich: Die "$rfImpute$" Funktion benötigte im Schnitt 10 Minuten und 52 Sekunden, hingegen die "$na.roughfix$" Funktion im Schnitt nur 0,2 Sekunden, um die fehlenden Werte zu ersetzen.

3.6.3 Bias in der Merkmalswichtigkeit

Alle Merkmalswichtigkeiten, die in den Abschnitten 3.4.2. und 3.4.3 beschrieben wurden, sind unter gewissen Umständen nicht erwartungstreu. Der Gini-

Index, das ist das im CART-Algorithmus verwendete Maß für die Heterogenität (2.2), bevorzugt Merkmale mit mehreren Kategorien. Strobl et al. (2007) hat empirisch nachgewiesen, dass Merkmale mit mehreren Kategorien öfter als Verzweigungspunkt eines Baumes verwendet werden, als Merkmale mit wenigen Kategorien.

Sie zeigte dieses Problem anhand zweier Modelle auf. Im ersten Modell korrelierten keine Merkmale mit der Antwortvariablen und es stellte sich heraus, dass die Merkmale mit mehreren Kategorien gegenüber solchen mit wenigen Kategorien bevorzugt wurden.

Im zweiten Modell korrelierte ein Merkmal mit der Antwortvariablen. Dieses Merkmal besaß nur zwei Kategorien. Selbst in diesem Fall wurde die Variable mit fünf Kategorien bevorzugt. Ein Grund dafür liegt in der Tatsache, dass die Anzahl der Möglichkeiten für einen Verzweigungspunkt exponentiell mit der Anzahl der Merkmale steigt. Deswegen können Entscheidungen nahe am Stamm des Baumes eine große Trennschärfe bei Merkmalen mit vielen Kategorien bewirken. Auch dann, wenn diese nur wenig für die Antwortfindung beitragen.

Die Bevorzugung der Merkmale mit mehreren Kategorien betrifft vorerst nur die Gini-Wichtigkeit und die der Verwendungswichtigkeit. Aber indirekt ist auch die permutierte Merkmalswichtigkeit davon betroffen, denn in den ersten Schnittpunkten werden oft Merkmale mit mehreren Kategorien verwendet. Strobl et al. (2008) machte auf ein weiteres Problem aufmerksam. Sie wies darauf hin, dass der CART-Algorithmus korrelierte Merkmale gerade in den ersten Verzweigungspunkten eines Baumes öfters verwendet als unkorrelierte. Eine Permutation nahe am Stamm des Baumes verursacht eine größere Veränderung als Entscheidungen, die kurz vor den Endknoten getroffen werden. Die Begründung liegt darin, dass die Entscheidungen am Ende des Baumes nur einen kleinen Teil der Beobachtungen beeinflussen.

Unter gewissen Bedingungen muss man den Wert der permutierten Merkmalswichtigkeit aber auch kritisch betrachten. Zueinander korrelierte Merkmale erhalten einen höheren Wert als unkorrelierte Merkmale, auch wenn unkorrelierte Merkmale mehr zur Antwortfindung beitragen, also eine bessere Trennschärfe haben. Anhand folgendem Beispiel wird versucht, den Fehler ersichtlich zu machen.

Es wird ein Random Forest auf dem Datensatz "readingSkills" vom R-Package "party" konstruiert (set.seed(1984)). Dieser Datensatz enthält 200 Schulkinder zwischen 5 und 11 Jahren mit den folgenden drei Merkmalen:

- "Muttersprache": Ist die Sprache die Muttersprache des Kindes?
  - 1: Wenn das Kind die Sprache als Muttersprache erlernt hat
  - 0: Sonst
- "Alter": Werte zwischen 5 - 11, welche das Alter des Kindes angeben

- "Schuhgröße": Werte zwischen 23 - 32, welche die Schuhgröße des Kindes angeben

Die Zielvariable y ist ein Wert zwischen 0 und 100 und stellt die Lesefähigkeit des Kindes dar. Wobei 0 bedeutet, dass das Kind nicht lesen kann und 100 bedeutet, dass es perfekt lesen kann. Die Kinder werden in 4 Klassen eingeteilt, um sie anhand der drei Merkmale nach ihrer Lesestärke zu klassifizieren.

Die Vermutung liegt nahe, dass die Merkmale "Muttersprache" und vor allem "Alter" die erklärenden Variablen für die Lesefähigkeit darstellen. Wenn man nun mit Hilfe der "randomForest()" Funktion, welche im R-Package "randomForest" implementiert ist, ein Klassifizierungsmodell erstellt, erhält man folgende Merkmalswichtigkeiten:

Merkmal	Muttersprache	Alter	Schuhgröße
perm Wichtigkeit	38.7	43.6	26.3
Gini-Wichtigkeit	11.5	57.1	34
Verwendung	0.24	0.37	0.38

Das Merkmal "Alter" bekommt die höchste permutierte Merkmalswichtigkeit (perm Wichtigkeit) zugewiesen. Das ist auch wenig verwunderlich, da ein älteres Kind sicherlich besser lesen kann als ein jüngeres. Etwas merkwürdig an diesem Ergebnis ist allerdings, dass die Schuhgröße bei allen drei Maßen eine hohe Wichtigkeit erhält. Schuhgröße und Alter sind zwar stark korreliert mit $r = 0.90$, aber intuitiv sollte das Merkmal "Muttersprache" und auch das Merkmal "Alter" ein viel besserer Indikator für die Beschreibung der Lesefähigkeit sein, als es das Merkmal "Schuhgröße" ist.

Um nun diese Intuition nochmals zu hinterfragen, wird jeweils ein Merkmal aus dem Datensatz "readingSkills" herausgenommen. Auf diesem Datensatz, der nun ein Merkmal weniger enthält, wird ein Random Forest aufgebaut. Je wichtiger ein Merkmal ist, desto größer sollte auch der Out-of-Bag Error sein. Wenn die obigen Merkmalswichtigkeiten stimmen, sollte das Ergebnis nur leicht schwanken. In der folgenden Tabelle wird der Out-of-Bag-Error dargestellt, wenn wie oben beschrieben jeweils ein Merkmal ausgeklammert wird:

fehlendes Merkmal	Muttersprache	Alter	Schuhgröße	keines
"OoB" Error	21.5%	34.5%	2%	5%

Diese Tabelle bestätigt nun, dass das Merkmal "Schuhgröße" kein wichtiger Indikator ist, um auf die Lesefähigkeit zu schließen. Wenn man einen Random Forest ohne Merkmal "Schuhgröße" bildet, ist sogar der Missklassifikationsfehler geringer, als wenn man den kompletten Datensatz heranzieht. Jetzt stellt sich die Frage, warum dieses Merkmal trotzdem einen so hohen Wert bei der Merkmalswichtigkeit erhält. Für ein weiteres Beispiel siehe Strobl et al. (2007, 2008).

Der Grund für die Überschätzung korrelierter Merkmale liegt in einer falschen Wahl der Null-Hypothese, die auf die Baumstruktur zurückzuführen ist. Die eigentliche Fragestellung lautet: Ist Merkmal x^j korreliert mit Y? Darum will man folgende Nullhypothese testen:

$$\mathcal{H}_0 : x^j \perp Y$$

Ein Baum wird rekursiv konstruiert, das heißt, alle zu treffenden Entscheidungen sind von den bereits zuvor getroffenen Entscheidungen abhängig. Wenn man einen Baum wachsen lässt, wird für den Verzweigungspunkt im Knoten K_t jenes Merkmal gewählt, welches die größte Trennschärfe unter den Klassen in den beiden Folgeknoten K_t^L und K_t^R bewirkt. Aus diesem Grund sind die nächsten beiden Entscheidungen in den Knoten K_t^L und K_t^R abhängig von der zuvor getroffenen. Die Suche nach der besten Entscheidung bezieht sich auf die Datenmenge, die resultiert, wenn man den Datensatz nach den vorangegangenen Entscheidungen trennt. Wenn dort anstatt Merkmal x^j ein anderes Merkmal x^i verwendet wird, kann man davon ausgehen, dass auch im Folgeknoten ein anderes Merkmal für die Entscheidungsfindung verwendet wird.

Ein Beispiel: Ein Datensatz besteht aus Individuen, die jeweils einer von vier Klassen $\{A, B, C, D\}$ angehören. Eine am Merkmal x^i getroffene Entscheidung bewirkt, dass in den beiden Folgeknoten die Individuen der Klassen $\{A, B\}$ von den Individuen der Klassen $\{C, D\}$ getrennt sind. Hingegen bewirkt eine auf dem Merkmal x^j getroffene Entscheidung, dass in den Folgeknoten die Individuen der Klassen $\{A, B, D\}$ mit den restlichen Individuen der Klassen $\{C\}$ getrennt sind. Somit kann man davon ausgehen, dass auch die Merkmale, die für die Entscheidungsfindung in den Folgeknoten in Betracht gezogen werden, nicht die selben sind.

Bei der permutierten Merkmalswichtigkeit will man herausfinden, ob das Merkmal x^j unabhängig von der Zielfunktion Y ist. Man permutiert die Werte aller Individuen des Merkmales x^j. Dabei wird allerdings nicht berücksichtigt, an welchem Knotenpunkt x^j zur Entscheidungsfindung verwendet wird. Man muß aber beachten, dass die im Knotenpunkt vorhandenen Individuen abhängig von den zuvor getroffenen Entscheidungen sind. Wenn nun der Fall eintritt, dass zwei Entscheidungen hintereinander auf korrelierten Merkmalen zu treffen sind, wird eine Permutation am Merkmal x^j der ersten Entscheidung auch große Veränderungen für die folgende bewirken.

Hier kann man wieder den Datensatz "ReadingSkills" in Betracht ziehen. Angenommen, die erste Entscheidung wird auf dem Merkmal "Schuhgröße" getroffen und die nächste am Merkmal "Alter". Die erste Entscheidung bewirkt, dass die Individuen, die nicht besonders gut lesen können, von den guten Lesern getrennt werden. Eine Entscheidung am Merkmal "Alter" trennt dann die verbliebenen Individuen nach den restlichen Klassen. Weil aber diese beiden Merkmale korreliert sind, bewirkt eine Permutation auf dem Merkmal "Schuhgröße", dass auch die Entscheidung am Merkmal "Alter" stark von

dieser Permutation betroffen ist, weil wieder alle Individuen der unterschiedlichen Klassen vorkommen. Folglich simuliert eine Permutation des Merkmals "Schuhgröße" nicht nur das Fehlen dieses Merkmals, sondern auch das Fehlen eines korrelierten Merkmals im Folgeknoten. Deswegen löst eine Permutation des Merkmals "Schuhgröße" auch einen hohen Missklassifikationsfehler aus, obwohl dieses Merkmal wenig zur Antwortfindung beiträgt.

Daher lautet die eigentliche Null-Hypothese: Ist das Merkmal x^j unabhängig von Y und auch von allen anderen Merkmalen $Z := (x^1, ..., x^{j-1}, x^{j+1}, ..., x^q)$ (vgl. Strobl et al. (2008)). Somit wird auf folgende Null-Hypothese getestet:

$$\mathcal{H}_0 : x^j \perp Y \wedge x^j \perp Z$$

Die Null-Hypothese wird verworfen, wenn die hier beschriebenen drei Umstände auftreten:

- x^j ist sowohl von y als auch von Z abhängig. In diesem Fall wird die permutierte Merkmalswichtigkeit von x^j nicht überschätzt.
- x^j ist abhängig von y aber unabhängig von Z. Auch in diesem Fall wird die permutierte Merkmalswichtigkeit von x^j nicht überschätzt.
- x^j ist abhängig von Z aber unabhängig von y. In diesem Fall wird die permutierte Merkmalswichtigkeit von x^j überschätzt.

