Ostfalia
Hochschule für angewandte Wissenschaften

Fakultät Informatik

Interaktive Kontrafaktische Analysen in Entscheidungsbäumen unter Rand- und Nebenbedingungen

Denise Langhof
Matrikel-Nr. 70457263

Masterarbeit im Studiengang Informatik
zur Erlangung des akademischen Grades:
Master of Science

Ostfalia Hochschule für angewandte Wissenschaften

1. Prüfer: Prof. Dr.-Ing. habil. Dirk Joachim Lehmann
2. Prüfer: Prof. Dr.-Ing. Frank Höppner

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist.

Goslar, 03.04.2023

Ort, Datum

Unterschrift

Zusammenfassung

Eine große Herausforderung im Bereich des maschinellen Lernens besteht darin, die Modelle zu interpretieren und ihre Entscheidungen zu erklären. Der Erklärungsbedarf ergibt sich in bestimmten Bereichen nicht nur aus gesetzlichen Vorgaben, sondern auch um das notwendige Vertrauen der Anwender zu erlangen. Dabei liegt es teilweise in der Natur des Modells, besser interpretierbar zu sein als andere Modelle. So sind Entscheidungsbäume oder -regeln leichter zu interpretieren als Künstliche neuronale Netze (KNNs). Unterschiedliche Anforderungen führen jedoch zur Anwendung unterschiedlicher Algorithmen. Beispielsweise kann es bei nichtlinearen Problemen sinnvoll sein, KNNs zur Modellbildung zu verwenden. Stehen jedoch nur geringe Datenmengen für das Training zur Verfügung, kann es zweckmäßig sein, einen anderen Algorithmus zu wählen.

Die Struktur von Entscheidungsbäumen bietet den klaren Vorteil, dass sie durch eine Visualisierung des Baumes leicht interpretierbar sind. So ist es möglich, dass verschiedene Algorithmen und Pruning-Verfahren anhand der Visualisierung eines Entscheidungsbaumes miteinander vergleichbar sind. Lokale Erklärungen über Klassifizierungen sowie Regressionen einer Instanz können durch die Darstellung von Klassifizierungspfaden in Entscheidungsbäumen abgeleitet werden.

Eine weitere Form lokaler Erklärungen ist die kontrafaktische Analyse. Kontrafaktuale können in der Form *Wenn [A] der Fall wäre, dann wäre [B] der Fall* definiert werden. Ein Anwendungsfall aus dem Bankwesen wäre die Vergabe eines Kredits. Der Kreditnehmer wird aufgrund aktueller Faktoren wie Alter, Beruf und Gehalt als nicht kreditwürdig eingestuft. Der Kreditnehmer möchte nun wissen, bei welcher Veränderung der Faktoren er als kreditwürdig eingestuft werden kann. Hier gibt es jedoch verschiedene Rand- und Nebenbedingungen. Sein Alter kann er nicht beeinflussen, er könnte aber den Beruf wechseln und damit sein Gehalt erhöhen. Allerdings kann er sein Gehalt nur bis zu einem bestimmten Betrag erhöhen. Ziel ist es, die Bonität mit minimaler Veränderung der Attribute zu erreichen. Dabei wäre es von Vorteil, aus einer Vielzahl von Kombinationen auswählen zu können, um die für den Anwendungsfall geeigneten Faktoren zu finden.

Die Ziele der kontrafaktischen Analyse sind zum einen, die Erklärungen für eine Entscheidung besser zu verstehen und zum anderen, mögliche Änderungen einer Instanz zu identifizieren, um die Entscheidung in ein anderes Ergebnis umzuwandeln. Um kontrafaktische Erklärungen generieren zu können, kommen Optimierungsverfahren zum Einsatz. Aus einer Menge an möglichen Lösungen muss die beste gefunden werden. Ein solches Problem kann durch geeignete Optimierungsverfahren gelöst werden. Je nach Zielfunktion und Nebenbedingungen gibt es eine ganze Reihe solcher Verfahren.

Ziel der Arbeit war die Klärung der Frage, ob die Interpretierbarkeit der kontrafaktischen Ergebnisse mit Hilfe von Entscheidungsbäumen gewährleistet ist und inwieweit diese ggf. noch verbessert werden kann. Daraus ergab sich die Motivation, eine neue interaktive Anwendung

zur kontrafaktischen Analyse auf der Basis von Entscheidungsbäumen zu entwickeln. In dieser kann der Anwender zum einen lokale Erklärungen für einzelne Instanzen aber auch globale Erklärungen am Entscheidungsbaum selbst problemlos ableiten. Der Anwender ist damit in der Lage neue Instanzen zur Vorhersage zu erzeugen und auf Basis dieser die Analyse durchzuführen. Der Fokus lag dabei auf der freien Konfiguration von Modellalgorithmen und -parametern sowie der Eingabe von Rand- und Nebenbedingungen für die kontrafaktische Analyse. Die kontrafaktische Generierung wurde mithilfe einer evolutionären Pareto-Optimierung umgesetzt.

Für die Auswertung dieser Arbeit wurde eine Nutzerstudie durchgeführt, welche mit anderen Ergebnissen aus dem Bereich der Visualisierung von kontrafaktischen Analysen verglichen wird. Aus diesen Ergebnissen wurden Handlungsempfehlungen abgeleitet.

Die Resultate der Arbeit sind aufschlussreich, da nahezu alle Probanden die Ergebnisse der kontrafaktischen Analyse nachvollziehen konnten und diesen Vertrauen geschenkt haben. Sie lässt sich frei konfigurieren und somit auch gut auf verschiedene Domänen abbilden. Darüberhinaus bietet die Anwendung eine gute Voraussetzung für weitere Arbeiten, wie z.B. der Verwendung dieser Methode auf nicht natürlich interpretierbare Modelle, um Erklärungen aus diesen ableiten zu können.

Inhaltsverzeichnis

1. Einleitung

Der Schwerpunkt dieser Arbeit liegt auf der interaktiven Analyse von Kontrafaktualen auf Basis von Entscheidungsbaummodellen, der Anwendung von Optimierungsmethoden unter verschiedenen Nebenbedingungen, welche durch einen Nutzer frei parametrisierbar sind, sowie der Nachvollziehbarkeit der daraus resultierenden Ergebnisse. Kontrafaktuale oder kontrafaktische Erklärungen sind eine Möglichkeit Erklärungen aus Modellen des maschinellen Lernens ableiten zu können und werden in Kapitel 3.3 genauer erläutert.
Viele existierende Lösungen zur Durchführung kontrafaktischer Analysen setzen einen großen technischen Wissensstand voraus. Visualisierungskonzepte sowie eine freie interaktive Parametrisierung sind in diesem Bereich selten. Darüberhinaus gibt es wenige Veröffentlichungen in denen Ergebnisse anhand von Nutzerstudien evaluiert werden. Das Ziel dieser Arbeit ist die Entwicklung einer intuitiven, interaktiven und visuellen Anwendung zur kontrafaktischen Analyse, welche es auch Nicht-Experten ermöglicht diesbezügliche Analysen eigenständig durchzuführen. Das Fehlen einer geeigneten Lösung motiviert zur Entwicklung einer neuen interaktiven Anwendung, die es dem Anwender ermöglicht, auf der Basis eines Datensatzes Entscheidungsbäume zu generieren und kontrafaktische Analysen durchzuführen. Der Fokus liegt auf der Nachvollziehbarkeit der Ergebnisse, indem eine Visualisierung der trainierten Modelle erfolgt. Zur Evaluierung der Anwendung wird eine Nutzerstudie durchgeführt werden, bei der die Probanden verschiedene Aufgaben mithilfe der Anwendung bearbeiten und anhand eines Fragebogens bewerten.
Im Folgenden wird die Struktur dieser Arbeit erläutert. In Kapitel 2 werden die Literaturen und verwandte Arbeiten beschrieben. Anschließend wird in Kapitel 3 ein gemeinsames Verständnis für die weiterführenden Themen geschaffen. Kernthemen sind die Bedeutung der Erklärbarkeit sowie Modelle des maschinellen Lernens mit Fokus auf Entscheidungsbäumen und der kontrafaktischen Analyse. In diesem Zusammenhang werden verschiedene Algorithmen zur Modellierung von Entscheidungsbäumen und Pruning-Methoden vorgestellt. Die Behandlung möglicher Optimierungsverfahren erfolgt in Kapitel 3.4, in dem ein Einblick in Optimierungsprobleme verschiedener Art gegeben und deren Unterschiede aufgezeigt werden. Darüber hinaus werden in diesem Kapitel Optimierungsverfahren auf Entscheidungsbaumalgorithmen behandelt. Mit Hilfe der behandelten Literatur wird in Kapitel 4 der Ansatz für die Entwicklung der Anwendung beschrieben. Des Weiteren werden die Unterschiede zu bestehenden Lösungen der kontrafaktischen Analyse aufgezeigt. Es wird eine Anwendung zur interaktiven kontrafaktischen Analyse entwickelt, deren Implementierung in Kapitel 5 beschrieben wird. Die Evaluierung des Programms und der Ergebnisse erfolgt anschließend im Kapitel 6 mit Hilfe einer Nutzerstudie, die im darauf folgenden Kapitel 7 ausgewertet und diskutiert wird. Hier werden auch Defizite kontrafaktischer Analysen und Grenzen der Erklärbarkeit von Modellen des maschinellen Lernens aufgezeigt. Die Behandlung neuer Fragestellungen und Themen für zukünftige Arbeiten erfolgt in Kapitel 8.

2. Referenzen und verwandte Arbeiten

Innerhalb dieses Kapitels werden zuerst die wichtigsten Referenzen zu den thematischen Grundlagen dieser Arbeit erläutert. Anschließend werden verwandte Arbeiten zu den kontrafaktischen Analysen sowie zu visuellen Anwendungen vorgestellt.

2.1. Referenzen zu thematischen Grundlagen

Da ein Ziel der Arbeit die Erreichung der Nachvollziehbarkeit von Erkärungen darstellt, werden die Arbeiten von Rijnbee und Huber herangezogen. Die Arbeiten behandeln die Grundlagen für erklärbare und interpretierbare Modelle des maschinellen Lernens [44], welche in Kapitel 3.2 behandelt werden. Huber erklärt in seiner Arbeit die Notwendigkeit der Interpretierbarkeit und Erklärbarkeit insbesondere in Kombination mit sogenannten Black-Box Modellen. Black-Box Modelle verbergen - im Gegensatz zu White-Box-Modellen - die innere Struktur, so dass es eine Herausforderung ist, aus diesen Modellen Erklärungen abzuleiten. In diesem Zusammenhang geht er auch auf Surrogat-Modelle ein und listet verschiedene methodische Ansätze auf [7]. Surrogat-Modelle werden in Kapitel 3.2.2 erläutert. In *Benchmark and survey of Frameworks for Automated Machine Learning* ist die Vorbereitung von Daten behandelt [82], welche notwendigerweise vor Verarbeitung dieser Daten erfolgen muss. Diese Arbeit wird in Kapitel 3.2.3 referenziert. Innerhalb der Veröffentlichung *Explanation Framework for Intrusion Detection* beschreiben Huber et al. darüberhinaus die kontrafaktische Analyse von Black-Box Modellen in ihren einzelnen Schritten anhand eines Beispiels [81]. Ein Ausblick zu dieser Methode wird in Kapitel 8.1 gegeben.