3.6.4 Conditional Inference Tree (bedingter Inferenz Baum)

Aufgrund der Tatsache, dass das Reinheitsmaß (Gini-Index) einerseits Merkmale mit mehreren Kategorien und andererseits zueinander korrelierte Merkmale bevorzugt, sollte man vor allem bei Datensätzen mit korrelierten Merkmalen einen anderen Zugang wählen. Um den Umfang dieser Arbeit aber in Grenzen zu halten, wird in diesem Abschnitt nur ein grober Überblick über den Algorithmus des Conditional Inference Tree geliefert.

Dieser wurde von Hothorn et al. (2006) , Hornik und Zeileis in ihrer Arbeit "Unbiased Recursive Partitionierung: A Conditional Inference Framework" beschrieben und veröffentlicht. Sie verwendeten einen neuen Ansatz zur Suche nach jenem Merkmal, welches am wirksamsten für die richtige Antwortfindung ist. Dabei wird nicht mehr nach einem Merkmal gesucht, das eines der Unreinheitsmaße in den Folgeknoten minimiert.

Bei einem Conditional Inference Tree werden jene Merkmale für das Wachstum des Baumes in Betracht gezogen, die zur Antwortvariablen am höchsten korreliert sind. Für die Konstruktion aber auch für die Suche nach dem besten Verzweigungspunkt wird ein Hypothesen-Test verwendet.

Die Nullhypothese lautet: Ist das Merkmal x^j unabhängig von der Zielvariablen Y? Wenn dies der Fall ist, ist die Verteilung der Zielvariablen $D(Y)$ und die durch x_j bedingte Verteilung $D(Y \mid x^j)$ die gleiche.

Somit gilt:

$$\mathcal{H}_0^j : D(Y) = D(Y \mid x^j).$$

Folglich wird unter der globalen Nullhypothese getestet, ob alle Merkmale unabhängig von Y sind, was formell wiederum bedeutet:

$$\mathcal{H}_0 = \cap_{j=1}^{m} \mathcal{H}_0^j \ \ bzw.$$

$$D(Y) = D(Y \mid (x^1, x^2, ..., x^m))$$

An jedem Knotenpunkt wird für deren Beobachtungen mittels permutiertem Testverfahren getestet, ob die globale Nullhypothese an diesem Knotenpunkt anzunehmen ist. Wenn dies zutrifft, das heißt, wenn keines der Merkmale einen Einfluss auf die Antwortfindung hat, stellt der Knoten einen Endknoten dar und wird im Weiteren nicht mehr unterteilt. Wenn allerdings die globale Nullhypothese verworfen wird, wird jenes Merkmal als Verzweigungspunkt verwendet, welches bei der permutierten Teststatistik den kleinsten $P-Wert$ aufweist.

Ein Entscheidungsbaum wird auf einer Lernmenge L_n mit $n \in \mathbb{N}$ Beobachtungen konstruiert. Jeder Knotenpunkt K_t bekommt einen Vektor mit Gewichten $w_t := (w_{t,1}, w_{t,2}, ..., w_{t,n})$, mit $w_{t,i} \geq 0 \wedge w_{t,i} \in \mathbb{N}, \forall i \in \{1, ..., n\}, \forall t$ zugewiesen. Dieser Vektor $w_{t,\cdot}$ gibt Auskunft, welche der n Beobachtungen im Knotenpunkt K_t vorkommen. Ein Eintrag $w_{t,i}$ des Vektors w_t ist nur dann 0, wenn die Beobachtung X_i nicht im Knotenpunkt K_t vorkommt. Zusätzlich kann man bei dieser Vorgangsweise auch jeder Beobachtung eine Gewichtung zuweisen, indem man "wichtigen" Beobachtungen einen höheren Wert zuteilt als "unwichtigen".

Mit folgendem Algorithmus wird ein Conditional Inference Tree erzeugt (vgl. Hothorn et al. (2006)):

1. Unter Berücksichtigung der Gewichte w_t des Knotenpunktes K_t wird die globale Nullhypothese $\mathcal{H}_0$ getestet.

   (a) Wenn die Nullhypothese nicht zu verwerfen ist, d. h. wenn keines der Merkmale Aufschluss auf das Ergebnis gibt, so stellt dieser Knotenpunkt einen Endknoten dar.

   (b) Wenn die globale Nullhypothese aber verworfen wird, so betrachte jenes Merkmal als Entscheidungsmerkmal, welches bezüglich der Teststatistik den kleinsten $P - Wert$ hat.

2. Als Entscheidungskriterium wählt man nun eine Teilmenge $A^* \subset x^j$. Anhand dieser Teilmenge wird dann entschieden, ob ein Merkmal des Individuums in dieser Teilmenge liegt oder nicht. Das heißt, der Knoten teilt sich in zwei Folgeknoten, die wiederum disjunkte Teilmengen besitzen.
   Die Gewichte der Folgeknoten lassen sich somit nun folgendermaßen bestimmen:
   $w_{links(t),i} := w_{t,i} I(x_i^j \in A^*)$ und $w_{rechts(t),i} := w_{t,i} I(x_i^j \notin A^*)$

3. Wiederhole die Schritte 1 bis 2 so lange, bis man die Nullhypothese unter Berücksichtigung der Gewichte bei jedem Knotenpunkt nicht mehr verwerfen kann.

Um in die exakte Vorgangsweise einzusehen, siehe Hothorn et al. (2006). In dieser Arbeit wird die verwendete permutierte Teststatistik hergeleitet und beschrieben. Außerdem wird noch erklärt, wie man die Teilmenge A^* zu wählen hat, damit die auf dieser Teilmenge basierende Entscheidung auch die beste ist.

3.6.5 Bedingte Merkmalswichtigkeit

Die Idee der bedingten Merkmalswichtigkeit (vgl: Strobl et al. (2008)) ist die, dass man nicht mehr die Werte eines Merkmales aller Individuen permutiert, sondern nur noch die Werte der Merkmale von ausgewählten Gruppen von Individuen. Um diese Gruppen zu bestimmen, wird ein Netz konstruiert. Es werden dann nur noch die Werte der Merkmale jener Individuen permutiert, die sich innerhalb der gleichen Grenzen dieses Netzes befinden.

Wie in Abbildung 2.1 auf Seite 6 ersichtlich ist, verursacht jeder Entscheidungsbaum eine Unterteilung des Merkmalsraumes. Jede Entscheidung unterteilt den noch im Knotenpunkt vorhandenen Merkmalsraum in zwei disjunkte Rechtecke. Es ist daher auch naheliegend, diese von einem Entscheidungsbaum erzeugte Unterteilung für die Definition des Netzes in Betracht zu ziehen. Der Vorteil dieser Vorgehensweise ist der, dass mit der Konstruktion eines Baumes automatisch ein zu diesem Baum gehöriges Netz mitbestimmt wird. Aus diesem Grund benötigt man keine weitere Methode, um das Netz zu konstruieren. Folglich wird sich auch die Rechenleistung in Grenzen halten.

Wenn man nun die von einem Baum erzeugte Merkmalswichtigkeit des Merkmals x^j bestimmen will, konstruiert man das vom Baum erzeugte Netz. Alle auf den restlichen Merkmalen $z := (x^1, x^2, ..., x^{j-1}, x^{j+1}, ..., x^m)$ getroffenen Entscheidungen erzeugen dieses Netz. Somit muss man zu jedem Merkmal ein separates Netz konstruieren.

Außerdem wird noch angenommen, dass jede getroffene Entscheidung den kompletten Merkmalsraum unterteilt. Mit Ausnahme der ersten Entscheidung am Stamm unterteilt bekanntlich jede weitere Entscheidung nur jeweils einen Teilraum des Merkmalsraumes (vgl. 2.1 auf Seite 6). Das heißt, dieses Netz ist feiner als ein von einem Entscheidungsbaum konstruiertes Netz. Durch ein feineres Netz wird die Anzahl der Individuen pro Gruppe kleiner und folglich werden die Werte eines Merkmales von wenigeren Individuen permutiert. Dies hat allerdings keine negativen Auswirkungen auf das Ergebnis (vgl: Strobl et al. (2008)).

Zur Bestimmung der Merkmalswichtigkeit wird wie in dem im Abschnitt 3.4.2 vorgestellten Algorithmus vorgegangen. Der einzige Unterschied besteht wie vorher beschrieben darin, dass nicht die Werte der Merkmale aller Individuen

Y	X_j	Z
y_1	$x_{\pi_j(1),j}$	z_1
$\vdots$	$\vdots$	$\vdots$
y_i	$x_{\pi_j(i),j}$	z_i
$\vdots$	$\vdots$	$\vdots$
y_n	$x_{\pi_j(n),j}$	z_n

Y	X_j	Z
y_1	$x_{\pi_{j\|Z=a}(1),j}$	$z_1 = a$
y_3	$x_{\pi_{j\|Z=a}(3),j}$	$z_3 = a$
y_{27}	$x_{\pi_{j\|Z=a}(27),j}$	$z_{27} = a$
y_6	$x_{\pi_{j\|Z=b}(6),j}$	$z_6 = b$
y_{14}	$x_{\pi_{j\|Z=b}(14),j}$	$z_{14} = b$
y_{21}	$x_{\pi_{j\|Z=b}(21),j}$	$z_{21} = b$
$\vdots$	$\vdots$	$\vdots$

Abbildung 3.5: Das neue und das alte Permutationsschema. Links ist das Vorgehen wie bei Breiman abgebildet und rechts die neue Vorgehensweise. Es werden nur noch die Werte der Individuen vertauscht, die sich innerhalb der selben Grenze des Netzes befinden (übernommen Strobl et al. (2008)).

permutiert werden, sondern nur die Werte der Individuen, die zur selben und vom Netz bestimmten Gruppe gehören. In Abbildung 3.5 ist die Idee nochmals dargestellt. Wie man sieht, werden nicht mehr alle Werte permutiert (links) sondern nur noch die von speziellen Gruppen.

Wenn man dieses Netz für eine Permutation verwendet, werden alle vom Baum getroffenen Entscheidungen berücksichtigt. Durch die Annahme, dass jede Entscheidung auch den ganzen Merkmalsraum teilt, macht es keinen Unterschied, ob ein Merkmal nahe am Stamm oder nahe am Endknoten als Entscheidungskriterium diente. Im Falle von korrelierten Merkmalen wird eine Permutation innerhalb des Netzes jene Merkmale aufspüren, welche die tatsächlich beste Trennschärfe auf die Antwortvariablen erzeugt.

Wenn zum Beispiel ein Merkmal als “unwichtig” erscheint, dieses aber mit einem “wichtigen” Merkmal korreliert ist, wird das Netz jene Entscheidungen berücksichten, die auf dem “wichtigen” Merkmal getroffenen wurden. Da das “wichtige” Merkmal eine hohe Trennschärfe auf die Antwortvariablen bewirkt, wird eine Permutation innerhalb des Netzes den Fehler nur geringfügig erhöhen. Wenn allerdings innerhalb eines Netzes die Grenzen der Entscheidungen von einem “wichtigen” Merkmal fehlen, wird eine Permutation einen höheren Missklassifikationsfehler nach sich ziehen.

Natürlich ist diese Methode mit mehr Aufwand verbunden, da man jedesmal ein neues Netz aufbauen muss. Damit sich die Rechenleistung bei der Berechnung der Merkmalswichtigkeit in Grenzen hält, kann man ein gröberes Netz konstruieren. Weil bei unkorrelierten Merkmalen die Merkmalswichtigkeit unverzerrt ist, kann man die auf diesen Merkmalen getätigten Entscheidungen außer Acht lassen. Wenn man die Merkmalswichtigkeit eines Merkmales bestimmen will, kann man ein Netz mit nur jenen Entscheidungen aufbauen, die an korrelierten Merkmalen getroffen wurden. Das heißt, für die Konstruktion des Netzes werden nur diejenigen Merkmale verwendet, die eine bestimmte

Schranke für den empirischen Korrelationskoeffizienten überschreiten.

Mit Hilfe der Funktion "cforest", die im R-Package "party" implementiert ist, kann man einen Zufallswald mit Conditional Inference Trees konstruieren. Die ebenfalls im Package enthaltene Funktion "varimp" berechnet die Merkmalswichtigkeiten. In der folgenden Tabelle sind die Ergebnisse der Merkmalswichtigkeiten des Reading-Skills-Datensatzes angegeben, die mittels der "cforest" und "varimp" Funktionen berechnet wurden.

	Muttersprache	Alter	Schuhgröße
Mit Netz	0.079	0.322	0.019
Ohne Netz	0.084	0.408	0.120

Wenn man ohne Verwendung eines Netzes permutiert, erhält das Merkmal "Schuhgröße" einen höheren Wert als das Merkmal "Muttersprache". Wenn man allerdings innerhalb des Netzes die Werte eines Merkmales permutiert, erhält man ein Ergebnis, welches bei der Beurteilung der Lesefähigkeit nun der Intuition nahe kommt, dass das Merkmal "Muttersprache" ein besserer Indikator sein sollte als das Merkmal "Schuhgröße".

Kapitel 4

Simulationen

In diesem Kapitel wird nun die in dieser Arbeit vorgestellte Methode des Random Forest analysiert. Es werden anhand eines Datensatzes mehrere Random Forests mit unterschiedlichen Parameterwerten aufgebaut. Darüber hinaus werden die Ergebnisse von zwei unterschiedlich aufgebauten Random Forest Verfahren (CART vs. Conditional Inference Tree) verglichen.

Bei diesem Datensatz sind die Klassen der Zielvariablen $\mathbb{Y}$ ungleichmäßig verteilt. Deswegen werden auf diesem unbalancierten Datensatz nochmals die im Abschnitt 3.6.1 vorgestellten Methoden des "Under Sample" bzw. "Over Sample" getestet.

Im ersten Abschnitt wird der verwendete Datensatz beschrieben. In diesem Zusammenhang wird auch erörtert, welche Informationen und welchen Nutzen man aus diesem Datensatz gewinnen kann. Im zweiten Abschnitt werden die Simulationen mittels der im "RandomForest" und der im "Party" Package implementierten Random Forest Funktionen durchgeführt. Weitere Graphiken dienen zur Veranschaulichung der Ergebnisse, der Eigenschaften der Daten sowie der angewandten Methoden. Aufgrund der Tatsache, dass ein Random Forest eine Art "Black Box" ist, wird im letzten Abschnitt versucht, die bisher vorliegenden Ergebnisse zu verbessern, indem man Parameterwerte variiert.

4.1 Daten

Seismische Stöße

Dieser Datensatz (von Sikora and Wrobel (2015)) behandelt das Thema seismischer Stöße in einer Kohlemine. Die Daten stammen von zwei Kohleminen in Polen, in denen jeweils das Strebbau-Verfahren zum Kohleabbau angewandt wird. Aufgearbeitet wurde der Datensatz von Marek Sikora und Lukasz Wrobel.

Seismische Stöße bzw. Wellen stellen eine große Gefahr bei Untertagarbeiten dar. Jährlich gibt es Unglücke in Minen und Stollen, die auf seismische

Bewegungen zurückzuführen sind. Diese Bewegungen können einerseits einen Steinschlag im Stollen auslösen und dadurch Arbeiter verletzen. Im schlimmsten Fall stürzen ganze Stollen ein, sodass die Arbeiter, die sich im Stollen befinden, verschüttet werden bzw. ihnen der Weg aus der Mine versperrt wird. Um diese Unglücke zu vermeiden, werden mittels eines Geophone die Bodenschwingungen an der Erdoberfläche gemessen. Durch diese Messungen soll ein Überwachungssystem aufgebaut werden, welches aufgrund einer seismischen Aktivität an der Erdoberfläche vorhersagt, ob in naher Zukunft eine Gefahr für die Arbeiter besteht.

Vor allem aufgrund der Komplexität der seismischen Prozesse und deren Folgen brachten bisherige statistische Verfahren eine wenig zufriedenstellende Erkenntnis. Ein weiteres Problem besteht darin, dass Messungen mit niedriger seismischer Aktivität eindeutig die Messungen mit hoher seismischer Aktivität (z. B. $> 10^{4J}$, mit $J = Joule$) überschreiten. Weil viele der bisher angewandten statistischen Verfahren aufgrund der Komplexität der Daten keine guten Vorhersagewerte liefern konnten, wurde in jüngster Vergangenheit vermehrt auf maschinelle Lernalgorithmen gesetzt.

Der Datensatz besteht aus 2584 Beobachtungen, die jeweils 18 Merkmale haben. Die meisten Attribute geben Aufschluss über die seismische Aktivität im vorangegangen Schichtbetrieb. Vier der 18 Merkmale sind kategoriell, die restlichen sind numerisch. Acht der numerischen Merkmale geben Auskunft über die Anzahl der aufgetretenen seismischen Bodenwellen im jeweils unterschiedlichen Energiebereich. Die Energie der Wellen wird in den Bereichen von niedriger Energie $[10^2, 10^3]$ bis hin zu hochenergetischen Wellen $[10^8, 10^{10}]$ unterteilt (siehe Abbildung 4.1 rechts unten). Zwei weitere Merkmale, die sich ebenfalls mit der seismischen Aktivität beschäftigen, geben Auskunft über die maximale Stärke der aufgetretenen Wellen und über die Gesamtenergie aller seismischen Stöße. Die kategoriellen Merkmale geben unter anderem Auskünfte, ob während der seismischen Aktivität eine Gefahr für die Untertagarbeiter in der Mine bestand (keine Gefahr, geringe Gefahr, mittlere Gefahr, hohe Gefahr) aber auch welcher Arbeit die Bergleute dort nachgingen (Kohleabbau, Vorbereitungsarbeit). Für detailliertere Informationen siehe Sikora and Wrobel (2015).

In den beiden Minen wird jeweils im Schichtbetrieb gearbeitet. Das erklärte Ziel ist nun eine Vorhersage, ob bei der aktuellen Schicht eine Bedrohung für die Arbeiter besteht. Dazu werden die durchgeführten Messungen aus den vorangegangen Schichten herangezogen. Das heißt, es gibt genau zwei Antwortmöglichkeiten $\mathbb{Y} := \{Gefahr, keine\ Gefahr\}$, wobei die Verteilung in diesem Fall wieder unausgewogen ist. Es gibt viel mehr Beobachtungen, bei denen keine Gefahr für die Arbeiter bestand, als Beobachtungen, bei denen tatsächlich ein gravierendes Ereignis eintrat.

Wie auch schon im Abschnitt 3.6.1 ist auch in diesem Fall gerade die kleinere Klasse $Y = Gefahr$ interessant. Der Minenbetreiber will wissen, wann genau eine Gefahr für die Arbeiter besteht. Ein weiteres Ziel ist natürlich auch die Minimierung der Fehlalarmrate, um nicht den Betrieb unter Tage einzuschränken und dadurch einen Verlust zu riskieren. Bei minimaler Fehlerrate

soll die Vorhersage eine gute Treffergenauigkeit für ein Unglück erzielen. Das heißt, dass im folgenden Abschnitt wieder die Methoden des "Under-Sample" bzw. des "Over-Sample" zur Vorhersage von Gefahren getestet werden.

In Abbildung 4.1 sind die numerischen Merkmalswerte in Abhängigkeit der beiden Klassen mit Hilfe von Boxplots abgebildet. Auffällig ist hierbei, dass gerade bei den Merkmalen "$M4$", $M5$", "$M9$", "$M17$" und "$M18$" die Verteilung der Werte bei den beiden Klassen unterschiedlich ist.

4.2 Simulationen

Seismische Stöße

Wie bereits im Abschnitt 4.1.1. geschildert, ist dieser Datensatz ein unbalancierter Datensatz. Von den 2584 Beobachtungen bestand in 170 Fällen eine Gefahr und in den restlichen 2414 Fällen bestand keine Gefahr für die Arbeiter in der Mine. Somit enthält der Datensatz nur etwa $\approx 6\%$ Beobachtungen, die ein gravierendes Ereignis zur Folge hatten.

Als erstes wird der Datensatz wieder in einen Trainings- und einen Testdatensatz geteilt (im Verhältnis: $2/3\,Training, 1/3\,Test$, mit $set.seed = 1984$). Jeder aufgebaute Random Forest basiert auf dem identen Trainingsdatensatz, sodass dieser auch auf der selben Testmenge getestet werden kann. Zur einfachen Darstellung werden die Ergebnisse wieder mit Hilfe einer Kontingenztabelle angeführt.

Beim ersten Versuch wird ein Random Forest auf dem kompletten Trainingsdatensatz trainiert. Für den Testdatensatz erhält man folgendes Ergebnis:

1.Versuch		Wahre Klasse	
	$Klassen$	"$keine\ Gefahr$"	"$Gefahr$"
Ergebnis	"$keine\ Gefahr$"	803	57
	"$Gefahr$"	2	0

Aus dieser Tabelle ist klar ersichtlich, dass dieser Random Forest zwei Warnungen einer bestehenden Gefahr für die Arbeiter ausgab. Diese zwei Warnungen waren allerdings allesamt Fehlalarme. Die restlichen 57 Beobachtungen, bei denen eine Gefahr für die Arbeiter bestand, erkannte dieser Random Forest nicht. Dies ist klarerweise aber nicht im Interesse des Minenbetreibers, der eine sichere Arbeit für die Bergarbeiter gewährleisten will. In Abbildung 4.2 (links) ist auch sehr gut erkennbar, dass, wenn man keine Restriktionen am Trainingsdatensatz vornimmt, der Random Forest Beobachtungen der beiden Klassen nicht gut trennen kann.