Da kontrafaktische Erklärungen mit Hilfe von Optimierungsmethoden generiert werden können, bilden Arbeiten von Jarre, Papagerorgiou und Pieper die Grundlage für das Kapitel 3.4 über Optimierungsverfahren [30, 53, 56]. Einen umfassenden Überblick über multikriterielle Optimierungsverfahren geben auch Marler und Scholz [46, 69]. Das Werk von Nissen legt den Schwerpunkt auf evolutionäre Optimierungsverfahren und zeigt Anwendungsgebiete für diese auf. Diese werden in Kapitel 3.4.5 erläutert. Formale Ansätze zur Optimierung sind in einer Veröffentlichung von Duong et al. aufgelistet und darüber hinaus sind in dieser Veröffentlichung auch Metriken zur Bewertung von Optimierungen genannt [21], um die angewandte Optimierungsmethode evaluieren zu können. Darüber hinaus wurde versucht, auf mögliche Probleme, die bei kontrafaktischen Analysen auftreten können, vorbereitet zu reagieren. Keane et al. identifizieren fünf verschiedene Defizite, die bei der Generierung von kontrafaktischen Erklärungen auftreten können [32]. Diese werden teilweise in Kapitel 7.3 behandelt.

Für die Visualisierung der Anwendung werden Konzepte von Munzner referenziert [50]. Sie stellt sowohl ein allgemeines Vorgehen als auch kleinere Designentscheidungen vor, welche dabei unterstützen Eingabedaten zu visualisieren. Zu den Designentscheidungen gehören z.B. Farb-

markierungen, welche dem Benutzer helfen, eine Bestätigung für eine ausgeführte Aktion zu erhalten. [50].

Zu guter Letzt wurde für die Entwicklung eines Fragebogens zur Nutzerstudie verschiedene Arbeiten herangezogen. Hegner et al. definieren Klassifikationen von Evaluationsmethoden sowie Evaluierungsmethoden zur Datenerfassung [27]. Thielsch et al. bieten darüber hinaus einen Ansatz zur Erstellung von Fragebögen zur Evaluation von Websites [65]. Ferner wird der Fragebogen ISONORM9241/ 110-S zur Beurteilung von Software als Unterstützung genutzt [17]. Alle geben Anregungen, um das Ziel der Nutzerstudie erreichen zu können. Das Ziel dieser Studie ist die qualitative Beurteilung der Anwendung sowie die Überprüfung der Nachvollziehbarkeit der Ergebnisse.

2.2. Verwandte Arbeiten der kontrafaktischen Analyse

Zuerst musste ein Überblick über verschiedene Lösungen zur kontrafaktischen Analyse geschaffen werden, um diese filtern und vergleichen zu können. Verma et al. publizierten eine Übersicht diverser Veröffentlichungen, welche Verfahren für das Generieren von Kontrafaktuale thematisieren und zwischen 2017 und 2022 veröffentlicht worden sind [73]. Dabei ordnen sie den verschiedenen Lösungen auch die Form der Eingabedaten zu, wie bspw. textuelle Form, Bild- oder Sprachdateien. Huber et al. haben auch eine Liste verschiedener Ansätze zur kontrafaktischen Generierung zusammengestellt, wie z.B. ein triviales Trial-and-Error-Verfahren, bei dem Attribute zufällig verändert werden [7, S. 288]. Innerhalb dieser Arbeiten wurden folgende ähnliche Ansätze gefunden.

Carreira-Perpiñán evaluieren den Einsatz des Algorithmus TAO als Entscheidungsbaum zu kontrafaktischen Analysen [9]. In dieser Arbeit wird jedoch keine Visualisierung als Hilfsmittel verwendet. Der Algorithmus spielt in dieser Arbeit auch eine wichtige Rolle und wird in Kapitel 3.5.3 erläutert.

Da für die Entwicklung der Anwendung die Programmiersprache Python verwendet wird, werden nur Lösungen auf Basis von Python vorgestellt. Es gibt einige Programmbibliothek für die kontrafaktische Generierung, wie Dice, alibi und crfnet [49, 33, 31]. Dice unterstützt drei verschiedene Methoden zur Generierung von Kontrafaktualen [49]. Dazu gehört das Finden zufälliger Stichproben oder auch die Anwendung genetischer Algorithmen, welche in Kapitel 3.4.5 erläutert werden [49]. Neben kontrafaktischen Erklärungen bietet alibi auch eine Reihe an weiteren Arten von Erklärungen für Modelle des maschinellen Lernens [33]. Dazu gehört z.B. SHAP welches innerhalb dieses Kapitels noch einmal erläutert wird [42]. Auch bietet alibi die Möglichkeit der Eingabe verschiedener Arten von Daten. crfnet bietet nur die Möglichkeit der kontrafaktischen Analyse auf Basis von Regressionsaufgaben [31]. Wiederum können Dice und alibi nur Analysen auf Basis von Klassifikationsaufgaben durchführen. Bei diesen Lösungen handelt es sich lediglich

um Programmbibliotheken ohne Visualisierung.

Neben Wachter liefern auch Dandl et al. einen Ansatz zur kontrafaktischen Analyse und die zugrundeliegenden mathematischen Ansätze der multikriteriellen Optimierung im Kontext von Black-Box Modellen [75, 15]. Auf dieser Basis hat Monteiro einen eigenen Ansatz für die kontrafaktische Generierung entwickelt [62, 61, 60]. In diesem Beitrag stellt er seinen Ansatz im Rahmen einer in Python geschriebenen Lösung zusammen mit den Ergebnissen vor. Er vergleicht diese wiederum mit den bereits genannten Bibliotheken Dice, alibi und crfnet [49, 33, 31] und nennt entscheidende Vorteile seiner Lösung diesen gegenüber. Diese Ansätze werden in Kapitel 4 noch einmal genauer erläutert, da für das Verständnis vorher grundlegende Begriffe in Kapitel 3 erklärt werden müssen.

2.3. Verwandte Arbeiten mit visueller Anwendung

Im Folgenden werden ähnliche Lösungen zu visuellen erklärbaren Modellen des maschinellen Lernens vorgestellt.

Einen Ansatz aus der Spieletheorie bietet SHAP, bei dem die Eingabedaten auf die Spieler und das Modell auf dem Spiel selbst abgebildet werden kann [42]. Wie auch der Ansatz LIME, welcher ebenfalls in diesem Abschnitt behandelt wird, bietet es Erklärungen für verschiedene Modelltypen [58]. Dabei erhalten die Merkmale einen sogenannten SHAP-Wert, welcher die Wichtigkeit von Merkmalen aufzeigt [42]. Diese Informationen können auch visualisiert werden. Es können jedoch mit SHAP keine kontrafaktischen Erklärungen erzeugt werden.

Eine weitere Möglichkeit, um Modelle des maschinellen Lernens zu interpretieren wäre die Anwendung von LIME bzw. sp-LIME zur Interpretation von Klassifikations- und Regressionsaufgaben [58]. Die Kernkompetenz von LIME sind lokal interpretierbare modelagnostische Erklärungen. Modellagnostisch bedeutet, dass die Erklärungen unabhängig vom Model funktionieren. Innerhalb dieses Ansatzes werden Vorhersagen jedes Black-Box Modells erklärt, indem diese zu natürlich interpretierbaren Modellen transformiert werden [58]. Diese aus der Transformation entwickelten Modelle werden auch Surrogatmodelle genannt. Es können mit diesem Ansatz jedoch keine kontrafaktischen Erklärungen erzeugt werden. LIME bietet zum einen die Möglichkeit der Visualisierung als HTML und zum anderen können alle Informationen mit Hilfe der Python-Bibliothek *matplotlib* visualisiert werden. Bei SHAP und LIME handelt es sich in erster Linie um Anwendungen für Experten, da diese nicht interaktiv über eine Anwendung bedient und parametrisiert werden können [58, 42].

Um einen aussagekräftigen Vergleich herstellen zu können, mussten Arbeiten gefunden werden, welche ebenfalls kontrafaktische Analysen behandeln. Keane et al. haben im Jahr 2021 ein Paper veröffentlicht, in dem sie verschiedene Veröffentlichungen im Kontext kontrafaktischer Analysen verglichen haben [32]. Dabei ist herausgekommen, dass lediglich 31% dieser Veröffentlichungen

eine Nutzerstudie durchgeführt haben. Einige dieser Studien vergleichen dabei die Methode zur kontrafaktischen Generierung selbst. Somit schränkte sich die Suche weiter auf Nutzerstudien ein, welche ebenfalls eine Visualisierung als Hilfestellung zur Interpretation der Ergebnisse verwendet.

Wexler et al. haben eine eigene Lösung publiziert für das automatisierte Identifizieren von Kontrafaktualen anhand bestimmter Datenpunkte [78]. Die Visualisierung findet anhand eines Streudiagramms von gruppierten Attributen mit Unterstützung durch farbliche Markierung statt. Für die Evaluierung wurden drei verschiedene Anwendungsfälle erzeugt. Bei einem wurde die kontrafaktische Analyse von mehreren Studenten evaluiert. Diese haben verschiedene Datenpunkte analysiert und durch das Experimentieren Kontrafaktuale mehr oder weniger gezielt generiert [78]. Die Nutzerstudie zeigt, dass sich die Anwendung für Nicht-Experten eignet. Durch verschiedene fortgeschrittene Parametrisierungen, wie bspw. die Auswahl von Distanzmetriken ist diese jedoch eher für Experten geeinget. Distanzmetriken werden im Kapitel 4.5.1 näher erläutert. Darüberhinaus können viele Evaluierungsmetriken, wie z.B. ROC-Kurven, welche in Kapitel 3.5.3 erläutert werden, zu den Ergebnissen visualisiert werden.

Mithilfe des CX-TOM Frameworks werden kontrafaktische Analysen auf Bildern mithilfe eines Black-Box Modells ermöglicht [2]. Nachdem der Anwender für ein ausgewähltes Bild eine Klasse vorhergesagt bekommt, kann dieser ein alternatives Ergebnis auswählen und bekommt daraufhin die Bereiche des Bildes angezeigt, welche für die alternative Klassifizierung falsch eingeordnet sind. Zur Evaluierung wurden 150 Probanden ohne technisches Vorwissen und 60 weitere mit diesem Wissen befragt. Es wurde das Vertrauen und die Zufriedenheit über die Erklärung gemessen und im Vergleich zu bspw. LIME oder SHAP sehr gute Ergebnisse erzielt [58, 42, 2]. Die Visualisierung ist einfach gestaltet und somit für Nicht-Experten geeignet. Dies wird bestätigt mit der erfolgreichen Durchführung der Aufgaben durch Nicht-Experten bei der Nutzerstudie.