Anschließend wird wieder die "naive" Methode getestet, um gleich viele Beobachtungen pro Klasse für die Trainingsmenge zu erhalten. Es werden aus dem Trainingsdatensatz zufällig 113 Beobachtungen aus jeder Klasse ohne Zurücklegen gezogen ($set.seed = 1984$) und auf dieser Menge wird dann ein

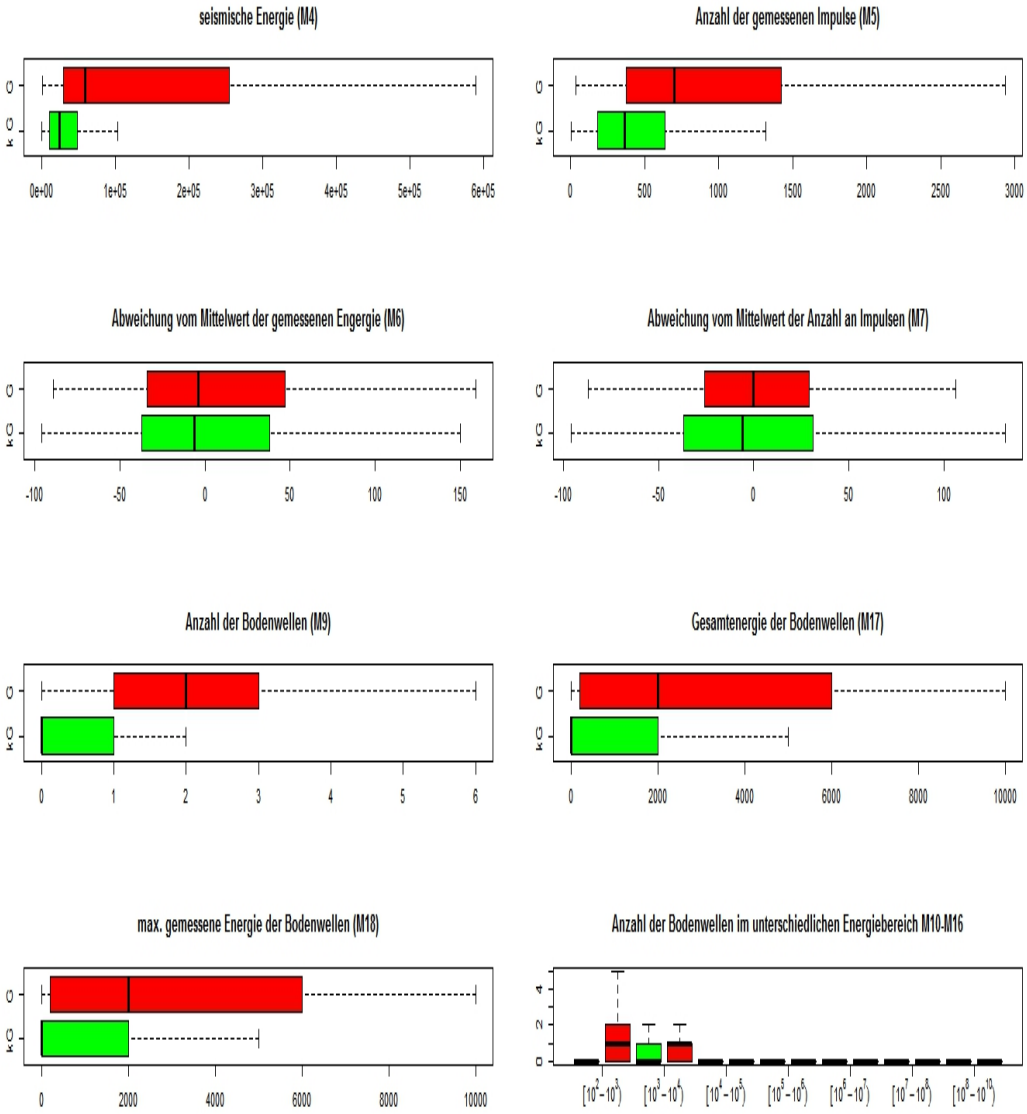

Abbildung 4.1: Boxplot aller numerischen Merkmale, getrennt in Klassen und ohne Ausreißer

Random Forest aufgebaut. Man erhofft sich dadurch, dass der Random Forest den Beobachtungen der Klasse $Y =$ "*keine Gefahr*" weniger Aufmerksamkeit schenkt, um damit tatsächliche Gefahren besser vorhersagen zu können. Das Ergebnis ist wiederum in folgender Tabelle ersichtlich:

Naive Methode		Wahre Klasse	
	Klassen	"*keine Gefahr*"	"*Gefahr*"
Ergebnis	"*keine Gefahr*"	557	21
	"*Gefahr*"	248	36

Dieser Random Forest erkannte 36-mal, dass eine Gefahr für die Arbeiter im Stollen bestand. In 64% der Fälle hätte dieser somit ein Unglück vorhersagen können. Ingesamt 21-mal hätte dieser Zufallswald ein Unglück nicht vorhergesagt. Bei 248 Fehlalarmen ($\approx 87\%$) wäre zudem die Arbeit auch oft unterbrochen worden, was eventuell einen Verlust für den Betreiber nach sich gezogen hätte.

Beim dritten Versuch wird die Methode des "Under Sample" getestet (vgl. Abschnitt 3.6.1). Es wird für jeden Entscheidungsbaum ein unterschiedlicher Trainingsdatensatz konstruiert, der jeweils gleich viele Beobachtungen für jede Klasse hat. Erzeugt wird dieser, indem man aus dem kompletten Trainingsdatensatz 113 Beobachtungen von jeder Klasse **mit** Zurücklegen zieht.

Under Sample (mZ)		Wahre Klasse	
	Klassen	"*keine Gefahr*"	"*Gefahr*"
Ergebnis	"*keine Gefahr*"	646	27
	"*Gefahr*"	159	30

Bei einer etwas geringeren Fehlalarmrate ($\approx 84\%$) konnte man jedoch mit der "Under Sample" Methode die Treffergenauigkeit bei der Vorhersage von Gefahren ($\approx 53\%$) nicht verbessern.

Nun wird erneut die Methode des "Under Sample" angewandt, allerdings wird **ohne** Zurücklegen gezogen. Das heißt wiederum, dass jeder Entscheidungsbaum im Wald die gesamte Information der Beobachtungen von der Klasse "Gefahr" zur Verfügung hat. Es sei an dieser Stelle noch vermerkt, dass man mit dieser Methode den Bagging-Algorithmus umgeht (vgl. Abschnitt 3.1.1 und Appendix). Daraus resultiert, dass der Missklassifikationsfehler höher sein wird (vgl. Gleichung 3.3 und Gleichung 3.6). Man erhält folgendes Ergebnis:

Under Sample (oZ)		Wahre Klasse	
	Klassen	"*keine Gefahr*"	"*Gefahr*"
Ergebnis	"*keine Gefahr*"	577	22
	"*Gefahr*"	228	35

In 61% der Fälle konnte man ein Unglück vorhersagen. Allerdings ist die Fehlalarmrate ($\approx 87\%$) etwas höher als im vorhergehenden Modell "Under Sample (mZ)".

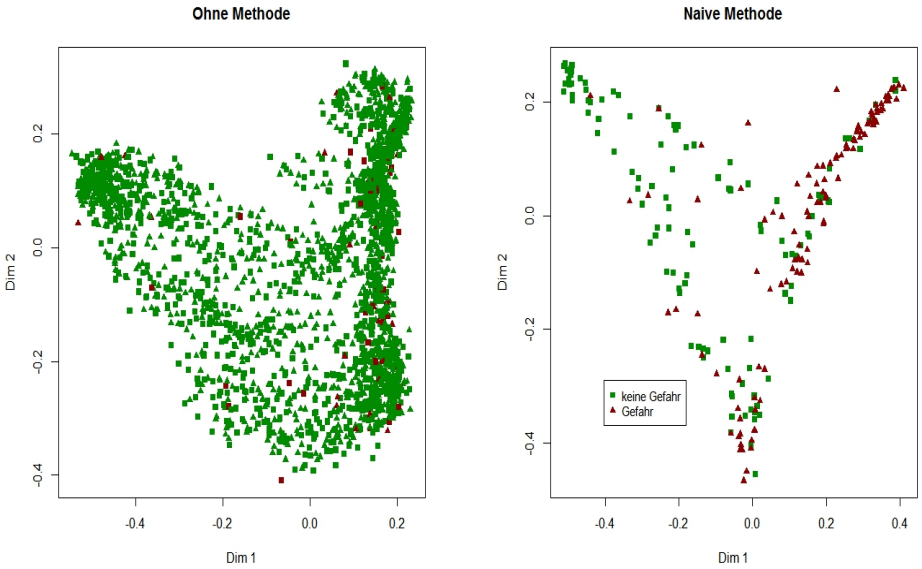

Abbildung 4.2: MDS-Plot der Proximity-Werte

Zu guter Letzt wird noch die "Over Sample" Methode angewandt. Als Ergebnis erhält man:

Over Sample		Wahre Klasse	
	Klassen	"*keine Gefahr*"	"*Gefahr*"
Ergebnis	"*keine Gefahr*"	792	49
	"*Gefahr*"	13	8

Wie auch im Abschnitt 3.6.1 erhält man bei dieser Methode ein wenig zufriedenstellendes Ergebnis. Die Fehlalarmrate ($\approx 62\%$) ist zwar eindeutig geringer als zuvor, allerdings konnte man mit dieser Methode nur in $\approx 14\%$ der Fälle ein Unglück verhindern.

In Abbildung 4.2 sind die Auswirkungen bei Anwendung der naiven Methode dargestellt. Es sind die Proximity-Werte (vgl. Abschnitt 3.5) mittels Multidimensional Scaling Plot eines Random Forest, der auf dem ganzen Datensatz (ohne jegliche Restriktionen) trainiert wurde und andererseits eines unter Anwendung der naiven Methode trainierten abgebildet. Man sieht, dass unter Anwendung der naiven Methode die beiden Klassen besser getrennt sind. Auf der rechten Abbildung sind zwei Cluster erkennbar, allerdings gibt es auch einen Bereich, an dem sich die Beobachtungen teilweise überlappen. Ohne Restriktionen am Trainingsdatensatz (linke Abbildung) überlappen sich die Beobachtungen beider Klassen und es ist auch kein Cluster erkennbar.

Die besten Vorhersagewerte für die Klasse "*Gefahr*" erhält man wieder mit der "Under Sample (mZ)" Methode und der "naiven" Methode. In der Folge werden bei 500 Durchläufen nochmals die Methoden des "Under Sample (oZ)" und "Under Sample (mZ)" bzw. auch die "*naive*" Methode mit jeweils dem identen Testdatensatz getestet. In Abbildung 4.3 sind die Ergebnisse mittels

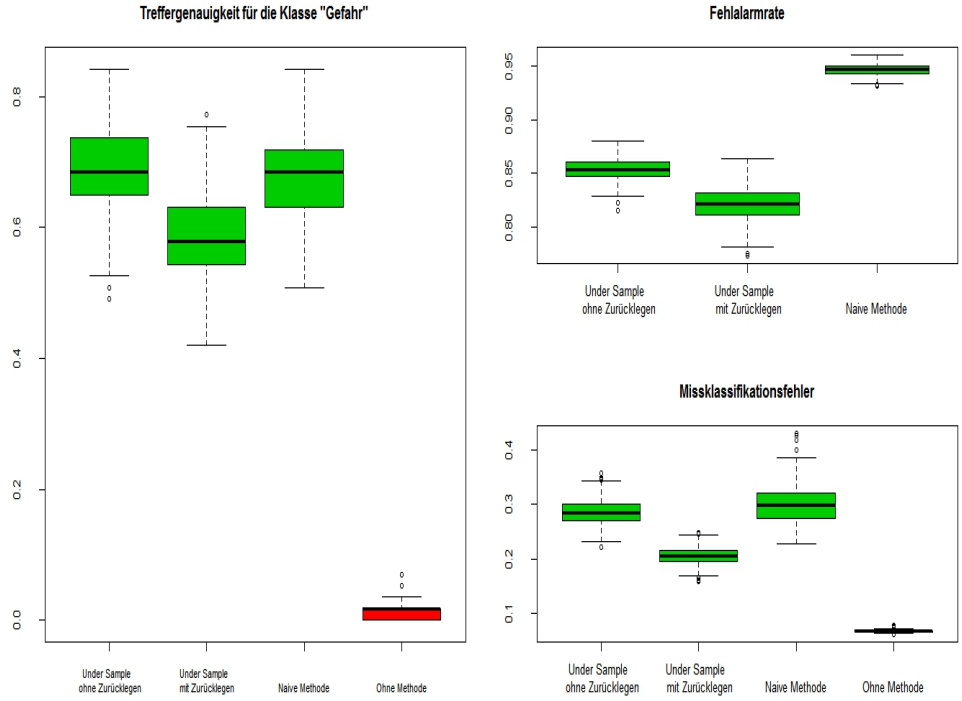

Abbildung 4.3: Boxplot für die Ergebnisse des Testdatensatzes mit jeweils unterschiedlichen Methoden

Boxplots abgebildet. Wie man sieht, bewirkt die Methode des "Under Sample (oZ)"die beste Vorhersage für Gefahren. Allerdings ist sowohl die Fehlalarmrate als auch der Missklassifikationsfehler höher als bei der Methode des "Under Sample (mZ)". Die höhere Missklassifikationsrate der "Under Sample (oZ)" Methode ist dahingehend begründet, dass jeder Entscheidungsbaum die komplette Information der Klasse "$Gefahr$" zur Verfügung hat. Damit wird aber auch die Korrelation unter den Bäumen größer, weil jeder Trainingsdatensatz aus zumindest 50% gleicher Beobachtungen besteht und dies hat eine höhere Missklassifikationsrate zur Folge. (vgl. Gleichung 3.3 und Gleichung 3.6)

Die "naive" Methode hat sowohl die höchste Fehlalarm- als auch die höchste Missklassifikationsrate. Allerdings zeigen die Boxplots, dass diese Methode eine ähnlich gute Treffergenauigkeit aufweist, wie die Methode des "Under Sample (oZ)". Wenn man keine der Methoden auf dem Datensatz anwendet, erhält man zwar eine sehr geringe Missklassifikationsrate aber man nimmt damit auch eine sehr geringe Treffergenauigkeit der Klasse "$Gefahr$" in Kauf und damit auch eine hohe Fehlalarmrate.

Nun werden wieder die Methoden des "Under Sample (mZ)" bzw. "Under Sample (oZ)" sowie die "naive" Methode angewandt. Allerdings werden nun die Ergebnisse mit einem von Conditional Inference Trees (CIT) und einem vom CART-Algorithmus aufgebauten Random Forest bei jeweils 500 Durchläufen verglichen. In Abbildung 4.4 sind die Ergebnisse mittels Boxplots wiedergegeben. Anhand dieser Boxplots ist erkennbar, dass ein mit CIT auf-

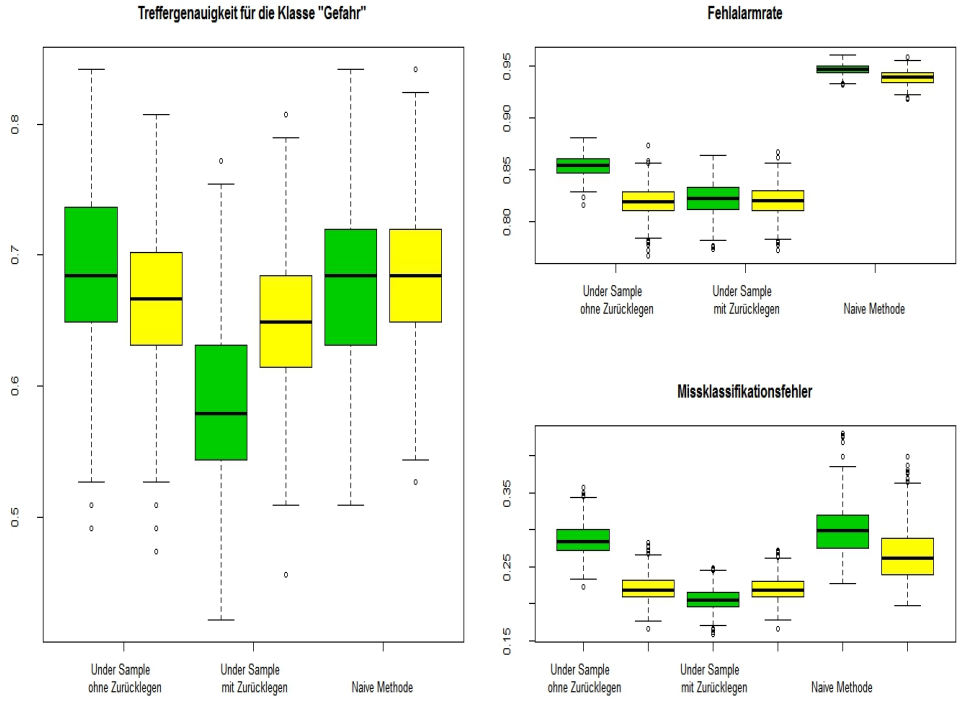

Abbildung 4.4: CART vs. Conditional Inference Trees: In grün sind die Ergebnisse des CART-Algorithmus und in gelb die der Conditional Inference Trees abgebildet

gebauter Random Forest teilweise ein besseres Ergebnis liefert, als ein mit dem CART-Algorithmus aufgebauter.

Wenn man die beiden wichtigen Faktoren (Fehlalarmrate, Treffergenauigkeit) für den Minenbetreiber betrachtet, sind die Ergebnisse größtenteils besser, wenn man einen Zufallswald mit Conditional Inference Trees verwendet. Bei der Methode des "*Under Sample* (*oZ*)" sind im Schnitt die Ergebnisse des mit CART-Algorithmus aufgebauten Random Forest etwas besser, sie weisen einen Mittelwert von 0.68 auf. Hingegen hat die "*Under Sample* (*oZ*)" Methode bei einem mit CIT aufgebauten Zufallswald einen Mittelwert von 0.66. Allerdings ist die Varianz der Ergebnisse des mit CART-Algorithmus aufgebauten Random Forest auch etwas höher (Varianz: CART=0.004, CIT=0.003), was eine höhere Streuung der Ergebnisse nach sich zieht.

Als nächstes werden die Merkmalswichtigkeiten von den zwei unterschiedlich aufgebauten Random Forest Funktionen (CART vs. Conditional Inference Trees) betrachtet. Wenn man Abbildung 4.1 betrachtet, liegt die Intuition nahe, dass vor allem die Merkmale M4, M5, M9, M17 und M18 wichtige Faktoren für die Erkennung von Gefahren sein werden. Der Grund dafür ist der, dass gerade an diesen Merkmalen die Beobachtungen der beiden Klassen unterschiedliche Werte aufweisen. Hierfür werden erneut für jeden Algorithmus 500 Zufallswälder (jeweils mit ihren Default Werten) aufgebaut. Zuerst werden bei diesen Durchläufen noch keine Einschränkungen am Datensatz

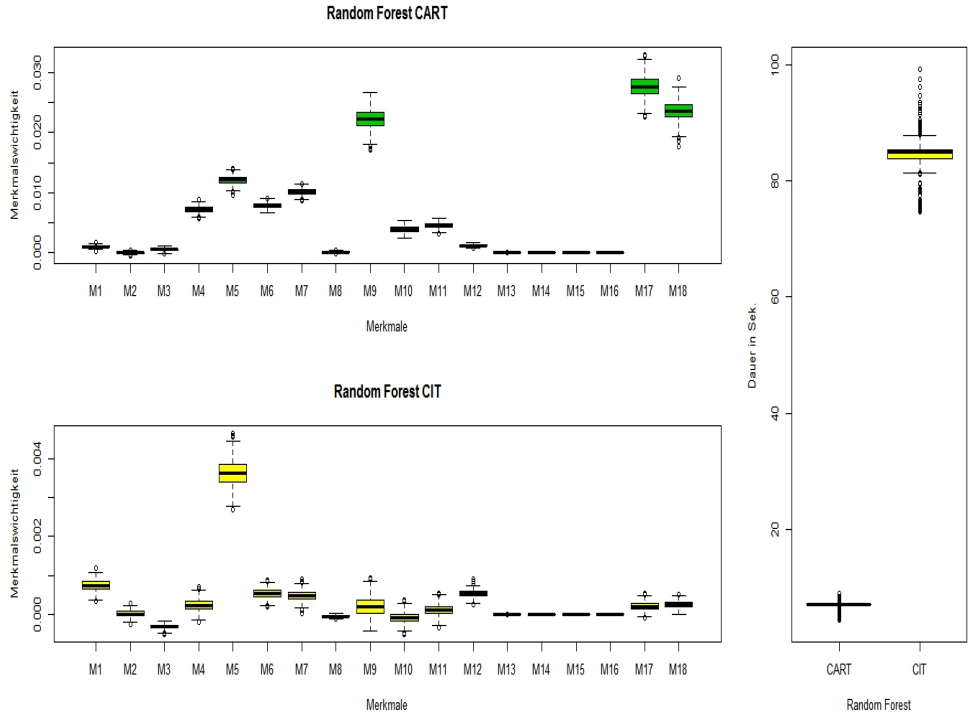

Abbildung 4.5: Auf der linken Seite sind die Merkmalswichtigkeiten von 500 Durchläufen (CART und CIT) abgebildet und auf der rechten Seite die dazu benötigte Zeit

vorgenommen. Das heißt, es wird noch keine der zuvor beschriebenen Methoden ("Under Sample oZ" , "Naive", etc.) angewandt, sodass jeder Random Forest den kompletten Datensatz für den Lerndatensatz zur Verfügung hat.

In Abbildung 4.5 sind die von den beiden Algorithmen berechneten Merkmalswichtigkeiten dargestellt, ergänzt noch um jeweils die dafür benötigte Berechnungszeit. Man erkennt, dass beide Algorithmen unterschiedliche Merkmale als wichtig kennzeichnen. Beim CART-Algorithmus sind vor allem für die Entscheidungsfindung folgende drei Merkmale wichtig:

- "$M18$"– max. Energie der Bodenwellen im vorangegangenen Schichtbetrieb
- "$M17$"– Gesamtenergie der Bodenwellen im vorangegangenen Schichtbetrieb
- "$M9$"– Anzahl der gemessenen Bodenwellen im vorangegangenen Schichtbetrieb

Bei einem Random Forest mit Conditional Inference Trees wird nur das Merkmal "$M5$" als wichtig gekennzeichnet.

- "$M5$"– Anzahl der vom Genophon gemessenen Impulse im vorangegangenen Schichtbetrieb

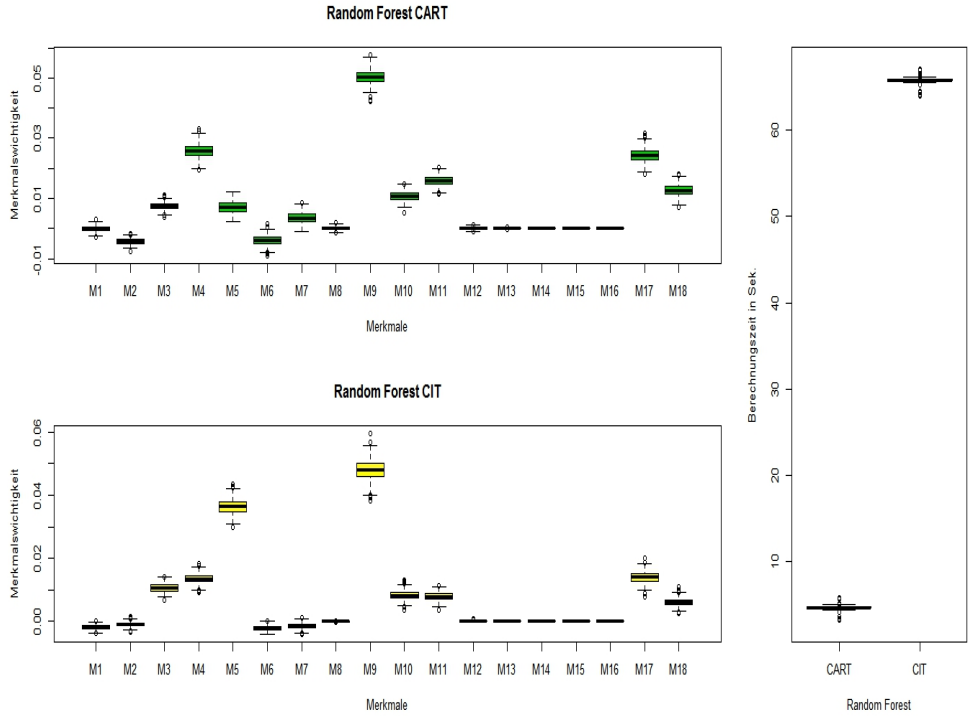

Abbildung 4.6: Merkmalswichtigkeiten unter Anwendung der "*naiven*" Methode bei jeweils 500 Durchläufen

Zu bemerken ist nochmals, dass bei diesen Durchläufen die Zufallswälder den kompletten Datensatz zum Trainieren zur Verfügung hatten. Um den Missklassifikationsfehler zu minimieren, versuchen diese Zufallswälder, in erster Linie die Klasse "$Keine Gefahr$" vorherzusagen und weisen somit auch nur eine minimale Treffergenauigkeit für die Klasse "Gefahr" auf (vgl. Abbildung 4.3).