Innerhalb der Veröffentlichung von Cheng et al. wurde ebenfalls versucht Kontrafaktuale anhand von Visualisierungen zu erklären und anschließend mithilfe einer Nutzerstudie zu evaluieren [11]. In der Visualisierung können neue Instanzen mit Hilfe von Slidern erzeugt werden. Darüberhinaus kann die Anzahl der Kontrafaktuale und die Spärlichkeit definiert werden. Die Spärlichkeit definiert den Unterschied von Originalinstanz und Kontrafaktual. Ist eine Instanz konfiguriert, werden die Kontrafaktuale gesucht und anhand von Polylinien entlang paralleler Achsen visualisiert. Es gibt eine spezielle Untergruppen-Ansicht, in welcher der Datensatz in Untergruppen eingeteilt und die kontrafaktische Analyse in diesem Bereich durchgeführt werden kann. Es werden sehr viele verschiedene grafische Darstellungen verwendet, wie Histogamme zu der Verteilung jedes Attributs. Über die Studie wurde zum Einen die Nutzung der Anwendung von drei technisch versierten Personen kommentiert und bewertet und anschließend der Nutzen der Anwendung von drei fachlichen Experten einer Domäne erläutert. Diese Anwendung ist für Nicht-Experten ungeeignet, da die Parametrisierung viel technisches Vorwissen voraussetzt.

3. Thematischer Hintergrund

In diesem Kapitel wird die Relevanz von erklärbaren und interpretierbaren Modellen des maschinellen Lernens erläutert. Weiterhin befasst sich dieses Kapitel mit der Vorstellung von Entscheidungsbäumen als Metrik für Modelle. Abschließend werden kontrafaktische Analysen behandelt und anhand von Anwendungsfällen veranschaulicht.

3.1. Erklärbarkeit und Interpretierbarkeit im Maschinellen Lernen

Die Begriffe Erklärbarkeit und Interpretierbarkeit haben im Bereich des maschinellen Lernens eine unterschiedliche Bedeutung. Interpretierbarkeit wird im Sinne des Modells als dessen natürliche Eigenschaft verstanden [7]. Beispielsweise sind Entscheidungsbäume in der Regel von Natur aus interpretierbar, wohingegen Künstliche Neuronale Netze nicht interpretierbar sind. Zu den interpretierbaren Modelltypen gehören lineare Modelle, Entscheidungsbäume und -regeln, Naive Bayes, k-Nächste Nachbarn, interaktive Modelle und Bayesianische Netze [7, S. 258]. Erklärbarkeit hingegen ist die Fähigkeit, globale oder lokale Entscheidungen auch mit Hilfe von Werkzeugen nachvollziehen zu können. Die Unterscheidung zwischen globalen und lokalen Erklärungen erfolgt im Kapitel 3.1.2.

Definition 1 (Erklärbarkeit) *Ein KI-System ist erklärbar, wenn das Aufgabenmodell intrinsisch interpretierbar ist oder wenn das nicht interpretierbare Aufgabenmodell durch eine interpretierbare und wahrheitsgetreue Erklärung ergänzt wird [44, S. 2].*

Definition 2 (Interpretierbarkeit) *Eine Erklärung ist interpretierbar, wenn:*

1. *die Erklärung eindeutig ist, d.h. sie liefert eine einzige Begründung, die für ähnliche Fälle ähnlich ist (Klarheit),*
2. *die Erklärung ist nicht zu komplex, d.h. in einer kompakt dargestellten Form (Sparsamkeit).*

Interpretierbarkeit beschreibt das Ausmaß, in dem ein Mensch eine Erklärung verstehen kann [44, S. 3].

Um nicht natürlich interpretierbare Modelle interpretieren zu können, gibt es einige Ansätze, wie z.B. die Verwendung von Surrogatmodellen, die im Kapitel 3.2.2 erläutert werden.

3.1.1. Relevanz von Erklärungen

Es gibt verschiedene Gründe für die Bereitstellung von Erklärungen bei Modellen des maschinellen Lernens. Es hängt immer vom Anwendungsfall und der Domäne ab, ob eine Erklärung erforderlich ist [44, S. 3-4]. Huber et al. definieren acht verschiedene Gründe für die Bereitstellung

[7, S. 249]. Dazu gehören neben dem Hauptgrund des Vertrauens auch Kausalität, Übertragbarkeit, Informativität, faire und ethische Entscheidungen, Verantwortlichkeit, Anpassungsfähigkeit und Proxy-Funktionalität.
Aus rechtlicher Sicht fallen diese Entscheidungen in der Datenschutz-Grundverordnung unter Artikel 22 sowie unter Erwägungsgrund 71, wenn die Entscheidung [74]:

1. die Verarbeitung von Personendaten betrifft,
2. ausschließlich auf einer automatisierten Verarbeitung beruht und die rechtliche Wirkung für die betroffene Person entfaltet
3. oder die Person in ähnlicher Weise erheblich beeinträchtigt.

Zusammenfassend ist festzuhalten, dass immer dann, wenn über Menschen entschieden wird, Erklärungsbedarf besteht. Insbesondere in Bereichen sicherheitskritischer Systeme, wie z.B. der Medizin oder dem autonomen Fahren sind Erklärungen zur Vertrauensbildung unumgänglich.

3.1.2. Arten von Erklärungen

Huber et al. erwähnen die Existenz von lokalen und globalen Erklärungen [7]. Bei einer lokalen Erklärung wird nur die Entscheidung einer einzelnen Instanz begründet, während bei globalen Erklärungen das System als Ganzes mit all seinen Handlungen betrachtet wird. Zu den lokalen Erklärungen zählen auch kontrafaktische Erklärungen. Diese werden in Kapitel 3.3 vertieft. Innerhalb dieser Arbeit wird der Ansatz in Abbildung 1 verfolgt, da es sich bei dem Lernalgorithmus um Entscheidunsbäume handelt.

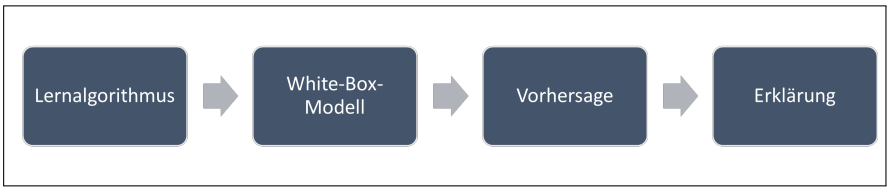


Abb. 1.: Lokale Erklärungen von White-Box Modellen (Grafik inspiriert von Huber et al. [7])

3.2. Überwachtes Maschinelles Lernen

Maschinelles Lernen ist die Fähigkeit zu Lernen ohne explizit für eine Aufgabe programmiert zu werden [39]. Anhand von Eingabedaten wird ein Modell trainiert, um mit diesem Vorhersagen treffen zu können. Beim überwachten maschinellen Lernen stehen im Gegensatz zum unüberwachten maschinellen Lernen Ein- und Ausgabedaten zur Verfügung, um ein Modell zu trainieren. Nach Huber et al. versucht der Lernalgorithmus das Optimierungsproblem wie in Gleichung 3.1 zu lösen. Diese Optimierung kann z.B. anhand des Fehlermaßes erfolgen. Dabei

soll der gemittelte Fehler minimiert und ein Modell h^* ausgegeben werden. Dabei ist h das Eingabemodell, x der Merkmalsvektor der Größe n, y das Ziel und S eine Lossfunktion [7].

$$h^* = \underset{h \in H}{argmin} \frac{1}{n} \sum_{i=1}^{n} S(h(x_i), y_i) \tag{3.1}$$

Es gibt verschiedene Formen der Lossfunktion, welche sich generell in Funktionen für Klassifikations- und Regressionsaufgaben einteilen lassen. Häufig werden sie verwendet, um den Fehler zwischen Vorhersage und tatsächlicher Klasse zu messen [76].
Methoden des überwachten maschinellen Lernens sind z.B. lineare Klassifikatoren und Regressoren, Support-Vektor-Maschinen, Entscheidungsbäume oder auch Künstliche Neuronale Netze.

3.2.1. Entscheidungsbäume als Lernalgorithmen

Entscheidungsbäume sind eine Methode des überwachten maschinellen Lernens. Klassifikation und Regression gehören zu den Aufgaben eines Entscheidungsbaums. Ein Entscheidungsbaum besteht aus Knoten (Wurzelknoten, Entscheidungsknoten, Blattknoten) und diese verbindenden Transitionen. Wenn sich ein Knoten in weitere Knoten aufteilt, nennen sich diese Kindknoten. Der dazugehörige Entscheidungsknoten nennt sich dabei Vaterknoten. Das Klassifikations- bzw. Regressionsergebnis wird dem Blattknoten entnommen. Durch eine Visualisierung dieser Entscheidungsbäume lassen sich je nach Tiefe des Baumes Erklärungen ableiten. Die Tiefe eines Baumes lässt sich aus der Anzahl an Ebenen bestimmen. Dabei befinden sich die Kindknoten eines Vaterknotens in einer Ebene. In der folgenden Abbildung 2 sind die Knoten mit Entscheidungen verbunden, siehe Wurzelknoten (time $\leq$ 73,50). Dabei sind die Entscheidungsknoten blau gekennzeichnet und die Blattknoten in weiß. Der Wurzelknoten wird ganz oben in der Darstellung angezeigt. Ist bei einer zu untersuchenden Instanz der Wert *time* kleiner als der dargestellte Wert, so wird der grüne Pfad zum nächsten Knoten (ejection_fraction $\leq$ 27,50) und bei größerem Wert der rote Pfad (serum_creatinine $\leq$ 1,45) gewählt. Dies wird solange durchgeführt, bis ein Blattknoten, d.h. eine Einstufung in 0 oder 1, erreicht ist.

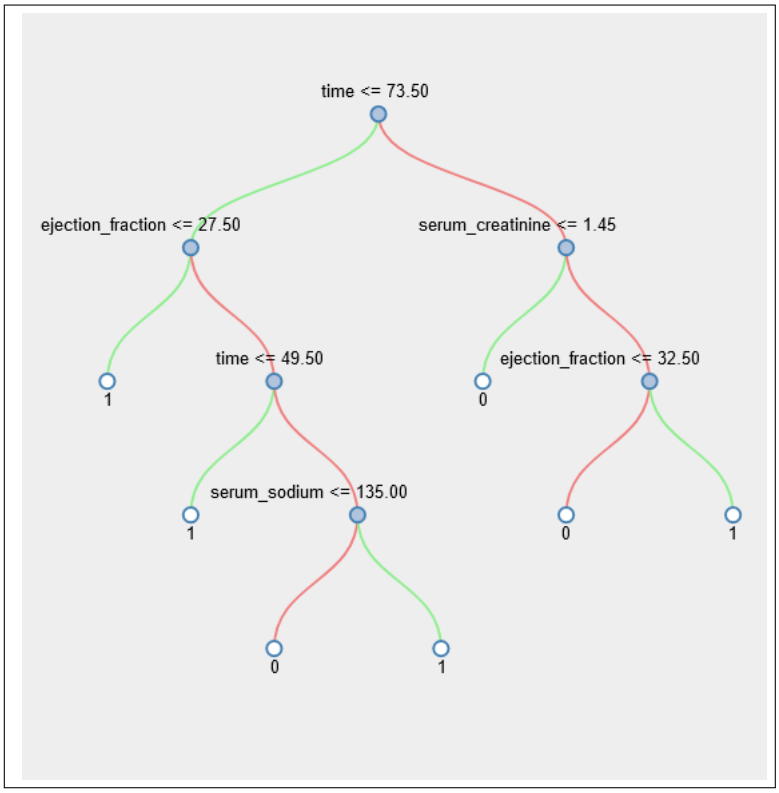


Abb. 2.: Entscheidungsbaum mit stetigen Attributen und binärer Klassifikation

Die Generierung von Entscheidungsbäumen erfordert einen Modellalgorithmus sowie einen Datensatz, dessen Instanzen aus Attributen und einer Zielvariablen bestehen. Die am weitesten verbreiteten Algorithmen sind ID3, C4.5 und CART. Neben diesen Varianten gibt es (Tiefe) Neuronale Entscheidungsbäume, bei denen die Knoten des Entscheidungsbaumes Perzeptronen (künstliche Neuronen) von Künstlichen Neuronalen Netzen darstellen [7, 28]. Dieses Verfahren wird nicht weiter behandelt.
Zu jedem Algorithmus gibt es verschiedene Parameter, welche konfiguriert werden können. Eine Herausforderung ist das Finden von geeigneten Parametern für den entsprechenden Anwendungsfall. Dies wird in Kapitel 3.5.2 behandelt.