In Abbildung 4.6 sind die Merkmalswichtigkeiten von Zufallswäldern abgebildet, wobei der Datensatz unter der "*naiven*" Methode geändert wurde. Der unter dem CART-Algorithmus erzeugte Zufallswald erkennt nun neben den Merkmalen "$M9$" und "$M17$" auch noch das Merkmal "$M4$" als wichtig an.

- "$M4$"$-$ seismische Gesamtenergie im vorangegangenen Schichtbetrieb

Hingegen ist für einen Zufallswald mit Conditional Inference Trees neben "$M5$" nun auch das Merkmal "$M9$" für die Vorhersage eines Unglücks wichtig. Auffallend ist erneut, dass auch diesmal der CIT-Algorithmus eindeutig mehr Rechenzeit benötigt, auch wenn der Trainingsdatensatz nun viel weniger Beobachtungen zur Verfügung hat.

Diese Ergebnisse untermauern wiederum die zuvor gemachte Intuition, dass vor allem die Merkmale "M4", "M5", "M9", "M17" und "M18" wichtig für die Entscheidungsfindung sein werden.

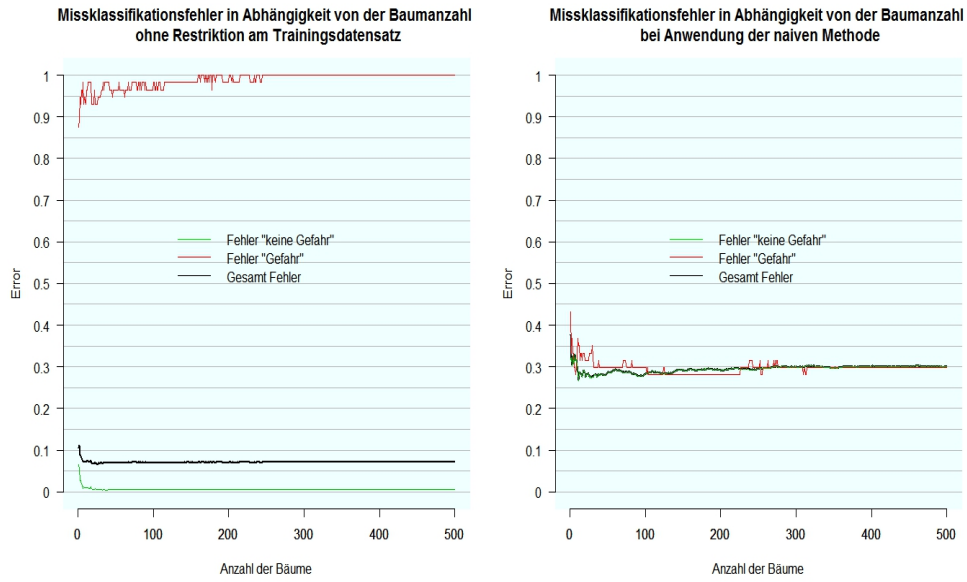

Abbildung 4.7: Der Missklassifikationsfehler in Abhängigkeit von der Anzahl der Bäume

4.3 Analyse

In diesem Abschnitt wird nun weiterhin versucht, den Fehler 1. Art (Fehlalarmrate) und den Fehler 2. Art (Missklassifikationsfehler für Klasse "Gefahr") zu minimieren, indem man unter anderem Einfluss auf den "Zufall" eines Random Forest nimmt. In den vorhergehenden Modellen wurden die Zufallswälder jedesmal mit den Default-Werten aufgebaut. Die im R-Package "randomForest" implementierte Funktion besitzt einige Parameter, mit denen man das Wachstum der Bäume aber auch den Zufall steuern bzw. beeinflussen kann. Mit dem Ziel, den Missklassikationsfehler zu verringern sowie die Treffergenauigkeit für ein Unglück zu erhöhen, werden in diesem Abschnitt mehrere Versuche mit jeweils unterschiedlichen Parameterwerten durchgeführt.

Zuerst wird aber noch gezeigt, wie sich der Missklassifikationsfehler in Abhängigkeit von der Anzahl der Bäume ändert. Es wird unter Anwendung der naiven Methode zuerst nur ein Baum erzeugt und dieser auf einem Testdatensatz getestet. Es wird dann schrittweise ein Baum hinzugefügt und dieser Wald auf dem selben Testdatensatz getestet. In Abbildung 4.7 sind die Ergebnisse dargestellt. Man erkennt einerseits, dass, je mehr Bäume ein Wald besitzt, desto stabiler verhält sich der Missklassifikationsfehler. Und wenn man die beiden Abbildungen miteinander vergleicht, erkennt man gut, dass die Vorhersage von der Klasse "Gefahr" aber auch der Missklassifikationsfehler bei Anwendung der naiven Methode erhöht wird.

Im Abschnitt 2.2.3 wurde die Methode des Cost-Complexity Pruning vorgestellt. Einzelne Entscheidungsbäume werden oft an die Trainingsdaten überangepasst. Deswegen ist es sinnvoll, wenn man diese stutzt. In den nächsten Versuchen wird untersucht, wie sich Änderungen der Baumgröße auf das

Ergebnis auswirken. Leider gibt es in der im "RandomForest"-Package implementierten Funktion "randomForest" keine Möglichkeit, die Methode des Cost-Complexity Pruning anzuwenden. Mit dem Parameter "maxnode" besteht allerdings die Möglichkeit, die Bäume nur so lange wachsen zu lassen, bis sie eine festgelegte Knotenanzahl erreicht haben. Dies ist allerdings eine etwas "rustikale" Vorgehensweise, da man unabhängig von der Reinheit der Knotenpunkte entscheidet, ob das Wachstum des Baumes gestoppt wird. Darum kann es auch vorkommen, dass wichtige Zusammenhänge des Datensatzes verloren gehen.

Ziel ist es weiterhin, Gefahren in den Minen vorherzusagen. Deswegen wird in den folgenden Versuchen die "naive" Methode auf dem Trainingsdatensatz angewandt. Es werden 500 Zufallswälder, bestehend aus Bäumen mit jeweils identer Knotenanzahl, konstruiert. Anschließend wird die Knotenanzahl um jeweils einen Knoten erhöht und erneut 500 Zufallswälder erzeugt. Wenn man die naive Methode auf die Random Forest Funktion ohne Restriktionen (d. h. mit Default Werten) anwendet, bestehen die Bäume im Schnitt aus 46 Knoten (inkl. Endknoten).

In Abbildung 4.8 ist das arithmetische Mittel des Missklassifikationsfehlers der Zufallswälder dargestellt, die jeweils aus Bäumen mit gleicher Knotenanzahl bestehen. Ab dem Zeitpunkt, an dem jeder Baum 19 Knotenpunkte enthält, steigt der Missklassifikationsfehler und dieser bleibt dann ab einer Knotenanzahl von 46 durchgehend stabil. Interessanterweise ist dies ist auch das arithmetische Mittel der Baumgröße von 500 Random Forests mit Default Werten. Wie man erkennen kann, neigt der Random Forest durch eine höhere Knotenanzahl zu einer Überanpassung an den Trainingsdaten. Die rote Kurve erreicht an den Punkten 3, 5 und 10 ein Minimum und somit erhält man an diesen Punkten den geringsten Missklassifikationsfehler für die Klasse "$Gefahr$".

Mit dem Parameter "nodesize" hat man eine weitere Möglichkeit, die Bäume zu kürzen. Man gibt mit diesem Parameter eine obere Schranke S für die Anzahl der Beobachtungen im Knotenpunkt an. Das heißt, dass beim Lernprozess des Baumes ein Knotenpunkt nicht mehr weiter unterteilt wird, genau dann, wenn in diesem weniger als S Beobachtungen vorkommen. Unter der Anwendung der "$naiven$" Methode besteht der Trainingsdatensatz aus 226 Beobachtungen. Es werden wieder 500 Zufallswälder mit dem identen "nodesize" Wert erzeugt. Beginnend mit dem größtmöglichen Baum mit nur einer Beobachtung im Endknoten, wird die Anzahl jedes Mal um eine Beobachtung erhöht, bis man schlussendlich bei maximal 200 Beobachtungen pro Endknoten angelangt ist.

In der Abbildung 4.9 ist das arithmetische Mittel des Missklassifikationsfehlers von den 500 Zufallwäldern mit jeweils dem identen Schrankenwert dargestellt. Die rote Kurve erreicht hier bei den Werten 16 und 24 ein Minimum, deswegen ist an diesen Stellen auch der Missklassifikationsfehler für die Klasse "$Gefahr$" am geringsten. Je mehr Beobachtungen in den Endknoten vorkommen, desto geringer wird der Missklassifikationsfehler des Zufallswaldes. Allerdings ist zu beachten, dass dabei der Klassifizierungsfehler der Klasse

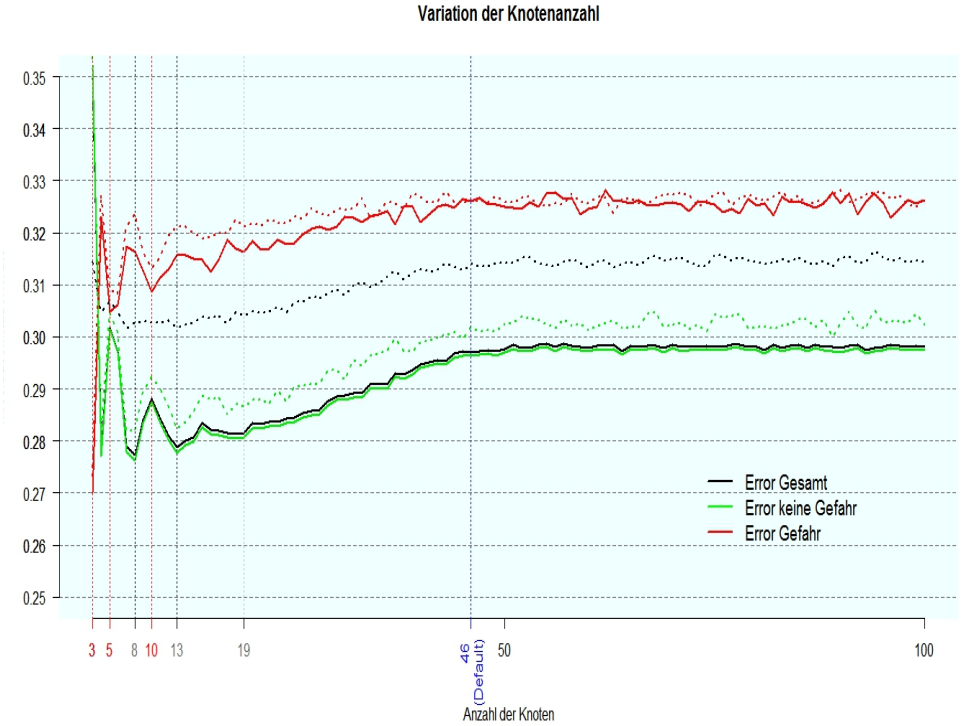

Abbildung 4.8: Variation bei der Knotenanzahl der Bäume. Punktiert dargestellt sind in dieser Abbildung die mittels der "Out of Bag" Daten geschätzten Missklassifikationsfehler

"$Gefahr$" im gleichen Maße zunimmt, wie er bei der Klasse "$keine\,Gefahr$" abnimmt. Auch in diesem Fall macht sich eine Überanpassung an die Trainingsdaten bemerkbar. Wenn man die Bäume bis zur maximalen Größe (nodesize=1) wachsen lässt, ist der Missklassifikationsfehler (schwarze Kurve) auch am größten. Je mehr Beobachtungen in den Endknoten vorkommen dürfen, desto geringer wird dann dieser Fehler.