Entscheidungsbaumalgorithmus ID3

Der iterative Dichotomisierungsalgorithmus (ID3) wurde bereits 1986 von Quinlan veröffentlicht und basiert auf Prinzipien wie dem Informationsgewinn und der Entropie [80]. Die Entropie ist dabei das Maß für die Unreinheit eines Knotens. Die Unreinheit eines Knotens beschreibt das Maß der Ungenauigkeit einer Klassifikation. Ziel ist es, dieses Maß so klein wie möglich zu halten. Die Entropie des gesamten Datensatzes ist wie folgt definiert. Dabei ist p_i die Auftrittswahrscheinlichkeit der Klasse i bei einer Anzahl von c Klassen im Testdatensatz S. [67].

$$Entropie(S) = \sum_{i=1}^{c} -p_i \cdot log_2 p_i \tag{3.2}$$

Mit Hilfe dieser Entropie wird der Informationsgewinn berechnet. Der Informationsgewinn ist in Gleichung 3.3 definiert. Dabei ist A das betrachtete Attribut, $Werte(A)$ die Menge aller Attributausprägungen und $Entropie(S_v)$ die Entropie für eine Ausprägung . Darüberhinaus ist $|S|$ die Anzahl aller Instanzen und $|S_v|$ die Anzahl der Instanzen mit der Attributausprägung v. Dabei nennt sich der Subtrahent der Gleichung auch bedingte Entropie [67].

$$Informationsgewinn(A) = Entropie(S) - \sum_{v \in Werte(A)} \frac{|S_v|}{|S|} \cdot Entropie(S_v) \tag{3.3}$$

Für jedes Attribut wird zunächst der Informationsgewinn berechnet und das Attribut mit dem höchsten Gewinn als Wurzelknoten gewählt. Anschließend wird dieser Schritt rekursiv für die Kindknoten mit den verbleibenden Attributen wiederholt, bis keine Attribute mehr übrig bleiben. Da ID3 weder Pruning des Baumes noch vorzeitigen Stopp der Generierung unterstützt, wurde mit C4.5 eine optimierte Version geschaffen. Beide Methoden werden in Kapitel 3.5.1 näher erläutert. Darüber hinaus haben die ID3-Modelle eine Tendenz zur Überanpassung.

Definition 3 (Über- und Unteranpassung) *Während bei der Überanpassung das Modell zu stark an die Trainingsdaten angepasst wird und nicht allgemein genug auf andere Datensätze anwendbar ist, wird bei der Unteranpassung das Modell zu stark verallgemeinert. Spezifische Eigenschaften des Modells können so bei der Unteranpassung nicht abgebildet werden.*

Entscheidungsbaumalgorithmus C4.5

Für den Algorithmus C4.5 wurde ein sogenannter vorzeitiger Stopp eingeführt. Dies kann im Rahmen von verschiedenen Methoden erreicht werden. So wird z.B. ein Schwellenwert für den Informationsgewinn eines Knotens eingeführt. Nur wenn der Informationsgewinn der verbleibenden Attribute bei der Rekursion diesen Schwellenwert überschreitet, werden weitere Zweige eingeführt. Eine weitere Methode besteht darin, die Anzahl der Ebenen eines Baumes zu begrenzen [80] .

Entscheidungsbaumalgorithmus CART

Der Modellalgorithmus wurde bereits im Jahr 1984 von Breiman et al veröffentlicht. Er basiert auf einer binären rekursiven Partitionierung [36, S. 4]. Die Generierung des Modells erfolgt in vier Schritten. Im ersten Schritt wird der Baum aufgebaut. Wie bei ID3 wird das Attribut mit

dem höchsten Informationsgewinn für den Wurzelknoten gesucht, um den Knoten in genau zwei Kindknoten zu unterteilen. Für die Auswahl des besten Attributs gibt es verschiedene Methoden, z.B. den Gini-Index oder die Kreuzentropie, die versuchen, eine maximale Reinheit der Kindknoten zu erreichen. Dies wird rekursiv für jeden Knoten durchgeführt, bis nur noch eine Instanz als Kindknoten in Frage kommt oder alle Instanzen die gleiche Verteilung der besten Attribute haben oder eine maximale Baumtiefe erreicht ist. Im nächsten Schritt wird jedem Knoten eine Klasse zugeordnet. Im dritten Schritt wird ein Pruning durchgeführt, um den Baum zu vereinfachen. Schließlich wird mit Hilfe der Kreuzvalidierung versucht, den Komplexitätsparameter α so zu wählen, dass das Modell weder über- noch unterangepasst ist [36].

Definition 4 (Kreuzvalidierung) *Die Kreuzvalidierung ist eine Methode, bei der die Trainingsdaten in mehrere Sätze aufgeteilt werden, um diese gegeneinander zu validieren. Dies kann besonders bei kleinen Datenmengen sinnvoll sein. Dabei gibt es verschiedene Unterformen, wie z.B. die k-fache Kreuzvalidierung oder die Leave-One-Out-Kreuzvalidierung [59].*

Abbildung 3 veranschaulicht eine k-fache Kreuzvalidierung.

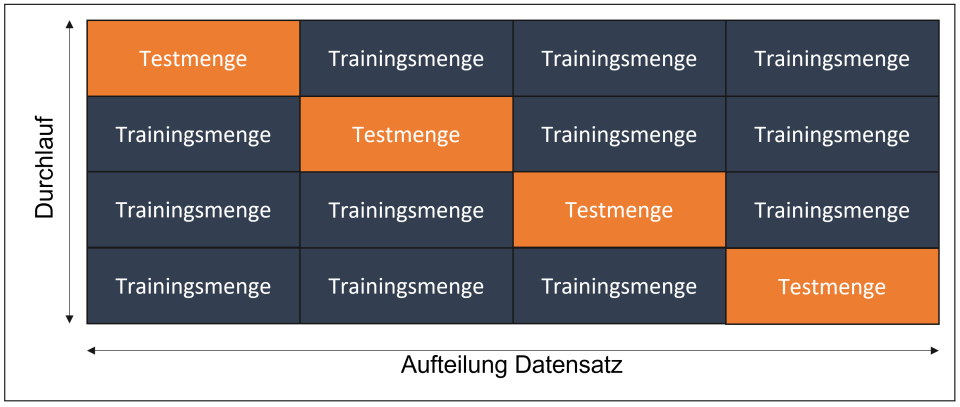


Abb. 3.: 4-fache Kreuzvalidierung (Grafik inspiriert von Phung et al. [55])

3.2.2. Surrogatmodelle zur Interpretierung von Black-Box Modellen

Wie bereits erwähnt ist die Interpretation von Künstlichen Neuronalen Netzen herausfordernd, da das Innere dieser Modelle verschleiert ist. Ein Surrogatmodell wird bei Black-Box Modellen verwendet, um diese in ein interpretierbares Modell umzuwandeln. So kann z.B. ein Künstliches Neuronales Netz in einen Entscheidungsbaum oder in Entscheidungsregeln überführt werden. Zur Generierung dieser Surrogatmodelle existieren verschiedene Ansätze. Während SP-Lime und k-Lime in ein lineares Modell transformieren, gibt es auch einige Modelle, die auf Entscheidungsbäumen basieren [7, S. 271].

Ein möglicher Ansatz zur Generierung solcher Surrogatmodelle besteht aus drei Schritten. Zuerst wird das Black-Box Modell mit einem Trainingsdatensatz trainiert und Vorhersagen für diesen

Datensatz mithilfe des trainierten Black-Box Modells getroffen. Anschließend wird der gleiche Datensatz verwendet und die ursprünglichen Klassifikationen durch die Vorhersagen ersetzt, um einen neuen Datensatz zu erhalten. Im letzten Schritt wird mit dem neuen Datensatz ein interpretierbares Modell für das Ersatzmodell trainiert [66]. Dieser Ansatz ist in Abbildung 4 veranschaulicht.

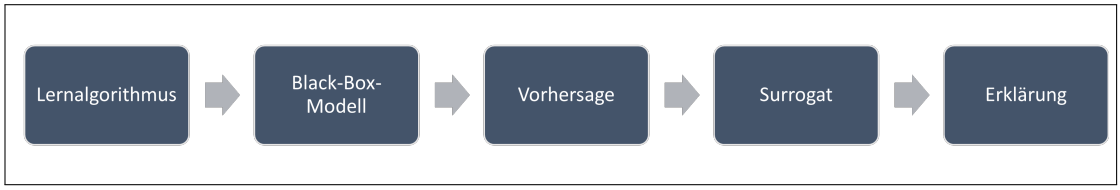


Abb. 4.: Lokale Erklärungen von Black-Box Modellen (Grafik inspiriert von Huber et al. [7])

3.2.3. Datenaufbereitung im Modellerstellungsprozess

Da für die reine Modellgenerierung mit anschließender Anwendung eine Vorbereitung der Daten notwendig ist, müssen weitere Arbeitsschritte berücksichtigt werden. Dazu gehören z.B. die Datenaufbereitung sowie die Merkmalsentwicklung, die dann zusammen mit der Generierung innerhalb einer Pipeline, der ML-Pipeline, als Modellerstellungsprozess dargestellt werden. Ein bekanntes Verfahren ist z.B. die One-Hot-Kodierung, bei der Werte, insbesondere nominale Werte, als Binärwerte dargestellt werden können. Huber et al. nennen zwei verschiedene Verfahren. Zum einen nennen sie ein Basismodell, bei dem innerhalb der Pipeline zunächst die Datenbereinigung, die Merkmalsentwicklung und dann die Modellgenerierung erfolgen. Zum anderen wird ein erweitertes Modell genannt, bei dem die Datenbereinigung in Imputation und Skalierung unterteilt wird, gefolgt von Merkmalsauswahl und Vorverarbeitung und abschließend ebenfalls die Modellgenerierung. Bei der Imputation können neue Daten eingefügt werden, falls diese bspw. fehlen sollten. Mithilfe der Skalierung können Werte standardisiert oder z.B. anhand des größten Wertes skaliert werden. Ziel der Datenbereinigung ist die Verbesserung der Datenqualität durch Eliminierung von Datenfehlern [82, S. 420]. Dazu gehören z.B. fehlende, verrauschte oder Dummy-Werte, kryptische oder widersprüchliche Daten, nicht eindeutige Identifikatoren oder Datenintegrationsprobleme [72]. Diese Datenfehler können durch manuelle Eingaben, sich ändernde Anforderungen an die Daten oder die Zusammenführung von Daten aus verschiedenen Systemen entstehen. Klassische Maßnahmen sind das Ersetzen fehlender Werte, das Entfernen ganzer Instanzen oder das Skalieren von Werten. Es kann auch notwendig sein, Datentypen zu transformieren, da manche Modellalgorithmen nur bestimmte Skalenniveaus unterstützen [82, S. 420]. Ein weiterer wichtiger Bestandteil ist die Merkmalsauswahl, bei der insbesondere das Domänenwissen eine wichtige Rolle spielt.

3.3. Einführung in die Kontrafaktische Analyse

Eine kontrafaktische Erklärung kann Gründe angeben, warum eine Entscheidung getroffen wurde und wie ein gewünschtes Ergebnis erreicht werden kann [75, S. 46]. Sie könnte wie folgt formuliert werden [7]:

1. Wenn x x' gewesen wäre, dann wäre y y' gewesen.
2. Wenn der Schüler anstatt einer Note 4 eine 2 in der Klassenarbeit erhalten hätte, dann hätte er anstatt einer Note 3 eine 2 auf dem Zeugnis bekommen.
3. Wenn der Patient mehr Sport getrieben hätte, dann würde er nun nicht an Herzproblemen leiden.

Dabei sollte bei dem ersten Beispiel x nach einer Distanzmetrik nahe bei x' liegen [81, S. 85]. Das bedeutet, dass sich die Attribute bei einer Änderung der Zielklasse möglichst wenig ändern dürfen.

Definition 5 (Kontrafaktische Erklärung) *Sei $\hat{f} : X \to \mathbb{R}$ eine Vorhersagefunktion, X der Merkmalsraum und $Y' \subset \mathbb{R}$ die gewünschte Ergebnismenge. [...] Eine kontrafaktische Erklärung x' für eine Instanz x* muss als Datenpunkt folgendes erfüllen. 1. Die Vorhersage $f(x')$ liegt nahe an der Ergebnismenge, 2. sie liegt nahe an x* im X Raum, 3. sie unterscheidet sich zu x* in nur wenigen Merkmalen 4. und handelt sich um einen plausiblen Datenpunkt in der Wahrscheinlichkeitsverteilung $\mathbb{P}_w$ [15, S. 4].*

Es gibt verschiedene Herausforderungen, welche bei kontrafaktischen Analysen bewältigt werden müssen. Zu diesen gehören z.B. die Einfachheit der Ergebnisse oder Kausalitäten zwischen den Attributen. Ersteres beschreibt die Ähnlichkeit zwischen der Originalinstanz und dem Kontrafaktual. Wenige Änderungen sind dabei prinzipiell eher gewünscht als Änderungen an vielen Attributen. Bzgl. der Kausalität kann es vorkommen, dass bei Änderung eines Attributes automatisch der Wert eines Attributes geändert wird [73].

3.3.1. Vorstellung verschiedener Anwendungsfälle

Um kontrafaktische Erklärungen zu verdeutlichen, werden im Folgenden Anwendungsfälle aus verschiedenen Bereichen vorgestellt.

Anwendungsfall Bankenwesen

Ein bekannter Anwendungsfall ist aus dem Bereich des Bankenwesens. Eine Person möchte von der Bank einen Kredit erhalten, wird aber aufgrund ihrer aktuellen Lebensumstände als nicht

kreditwürdig eingestuft. Diese Person möchte nun wissen, welche Umstände sie ändern könnte, die maßgeblich zu dieser Entscheidung beigetragen haben, um als kreditwürdig eingestuft zu werden. Da nur zwei Ergebnisse möglich sind (kreditwürdig/nicht kreditwürdig), handelt es sich um eine binäre Klassifikation. Dabei können auch bestimmte Nebenbedingungen relevant sein, die im Kapitel 3.3.2 erläutert werden. Im Anwendungsfall könnte dies bedeuten, dass die Person ihr Gehalt oder ihr Alter nicht ändern darf oder dass das Gehalt nur um einen bestimmten Betrag erhöht werden darf.

Anwendungsfall Medizin

Auch in vielen anderen Bereichen wie der Medizin werden kontrafaktische Analysen eingesetzt, um einerseits Risikofaktoren für eine Erkrankung zu ermitteln und andererseits Maßnahmen zur Verringerung eines möglichen Risikos für diese Erkrankung zu identifizieren. Das Ergebnis kann dann in die Kategorien hohes, mittleres und niedriges Risiko eingeteilt werden. So kann sich beispielsweise herausstellen, dass die betreffende Person einmal pro Woche mehr Sport treiben oder ein bestimmtes Nahrungsmittel meiden sollte, um das Risiko für eine bestimmte Erkrankung zu verringern.

Anwendungsfall Auto-Verkauf

Ein weiterer Anwendungsfall wäre der Verkauf eines Autos. Der Verkäufer möchte wissen, welchen Preis er für sein Auto verlangen kann. Dies ist durch eine einfache Modellprognose möglich. Nun möchte er aber auch wissen, wie er den Wert und damit den Preis des Autos steigern kann, wenn er Maßnahmen wie z.B. Reparaturen durchführen lässt. Das Ergebnis könnte beispielsweise lauten, dass das Auto für 1000 € mehr verkauft werden kann, wenn es TÜV-geprüft wird, und für weitere 250 € mehr, wenn es professionell gereinigt wird. Dabei handelt es sich im Gegensatz zu den anderen Anwendungsfällen um eine Regressionsaufgabe.

3.3.2. Nebenbedingungen der kontrafaktischen Analyse

In einigen Anwendungsfällen kann es durchaus vorkommen, dass Attribute nicht oder nur in einer bestimmten Weise verändert werden dürfen. Um das bestmögliche Ergebnis zu erzielen, können Neben- und Randbedingungen angewendet werden. Beispielsweise ist im medizinischen Bereich das Alter von Personen ein wichtiger Parameter, der in Entscheidungen einfließt, aber nicht verändert werden kann. Die Nebenbedingung, dass ein bestimmtes Attribut geändert werden soll, ist eine Ergänzung dazu, da eine Änderung an dieser Stelle am einfachsten zu erreichen ist. Darüber hinaus gibt es Attribute, die nur in einem bestimmten Umfang verändert werden dürfen, weil es nicht möglich ist oder nicht in die Betrachtung einfließt. Es kann auch sinnvoll sein, nur

eine minimale Anzahl von Attributen zu ändern, um eine geringfügige Änderung zu erzielen. Im Folgenden sind mögliche Nebenbedingungen zu kontrafaktischen Analysen aufgeführt, auf die sich bei der Umsetzung der Anwendung konzentriert wird.

Nebenbedingung 1 (Minimale Anzahl an Attributen) *Es darf sich nur eine minimale Anzahl von Attributen ändern.*

Nebenbedingung 2 (Toleranzbereich der Attribute) *Die Änderung eines Attributs darf einen festen Toleranzbereich nicht überschreiten.*

Nebenbedingung 3 (Unveränderliche Attribute) *Bestimmte Attribute dürfen sich nicht ändern.*

3.4. Einführung in Optimierungsverfahren

Die Generierung von kontrafaktischen Erklärungen ist ein Optimierungsproblem, da die beste Lösung innerhalb zahlreicher möglichen Lösungen gesucht wird. Aus diesem Grund werden in diesem Kapitel mögliche Optimierungsverfahren behandelt.
Für verschiedene Optimierungsprobleme gibt es Methoden, die für die jeweiligen Aufgaben geeignet sind. Dabei muss zwischen einer einfachen Verbesserung und einer Optimierung unterschieden werden [56]. Mathematisch gibt es ein Ziel in Form einer Zielfunktion f(x) wie in Gleichung 3.4, das es zu minimieren oder zu maximieren gilt, und Nebenbedingungen, die auf dieses Ziel anzuwenden sind. P ist der zulässige Bereich [69].

$$min\, f(x),\ sodass\, x \in P \tag{3.4}$$

Wenn die Definitionsmenge $P \neq \mathbb{R}^n$ handelt es sich um ein restringiertes Optimierungsproblem [68]. Das Ziel der Optimierung ist es, die beste (Menge) aller möglichen Lösungen zu finden, indem das Minimum oder Maximum einer Funktion gefunden wird. [77]
Im Folgenden wird zunächst zwischen den verschiedenen Arten von Optimierungsverfahren unterschieden. Zudem werden ausgewählte Verfahren innerhalb der verschiedenen Arten von Optimierungsverfahren vorgestellt.

3.4.1. Arten von Optimierungsverfahren

Optimierungsprobleme können in **analytische und numerische Verfahren** eingeteilt werden. Bei den analytischen Verfahren kann eine exakte Lösung durch Formeln generiert werden, während bei den numerischen Verfahren nur Näherungslösungen möglich sind, da für diese Probleme

keine exakte Formel existiert oder eine schnelle Berechnung nicht gewährleistet werden kann. Analytische Lösungen lassen sich bekanntlich durch Extremwertberechnung mihilfe von Ableitungen berechnen. Lassen sich diese Ableitungen aufgrund der Funktionseigenschaften nicht berechnen, so werden numerische Verfahren angewandt. Auch bei der Optimierung gibt es sowohl notwendige als auch hinreichende Kriterien. Erstere können mögliche optimale Punkte finden, letztere können bestätigen, dass es sich tatsächlich um einen optimalen Punkt handelt. Diese werden weiter in Kriterien erster und zweiter Ordnung unterteilt. Für Optimalitätskriterien erster Ordnung wird die erste Ableitung und für Optimalitätskriterien zweiter Ordnung die zweite Ableitung verwendet. Analytische Optimalitätskriterien lassen sich sehr einfach durch Ableitungen bestimmen. Dabei sei $x*$ ein bekanntes lokales Optimum. Die notwendige Bedingung erster Ordnung, die notwendige Bedingung zweiter Ordnung und die hinreichende Bedingung lauten wie in folgenden Gleichungen. Diese gelten für lokale Minima von Funktionen mit einer Variablen und ohne Nebenbedingungen [53].

$$f'(x^*) = 0 \tag{3.5}$$

$$f'(x^*) = 0 \text{ und } f''(x^*) \geq 0 \tag{3.6}$$

$$f'(x^*) = 0 \text{ und } f''(x^*) > 0 \tag{3.7}$$

Lokale Optimierungsverfahren versuchen, die optimale Lösung innerhalb einer definierten Nachbarschaft zu finden, während bei der **globalen Optimierung** die optimale Lösung innerhalb der gesamten Domäne gefunden werden soll. Die optimale Lösung kann dabei eine einzelne Instanz oder eine Menge von Instanzen darstellen und sowohl Maximum als auch Minimum sein. Wobei bei der Optimierung das Minimum einfach als Negation des Maximums dargestellt wird [77].

Auch die Anzahl der Eingangsparameter spielt eine wichtige Rolle. Während bei **univariaten Verfahren** nur ein Parameter berücksichtigt wird, werden bei **bivariaten** zwei Parameter und bei **multivariaten Verfahren** mehrere Parameter in die Auswertung einbezogen.