In Abbildung 4.10 werden die Ergebnisse von 500 Random Forests einerseits mit den Default Werten und andererseits mit maxnode=3, maxnode=5, maxnode=10 und maxnode=19 bzw. mit nodesize=16 und nodesize=24 gegenübergestellt. Die Treffergenauigkeit ist bei allen Parameterwerten besser als bei den Default Werten. Wenn man die Boxplots genauer betrachtet, erhält man das beste Ergebnis, wenn man das Wachstum des Baumes stoppt, sobald 16 Beobachtungen (nodesize=16) in den Endknoten vorkommen. Mit einer maximalen Knotenanzahl von drei (maxnode=3) erhält man einerseits die beste Treffergenauigkeit für die Klasse "Gefahr", allerdings sind der Missklassifikationsfehler sowie die Fehlalarmrate auch größer.

In folgender Tabelle ist das arithmetische Mittel, die empirische Standardabweichung (SD) und auch der Median von den Ergebnissen der "Treffergenauigkeit" wiedergegeben.

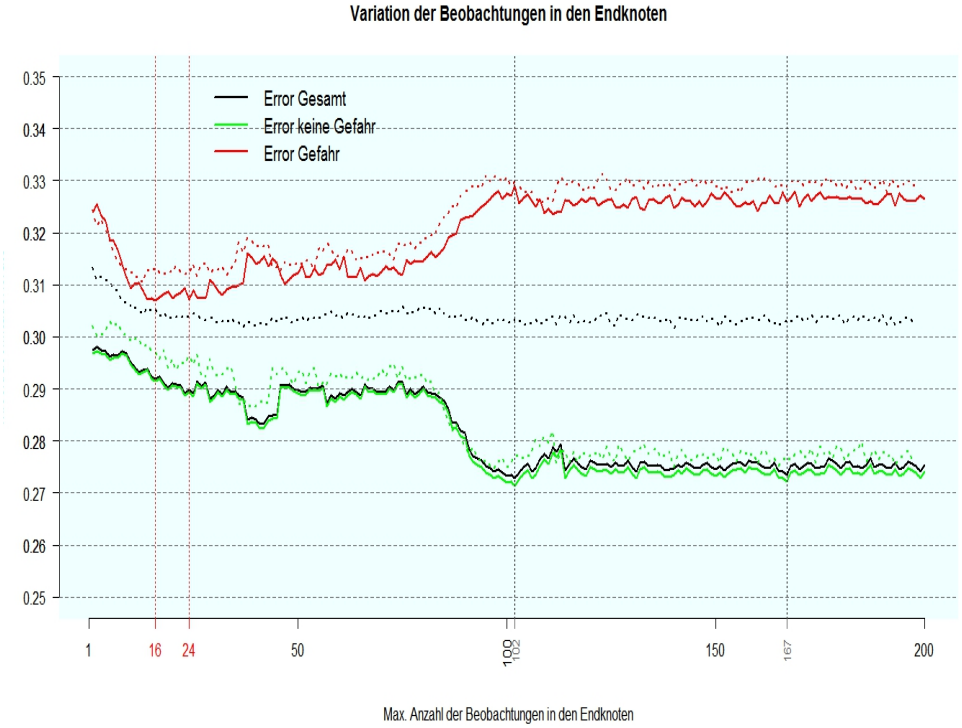

Abbildung 4.9: Variation bei der Anzahl an Beobachtungen in den Endknoten. Punktiert dargestellt sind in dieser Abbildung die mittels der "Out of Bag" Daten geschätzten Missklassifikationsfehler

Treffergenauigkeit	maxnode 3	maxnode 5	maxnode16	maxnode19
Median	0.736	0.701	0.684	0.666
Mean	0.729	0.697	0.685	0.676
SD	0.062	0.060	0.066	0.068
	nodesize 16	nodesize 24	Default	
Median	0.701	0.684	0.684	
Mean	0.693	0.687	0.677	
SD	0.066	0.067	0.063	

Als nächstes wird versucht, die Ergebnisse der "*Under Sample* (*oZ*)" Methode zu verbessern. Die Idee dahinter ist die, dass der Missklassifikationsfehler verringert wird, wenn sich die Trainingsdatensätze der Bäume stärker unterscheiden. Bei dieser Methode hatte im Abschnitt 4.2 jeder Baum als Trainingsmenge alle Beobachtungen der Klasse "$Gefahr$" zur Verfügung und dadurch waren auch mindestens 50% der Beobachtungen im Trainingsdatensatz ident.

In den folgenden Versuchen werden für die Trainingsmenge nicht mehr 113 Beobachtungen jeder Klasse ohne Zurücklegen gezogen, sondern nur noch ein prozentueller Anteil davon. Damit variieren in den Trainingsmengen der Bäume aber auch die Beobachtungen der Klasse "$Gefahr$", was folglich auch die empirische Korrelation unter den Bäumen verringern sollte.

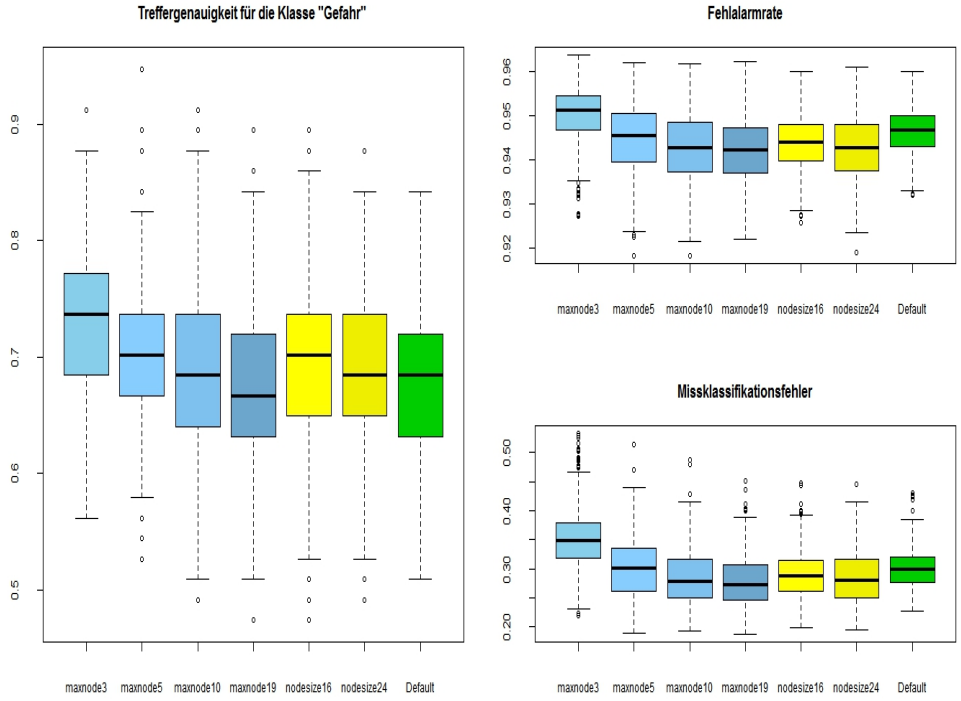

Abbildung 4.10: Ergebnisse von 500 Durchläufen bei unterschiedlichen Parameterwerten für maxnode und nodesize

Es werden im Folgenden 500 Zufallwälder aufgebaut, bei denen die Trainingsmenge zuerst aus 28 ($\approx 25\%$) Beobachtungen pro Klasse besteht, dann aus 57 ($\approx 50\%$) und zum Schluss aus 85 ($\approx 75\%$) Beobachtungen. In Abbildung 4.11 sind wiederum die Ergebnisse mittels Boxplots abgebildet.

Die erhoffte Verringerung des Missklassifikationsfehlers ist eingetreten. Wenn man vor allem die Ergebnisse bei 50% betrachtet, erkennt man, dass mit dem geringeren Missklassifikationsfehler die Fehlalarmrate verringert wurde und auch die Treffergenauigkeit für die Klasse "$Gefahr$" erhöht wurde. Das arithmetische Mittel für die Treffergenauigkeit liegt bei 69%. In folgender Tabelle ist wiederum der Median, die empirische Standardabweichung (SD) und das arithmetische Mittel der Treffergenauigkeit für die Klasse "Gefahr" angegeben.

Treffer "$Gefahr$"	25%	50%	75%	100%
Median	0.701	0.701	0.701	0.684
arithmetisches Mittel	0.689	0.692	0.695	0.688
SD	0.06	0.068	0.059	0.063

In unten angeführter Tabelle werden der Median, die empirische Standardabweichung (SD) und das arithmetische Mittel des Missklassifikationsfehlers angegeben:

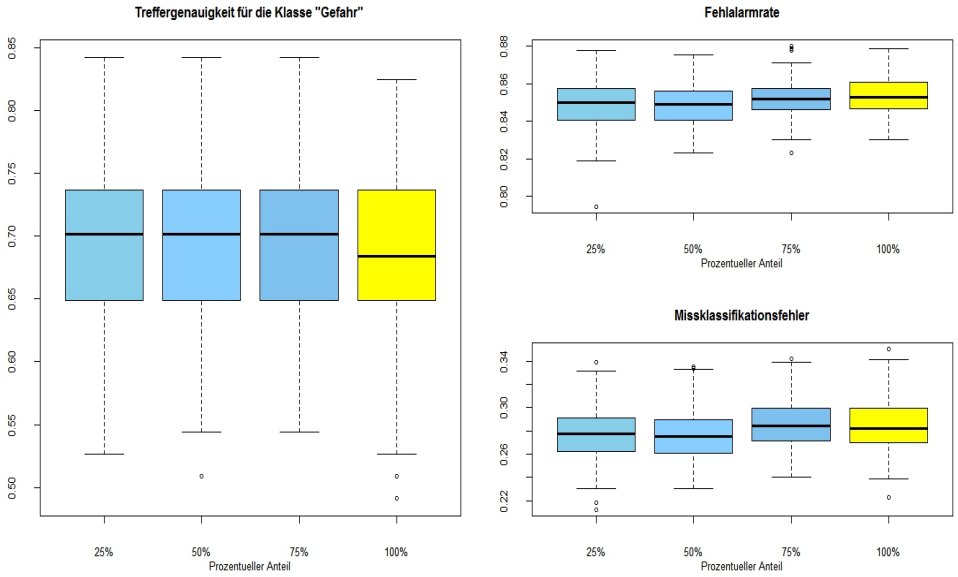

Abbildung 4.11: Ergebnisse der "Under Sample (oZ)" Methode mit unterschiedlicher Größe der Trainingsmenge

Missklassifikationsfehler	25%	50%	75%	100%
Median	0.277	0.274	0.284	0.281
arithmetisches Mittel	0.277	0.277	0.285	0.284
SD	0.02	0.02	0.02	0.02

Wie in der Abbildung 4.11 bereits ersichtlich wurde, ist anhand dieser Tabellen ebenfalls zu erkennen, dass man etwas bessere Ergebnisse erzielt, wenn man als Trainingsmenge für die Bäume nicht die ganze Information nützt. Zwar hat sich die Treffergenauigkeit nur minimal verbessert, allerdings wird der Missklassifikationsfehler reduziert, was auch zu einer Verringerung des Fehlers 1.Art führt.

Um die Korrelation unter den Bäumen noch weiter zu verringern, besteht eine weitere Möglichkeit darin, dass man die Anzahl der Merkmale variiert, die ein Entscheidungsbaum für die Entscheidungsfindung zur Verfügung hat. Je weniger Merkmale die Bäume für ihre Entscheidungsfindungen zur Verfügung haben, desto zufälliger werden diese aufgebaut sein und somit werden sich die Bäume auch weniger gleichen. (vgl. Abschnitt 3.1.3)

Mit dem Parameter mtry bei der Random Forest Funktion kann man eine Anzahl festlegen, wie viele Merkmale pro Entscheidung zufällig gezogen werden. Der Default-Wert bei einer Klassifikation ist für diese Funktion $\sqrt{n}$, mit $n := Anzahl\,der\,Merkmale$. Wenn man als Wert 1 wählt heißt das, dass für jeden Entscheidungspunkt zufällig nur ein Merkmal gezogen wird und auf diesem der beste Verzweigungspunkt gesucht wird. Im Folgenden wird wieder die Methode "$Under\,Sample\,(oZ)$" (mit 57 Beobachtungen pro Klasse) angewandt, allerdings mit jeweils unterschiedlichen mtry Werten. Die Bäume haben zuerst nur ein Merkmal pro Entscheidungsfindung zur Verfügung und

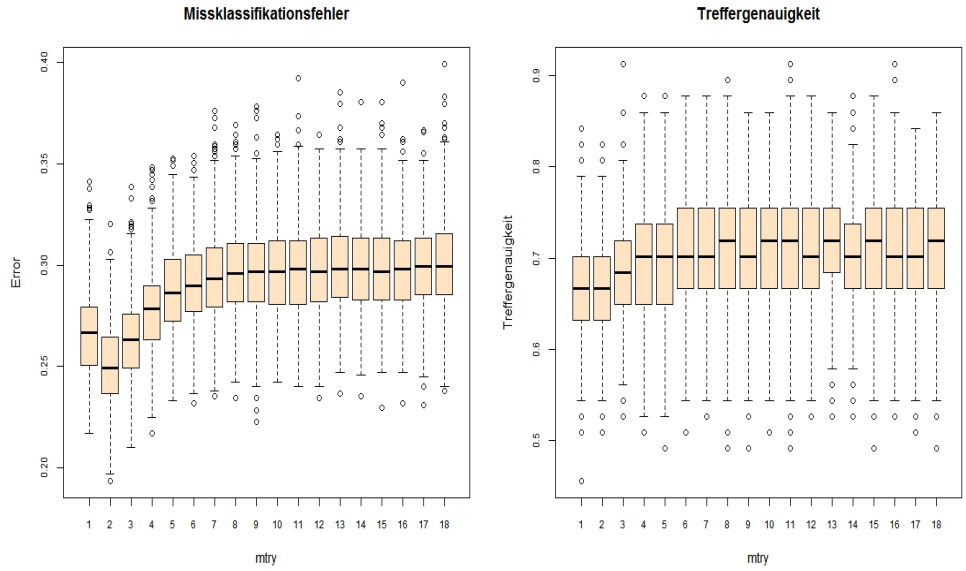

Abbildung 4.12: Boxplot mit jeweils unterschiedlichen mtry Werten

in den nächsten Iterationen wird der Parameterwert jeweils um ein Merkmal erhöht.

Aus der Abbildung 4.12 sind die Ergebnisse von 500 Durchläufen ersichtlich. Es ist festzustellen, dass, je kleiner der Wert für mtry ist, desto geringer auch der Missklassifikationsfehler ist. Das heißt, dass, je "zufälliger" die Bäume aufgebaut sind und je geringer auch die Korrelation unter den Bäumen ist, desto geringer ist auch der Missklassifikationsfehler (vgl. Gleichung 3.6). Allerdings nimmt man dafür auch eine etwas geringere Treffergenauigkeit in Kauf. Je höher der Wert für "mtry" ist, desto öfter könnte ein Unglück vorhergesehen werden.

Im letzten Abschnitt ist bei den Ergebnissen gut ersichtlich, welch wichtigen Faktor der "Zufall" in einem Random Forest darstellt. Je zufälliger der Wald konstruiert wird, desto geringer ist auch der Missklassifikationsfehler. Allerdings konnte man damit gerade bei unbalancierten Datensätzen die Treffergenauigkeit nur minimal für Beobachtungen der kleineren Klasse erhöhen. Jedoch wird durch die Verringerung des Missklassifikationsfehlers gerade der Fehler der 1. Art reduziert.

Des Weiteren war festzustellen, dass man mit den Methoden "Under Sample (oZ)" bzw. mit der naiven Methode die besten Werte für die Treffergenauigkeit erhält. In etwa 70% der Fälle konnte man ein Unglück vorhersagen. Wenn man noch dazu die Knotenanzahl unter der naiven Methode beeinflusst, erhält man sogar das beste Ergebnis (73%) für die Treffergenauigkeit. Allerdings ist eine Reduzierung bzw. eine Festlegung der Knotenanzahl nur mit Vorsicht zu genießen. Es können dadurch einige wichtige Zusammenhänge der Daten verloren gehen. Gerade auch, weil ein Random Forest eine Art "Black Box" ist, ist es trotz teilweise guter Ergebnisse schwer ersichtlich, wie das Ergebnis zustande kam und welche weiteren Auswirkungen eine Reduzierung

der Knotenanzahl haben wird.

Auch wenn man mit einer Erhöhung der Baumanzahl zu keiner Überanpassung der Trainingsdaten kommt, kann ein Random Forest gerade durch eine hohe Knotenanzahl der Bäume trotzdem an die Trainingsdaten überangepasst sein. Wie im vorhergehenden Beispiel ersichtlich war, war der Random Forest überangepasst, weil man alle Bäume bis zur maximalen Größe wachsen hat lassen. Bei einer Einschränkung der Knotenanzahl waren die Ergebnisse jedenfalls besser.

Appendix

R-Code: Under Sample und Over Sample

My Random Forest

```
FORMULA <- y ~ .
MyRandomForest <- function(FORMULA, data, numberOfTrees = 500, SizeTrain=2/3,
                  method="DownSample", klassengr=c(1,1), REPLACE=TRUE){

#Erzeugung eines Trainings- und Testdatensatzes

No <- data[data$y == "no",]; Yes <- data[data$y == "yes",]

NoTraining <- sample(1:dim(No)[1], as.integer(SizeTrain*dim(No)[1]))
YesTraining <- sample(1:dim(Yes)[1], as.integer(SizeTrain*dim(Yes)[1]))
testdata=rbind(No[-NoTraining,],Yes[-YesTraining,])
TrainNo<-No[NoTraining,]
TrainYes<-Yes[YesTraining,]

#Downsample

forests <- list()
NoSize <- dim(TrainNo)[1]
YesSize <- dim(TrainYes)[1]

if(method=="DownSample"){
   print("DownSample")
   samp_y<-round(klassengr[1]*(YesSize))
   samp_n<-round(klassengr[2]*(YesSize))
   a<-Sys.time()
      for(i in 1:numberOfTrees){
           #konstruiere jedes Mal einen neuen Trainingsdatensatz
           #Replace=TRUE -> Bagging (mit Zurücklegen)
           #Replace=FALSE -> "UnderSample2" (ohne Zurücklegen)

        trainingSampleNo <- sample(1:NoSize, samp_n, replace=REPLACE)
        trainingSampleYes <- sample(1:YesSize, samp_y,replace=REPLACE)
        traindata<-rbind(TrainNo[trainingSampleNo,],
                                      TrainYes[trainingSampleYes,])

        forests[[i]] <- randomForest(
                         FORMULA, traindata,
                         ntree=1, #Nur ein Baum pro Durchlauf
                         replace=FALSE,
                          #FALSE damit man selbst bestimmen kann,
                        #welche Daten der CART-Alg. verwendet
                         sampsize=c(yes=samp_y,no=samp_n)
                          #Wieviel er pro Klasse ziehen soll.
                          #gemeinsam mit replace=False, wird der
                          #Baum auf der gewünschten Trainingsmenge
                          #konstruiert
                        )
                    }
     b<-Sys.time()
```

```
          time<-b-a
          print(time)
      }

if(method=="OverSample"){
    print("OverSample")
    a<-Sys.time()
    for(i in 1:numberOfTrees){
        trainingSampleNo <- sample(1:NoSize,
                        round(klassengr[1]*(NoSize)), replace=REPLACE)
        trainingSampleYes <- sample(1:YesSize,
                        round(klassengr[2]*(NoSize)), replace=TRUE)

        traindata<-rbind(TrainNo[trainingSampleNo,],
                        TrainYes[trainingSampleYes,])

        forests[[i]] <- randomForest(traindata$y~., traindata,
                                ntree=1,
                                replace=FALSE,
                                sampsize=c(yes=round(klassengr[1]*(YesSize)),
                                           no=round(klassengr[2]*(YesSize)))
                            )
                        }
       b<-Sys.time() time<-b-a print(time)
   }
 ensemble <- list(numberOfTrees = numberOfTrees,
                        forests = forests, Testdata=testdata)
 class(ensemble) <- append(class(ensemble), "MyRandomForest")
 return(ensemble)
}
```

Predict My Random Forest:

```
predict.MyRandomForest <- function(object, newData) {
  pred <- cbind(sapply(object$forests,
                function(rf) predict(rf, newData, type = "response")))
  result <- pred[, 1]
    for(i in 1:dim(newData)[1]) {
          candidates <- pred[i, ]
          ind <- which.max(sapply(unique(candidates),
                                  function(x) sum(candidates == x)))
          result[i] <- unique(candidates)[ind] } return(result)
      }
   return(result)
}
```

Literaturverzeichnis

Anderson, E. (1935). The irises of the gaspe peninsula. *Bulletin of the American Iris Society*, 59:2–5.

Becker, R., Chambers, J., and Wilks, A. (1988). *The New S Language*. Wadsworth, New York.

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

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

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Chapman and Hall, New York.

Chen, C., Liaw, A., and Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley*.

Dietterich, T. G. and Kong, E. B. (1995). Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Department of Computer Science, Oregon State University.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer series in statistics Springer, Berlin, 2 edition.

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

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

Sikora, M. and Wrobel, L. (2015). seismic-bumps data set.

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

Strobl, C., Boulesteix, A.-L., Zeileis, A., and Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8:25.

Yang, F., Wang, H., Mi, H., Cai, W., et al. (2009). Using random forest for reliable classification and cost-sensitive learning for medical diagnosis. *BMC Bioinformatics*, 10(Suppl 1):22.