Wie bereits erwähnt, gibt es Verfahren mit und ohne Nebenbedingungen. Darüber hinaus gibt es noch **unikriterielle und mutlikriterielle Optimierung**. Bei der unikriteriellen Optimierung, die in den Kapiteln 3.4.2 bis 3.4.5 behandelt wird, wird genau ein Ziel in der Zielfunktion verfolgt. Bei der mutlikriteriellen Optimierung werden dagegen mehrere Ziele verfolgt. Multikriterielle Optimierungsprobleme können auch mit Nebenbedingungen dargestellt werden, indem nur eines der Ziele als Zielfunktion betrachtet wird und die übrigen Ziele als Nebenbedingungen formuliert werden [51, S. 83].

3.4.2. Lineare Optimierung

Lineare Optimierungsprobleme haben, wie der Name bereits vermuten lässt, eine lineare Zielfunktion und besitzen ebenfalls lineare Nebenbedingungen. Im Gegensatz zu nichtlinearen Optimierungsproblemen sind lineare Optimierungsprobleme meist effizient lösbar. Bei folgender Gleichung 3.8 für lineare Programme sind die m x n Matrix A sowie die Vektoren c und b gegeben. Dabei stellt c den Kostenvektor dar und das T steht in dieser gesamten Arbeit für die transponierte Matrix.

$$min\, c^T \cdot x,\ sodass\, A \cdot x \leq b\, und\, x \geq 0 \tag{3.8}$$

Um geeignete x zu finden und die Zielgröße zu minimieren, gibt es die Möglichkeit, eine graphische Optimierung oder die Simplex-Methode durchzuführen, die in den folgenden Kapiteln erläutert werden. Bei einfachen univariaten Problemen reicht eine Extremwertbetrachtung aus, um das Optimum zu finden. In der Regel handelt es sich jedoch um multivariate Probleme, die mit den in den folgenden Kapiteln beschriebenen Methoden gelöst werden können.

Graphische Optimierung

Bei der graphischen Optimierung werden Zielfunktion und Nebenbedingungen graphisch wie in Abbildung 5 dargestellt. Die Zielfunktion kann nun als Normale (schwarze Linien) parallel zum Schnittpunkt der Nebenbedingungen (blaue Linien) verschoben werden, bis sich die Gerade und der Punkt schneiden, um entweder das Ergebnis zu minimieren oder zu maximieren. Die optimalen Parameter x können dann abgelesen werden [30].

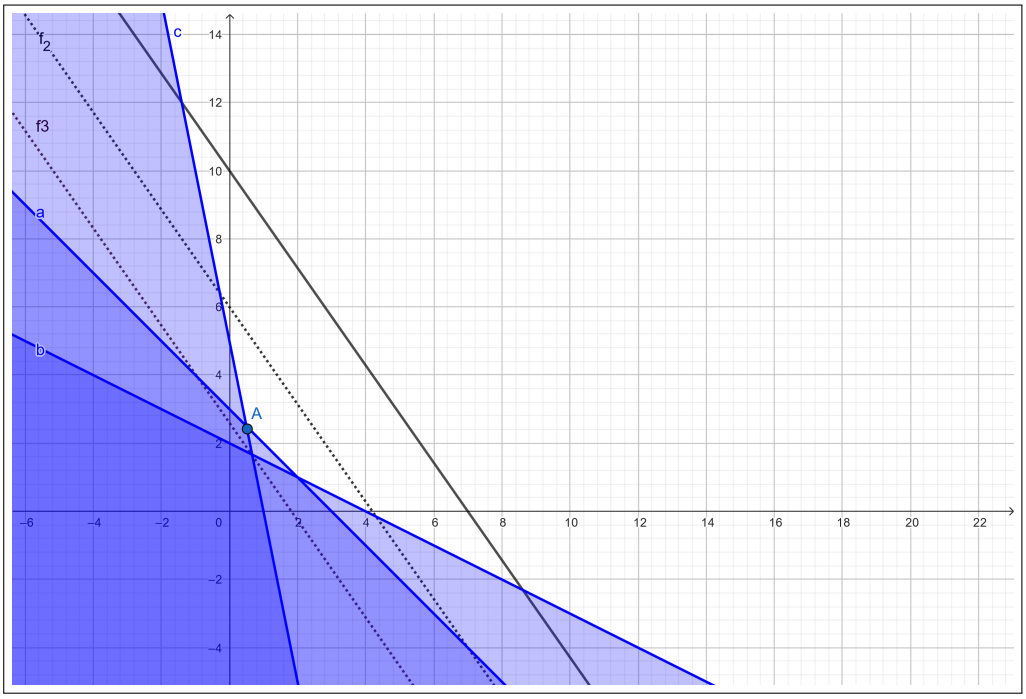


Abb. 5.: Lineare Zielfunktion mitsamt linearer Nebenbedingungen (Grafik inspiriert von Jarre et al. [30])

Simplex-Methode

Das Simplex-Verfahren ist ein Suchverfahren, bei dem iterativ mit Hilfe von Pivotelementen, nach der optimalen Lösung gesucht wird. Die Pivotspalte- und Pivotzeile werden bevorzugt anhand der kleinsten Werte gewählt. Das Ergebnis ist immer ein Optimum oder die Aussage, dass es sich um ein unzulässiges unbeschränktes Problem handelt [69]. Formal kann das lineare Optimierungsproblem in Simplexform äquivalent zu 3.8 wie in Gleichung 3.9 dargestellt werden. Dabei ist z eine freie Variable zur Überführung der Gleichung.

$$max\left\{ z \middle| \begin{bmatrix} A & 0 \\ c^T & 1 \end{bmatrix} \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, x \geq 0 \right\} \tag{3.9}$$

Grafisch lässt sich dies wie in Abbildung 6 anhand eines Polyeders veranschaulichen, welches durch Gleichungen und Ungleichungen beschrieben ist. Ein Simplex ist ein n-dimensionales Polytop mit n + 1 Ecken [30]. Ziel ist es, entlang der Kanten (blaue Linien) einen optimalen Eckpunkt (blaue Punkte) des Polytops zu finden.

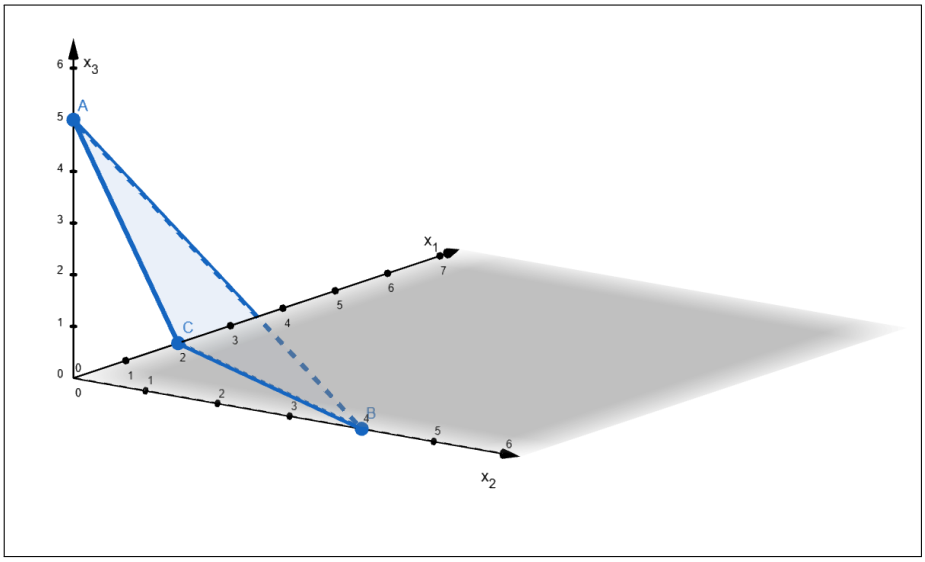


Abb. 6.: Polyeder mit drei Eckpunkten (blau) (Grafik inspiriert von Jarre et al. [30])

Mathematisch muss das lineare Programm in einer einheitlichen Form vorliegen, indem u.a. alle Ungleichungen der Nebenbedingungen mit Hilfe von Schlupfvariablen (Basisvariablen) in Gleichungen überführt und Nicht-Negativitätsbedingungen hinzugefügt werden. Im Gegensatz zum primalen Simplex können beim dualen Simplex negative rechte Seiten der Normalform auftreten. Auch beim dualen Simplex gilt das Dualitätsprinzip, das besagt, dass ein Minimierungsproblem in ein Maximierungsproblem umgewandelt werden kann. Gegeben sei ein lineares Programm in Normalform, siehe Definition 3.8. Das zugehörige duale lineare Programm ist in Gleichung 3.10 definiert. Dabei sei s ein Vektor an Schlupfvariablen [30].

$$max\, b^T \cdot y,\ sodass\ A^T \cdot y - s = c\ und\ s \geq 0\ und\ y \neq 0 \tag{3.10}$$

Durch äquivalente Umformung dieses Satzes, z.B. dass jeder Vektor als Differenz zweier nichtnegativer Vektoren dargestellt werden kann, erhält man den allgemeineren Dualitätssatz, welcher in Gleichung 3.11 definiert ist. Dabei werden y_1 und y_2 als **Lagrangemultiplikatoren** bezeichnet [30].

$$\begin{aligned} min\left\{ c_1^T x_1 + c_2^T x_2 \,\middle|\, \begin{array}{l} A_{11}x_1 + A_{12}x_2 \geq b_1 \\ A_{21}x_1 + A_{22}x_2 = b_2 \end{array} , x_1 \geq 0 \right\} \\ = max\left\{ b_1^T y_1 + b_2^T y_2 \,\middle|\, \begin{array}{l} A_{11}^T y_1 + A_{21}^T y_2 \leq c_1 \\ A_{12}^T y_1 + A_{22}^T y_2 = c_2 \end{array} , y_1 \geq 0 \right\} \end{aligned} \tag{3.11}$$

Bei beiden Methoden wird anfangs ein Simplex-Tableau aus Nebenbedingungen, Zielfunktion und Basisvariablen aufgestellt. Beim dualen Simplex ist der erste Schritt analog zum primalen

Simplex die Transformation der Tabelle, so dass die rechten Seiten positiv sind. Der zweite Schritt beim Dualen Simplex ist die Anwendung des Primalen Simplex. Anhand der Tabelle wird iterativ die neue Basis berechnet, bis diese nicht mehr negativ ist. Sobald dies der Fall ist, ist die optimale Lösung gefunden [30].

Innere Punkte Verfahren

Die inneren Punktverfahren basieren ebenfalls auf dem Dualitätsprinzip. Graphisch kann man sie sich im Gegensatz zum Simplex so vorstellen, dass ein Punkt auf einer Kante nicht entlang der Kante, sondern durch das Polyeder gefunden wird. Auch die Namensgebung erklärt sich aus diesem Grund. Mit Hilfe der Definitionen 3.8 und 3.10 und des Gleichungssystems in Definition 3.12 werden Nullstellen bestimmt, um gleichzeitig eine Lösung x des Primalen und y des Dualen zu finden. Dabei dient μ zur Kompensation des Konfliktes $x*s = 0$, wobei $x > 0$ und $s > 0$. μ wird in jeder Iteration so reduziert, dass $g(x, y, s)$ (entspricht $f(x, y, s)$, ohne μ) gegen eine Nullstelle konvergiert. Alle großgeschriebenen neu eingeführten Zeichen entsprechen der Diagonalmatrix und $e = (1, ..., 1)^T \subset \mathbb{R}^n$ [69].

$$f(x, y, s) = \begin{pmatrix} A \cdot x - b \\ A^T \cdot y + s - c \\ X \cdot s - \mu \cdot e \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.12}$$

Die Nullstelle wird mit Hilfe der Newton-Methode gefunden, aber dazu wird die Jacobi-Matrix benötigt. Diese ist wie folgt definiert [69].

$$Df(x) = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \dots & \frac{\delta f_1}{\delta x_n} \\ \dots & \dots & \dots \\ \frac{\delta f_m}{\delta x_1} & \dots & \frac{\delta f_m}{\delta x_n} \end{bmatrix} \tag{3.13}$$

Definition 6 (Newton-Verfahren) *Mithilfe dieses Verfahrens können Nullstellen iterativ und näherungsweise bestimmt werden. Die Rekursionsvorschrift, auch als Newton-Iteration bezeichnet, ist eine spezielle Form der Fixpunktiteration und der k-te Schritt ist wie in Gleichung 3.14.*

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \tag{3.14}$$

Es ist wichtig, dass bei diesem Verfahren die richtigen Ausgangswerte verwendet werden. Durch Einsetzen des Ergebnisses in die Funktion kann die Genauigkeit überprüft und damit das Abbruchkriterium festgelegt werden [70].

Mit dem gedämpften Newton-Verfahren, bei dem ein Dämpfungsparameter verwendet wird, kann im Gegensatz zum ungedämpften Verfahren eine globale Konvergenz erreicht werden.

Es gibt verschiedene Unterformen der Inneren Punkte Methode, wie z.B. die Kurz-Schritt-Methode, die Prädikator-Korrektor-Methode oder auch die Lang-Schritt-Methode. Sie alle unterscheiden sich in der Wahl von μ. Im ersten Schritt des Kurz-Schritt-Verfahrens müssen die Startparameter x, y, s und μ sowie k = 0 gesetzt und eine Abbruchgenauigkeit $\epsilon > 0$ gewählt werden. Die Ableitungsmatrix von 3.12 bzw. die dazugehörige Jacobi-Matrix ist in Gleichung 3.15 definiert. Dabei ist I_n die Einheitsmatrix dar.

$$Df(x, y, s) = \begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I_n \\ S & 0 & X \end{pmatrix} \tag{3.15}$$

Anschließend sind die der Newton-Richtung entsprechenden Werte $(x^N_{(k)}, y^N_{(k)}, s^N_{(k)})$ über das Gleichungssystem 3.16 zu bestimmen [69, 30].

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I_n \\ S & 0 & X \end{pmatrix} \cdot \begin{pmatrix} x^N_{(k)} \\ y^N_{(k)} \\ s^N_{(k)} \end{pmatrix} = \begin{pmatrix} b - A \cdot x_{(k)} \\ c - A^T \cdot y_{(k)} - s_{(k)} \\ \mu_{(k)} \cdot e - X_{(k)} \cdot s_{(k)} \end{pmatrix} \tag{3.16}$$

Anschließend wird der nächste Iterationsschritt wie in folgender Gleichung 3.17 vorbereitet.

$$(x_{(k+1)}, y_{(k+1)}, s_{(k+1)}) = (x_{(k)}, y_{(k)}, s_{(k)}) + (x^N_{(k)}, y^N_{(k)}, s^N_{(k)})) \tag{3.17}$$

Ansonsten muss μ wie in Gleichung 3.18 gesetzt, k um 1 erhöht und das Lösen des Gleichungssystems mit den neuen Werten wiederholt werden [69].

$$\mu_{(k+1)} = \mu_{(k)} \cdot (1 - \frac{1}{6 \cdot \sqrt{n}}) \tag{3.18}$$

3.4.3. Nichtlineare Optimierung ohne Nebenbedingungen

Im Gegensatz zur linearen Optimierung sind bei der nichtlinearen Optimierung die Zielfunktion oder die Nebenbedingungen nichtlinear. Dazu gehören z.B. die Abstiegsverfahren, zu denen das Gradientenverfahren, das Polak-Ribière-Polyak-Verfahren oder der steilste Abstieg für konvexe quadratische Funktionen gehören. Spezialfälle sind die Optimierung quadratischer Funktionen sowie konvexer und konkaver Funktionen. Erstere unterscheidet sich von der linearen Optimierung nur durch die Zielfunktion, die in Gleichung 3.19 definiert ist und in der C eine symmetrische

Matrix darstellt. Letztere wird in einem eigenen Kapitel 3.4.3 behandelt [68].

$$min\, f(x) = c^T x + \frac{1}{2} x^T C x \tag{3.19}$$

Gradientenverfahren

Das Gradientenverfahren ist eines der bekanntesten Verfahren und die Bestimmung des Gradienten für kartesische Koordinatensysteme ist in Gleichung 3.20 definiert. Der Gradient gibt den steilsten Anstieg einer Funktion an und kann somit die Richtung zu einem Minimum oder Maximum anhand der partiellen Ableitungen der Funktion an der Stelle x angeben [69].

$$\nabla f(x) = (\frac{\partial}{\partial x_1} f(x), ..., \frac{\partial}{\partial x_n} f(x)) \in \mathbb{R}^n \tag{3.20}$$

Ziel ist die Näherung von einem Ausgangspunkt in Richtung des Abstiegs, bis keine weitere Verbesserung mehr erzielt werden kann. Die Richtung wird durch den Gradienten bestimmt. Neben dem Optimierungsproblem $min\, f(x)$ mit stetig differenzierbaren Zielfunktionen $f : \mathbb{R}^n \rightarrow \mathbb{R}$ seien der Startwert $x_{(0)} \in \mathbb{R}^n$, ein Abbruchkriterium ϵ, die Parameter $0 < \alpha < 1$ sowie $\beta > 1$ als Schrittweitensteuerung und schließlich k und $t_{(0)=1}$ als zusätzliche Parameter. Im Allgemeinen wird der Gradient für geänderte Parameter iterativ bestimmt, bis das Abbruchkriterium erreicht ist. Die Schrittweite ist dahingehend von Bedeutung, dass diese nicht zu klein sein darf, um auf der einen Seite nicht nur ein lokales, sondern auch ein globales Optimum zu finden. Das Problem des lokalen Minimums kann auch durch die Wahl unterschiedlicher Startparameter vermieden werden. Eine weitere Möglichkeit wäre auch, mit der Gradientenmethode grob das globale Minimum zu finden und dieses z.B. mit der Newtonmethode zu verfeinern. Hier wird deutlich, dass eine Kombination verschiedener Methoden möglich ist. Grafik 7 verdeutlicht die Anwendung des Gradientenverfahrens, angefangen bei einem Startpunkt A bis das Verfahren bei dem Punkt H konvergiert [69].

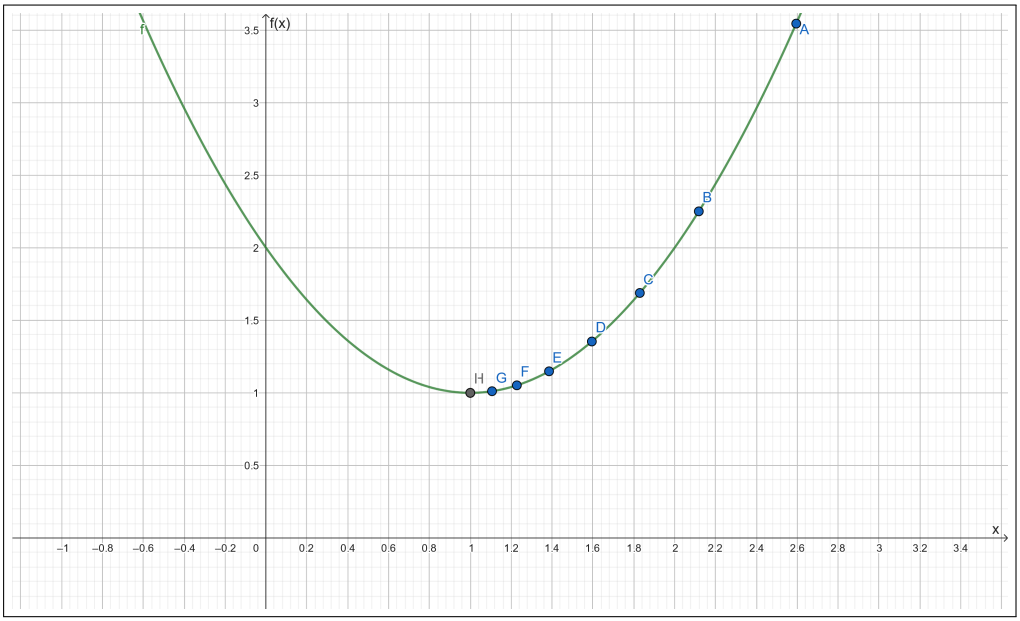


Abb. 7.: Gradientenverfahren mit schrittweiser Neuberechnung der Punkte (blaue) und Optimum (H) (Grafik inspiriert von Ceneda [10])

Neben den Gradientenverfahren gibt es konjugierte Gradientenverfahren, wie z.B. den q-Polak-Ribière-Polyak konjugierten Gradientenalgorithmus oder das Fletcher-Reeves-Verfahren, die bei besonders großen Gleichungssystemen eingesetzt werden [48].

Downhill-Simplex-Verfahren

Das Downhill-Simplex-Verfahren von Nelder und Mead ist wahrscheinlich das am häufigsten verwendete nichtlineare Optimierungsverfahren. Dieses Verfahren ist relativ trivial, liefert aber nicht die qualitativ besten Ergebnisse. Es basiert auf der schrittweisen Neuberechnung der Eckpunkte des Simplex durch Reflexion, Expansion, Kontraktion und Multikontraktion. Dies wird in Grafik 8 verdeutlicht [34].

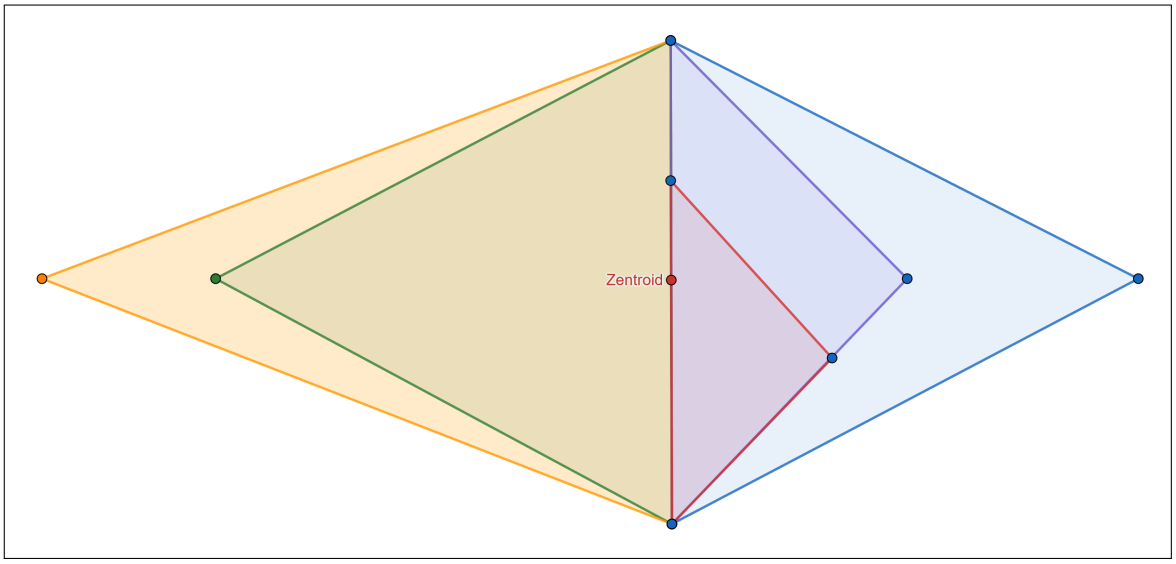


Abb. 8.: Das 2-dimensionale Polytop im Startzustand (blau), Reflexion (grün), Expansion und Reflexion (gelb), Kontraktion (violett), Mehrfachkontraktion (rot) (Grafik inspiriert von Cheng et al. [12])

Im ersten Schritt werden die Startparameter ausgewählt. Anschließend werden die Eckpunkte sortiert und der Zentroid bestimmt. Der schlechteste Eckpunkt wird nun mit den verschiedenen Methoden neu berechnet oder es wird eine Mehrfachkontraktion durchgeführt. Das ganze Verfahren wird dann solange wiederholt, bis ein Konvergenzkriterium erreicht ist [34].

Goldener Schnitt

Das Verfahren des Goldenen Schnitts wird für unimodale Funktionen verwendet. Dies sind Funktionen, die sowohl lokal als auch global genau ein Optimum besitzen, wie z.B. quadratische Funktionen. Die Idee besteht darin, ein berechnetes Intervall schrittweise um einen konstanten Kontraktionsfaktor zu verkleinern, bis ein Konvergenzkriterium erfüllt ist [53].

Quasi-Newton-Verfahren

Im Gegensatz zu den gedämpften Newton-Verfahren versucht das Quasi-Newton-Verfahren den Rechenaufwand zu reduzieren, ohne die Konvergenzgeschwindigkeit zu erhöhen. Als Hilfsmittel wird die inverse Hessesche Matrix verwendet, um die Suchrichtung zu bestimmen. Die Hessesche Matrix ist in Gleichung 3.21 definiert [53].

$$h(f(x)) = \begin{bmatrix} \frac{\delta^2 f}{\delta x_1^2} & \dots & \frac{\delta^2 f}{\delta x_1 \delta x_n} \\ \dots & \dots & \dots \\ \frac{\delta^2 f}{\delta x_n \delta x_1} & \dots & \frac{\delta^2 f}{\delta x_n^2} \end{bmatrix} \tag{3.21}$$

Die Approximation G dieser Matrix wird entweder mit der DFP-Formel (Davidon, Fletcher, Powell) oder der BFGS-Formel (Broyden, Fletcher, Goldfarb, Shanno) berechnet, und die Suchrichtung ist in Gleichung 3.22 bestimmt [53].

$$s^{(l)} = -G^{(l)} \nabla f(x^{(l)}) \tag{3.22}$$

Dabei ist G die Approximation der Hessesche Matrix und l der Iterationsindex. Eine andere Form des Newton-Verfahrens ist das sogenannte Sekantenverfahren, bei dem die Ableitung durch den Differenzenquotienten ersetzt und das Verfahren dadurch vereinfacht wird [53].

Trust-Region Verfahren

Beim Trust-Region Verfahren nähert ein sogenannter Vertrauensbereich iterativ immer weiter an das Optimum an. Dazu wird ein Startwert und ein Vertrauensbereich gewählt und anschließend die quadratische Approximation berechnet, die in Gleichung 3.23 definiert ist. Dabei stellt δ den Vertrauensbereich dar [53].

$$min\Phi(\delta x) = \frac{1}{2}\delta x^T \nabla^2 f(x^{(l)})\delta x + \nabla f(x^{(l)})^T \delta x + f(x^{(l)}) \tag{3.23}$$

Im nächsten Schritt wird die Güte der Approximation anhand der Kostenfunktion berechnet. Dies wird so lange wiederholt, bis das Konfidenzintervall gegen einen bestimmten Wert, im Idealfall 0, konvergiert [53].

Konvexe und konkave Optimierung

Konvexe (konkave) Funktionen sind eine Sonderform der nichtlinearen Funktionen. Für zwei Werte x_1 und x_2 sowie beliebige ganzzahlige Werte $\lambda \in [0, 1]$ gilt für konvexe Funktionen die Ungleichung 3.24 sowie für konkave Funktionen die Ungleichung 3.25 [68].

$$f(\lambda x_2 + (1-\lambda)x_1) \leq \lambda f(x_2) + (1-\lambda) f(x_1), \tag{3.24}$$

$$f(\lambda x_2 + (1-\lambda)x_1) \geq \lambda f(x_2) + (1-\lambda) f(x_1) \tag{3.25}$$

Grafisch sind diese Funktionen in Abbildung 9 exemplarisch abgebildet.

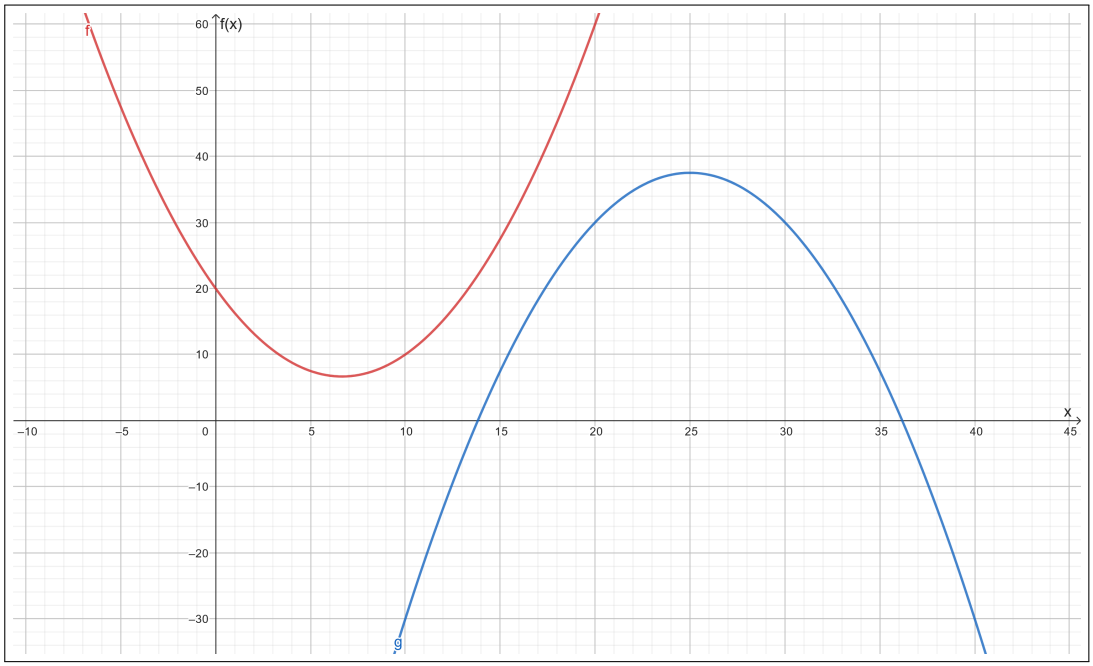


Abb. 9.: Eine konvexe (rot) und konkave (blau) quadratische Funktion

Die Besonderheit besteht darin, dass bei konvexen Funktionen jedes lokale Minimum gleichzeitig das globale Minimum darstellt [69]. Dieses kann mit Hilfe des Subgradientenverfahrens gefunden werden, das sich analog zum Gradientenverfahren aus Kapitel 3.4.3 verhält. Mit Hilfe eines Startpunktes, einer Schrittweite und des Subgradienten wird eine Suchrichtung festgelegt. Nach Aktualisierung des Startpunktes wird das Verfahren solange wiederholt, bis der Subgradient ξ gegen 0 konvergiert. Der Gradient ist ebenfalls der Subgradient der Funktion, wenn die Funktion differenzierbar ist. Für nicht differenzierbare Funktionen muss der Gradient anders berechnet werden. Aus diesem Grund wird an dieser Stelle das Subdifferential berechnet, das eine Menge von Subgradienten darstellt. Die Unterscheidung wird in Abbildung 10 verdeutlicht. Während die Berechnung eines Subgradienten die einfachere Variante darstellt, erweist sich die Berechnung des Subdifferentials als größere Herausforderung. Für beide Ansätze gibt es verschiedene Methoden, die in der Arbeit von Boyd et al. behandelt werden [64].

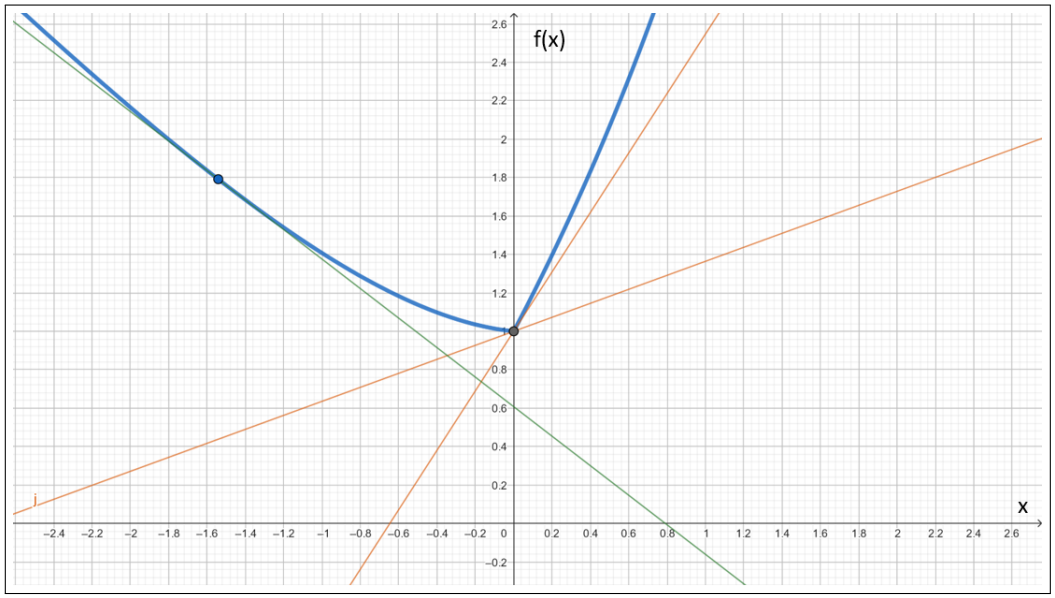


Abb. 10.: Subgradient (grün) sowie Subdifferential (orange) einer nicht differenzierbaren konvexen Funktion (Grafik inspiriert von Yao et al. [79])

3.4.4. Nichtlineare Optimierung mit Nebenbedingungen

Eine weitere Herausforderung ist die nichtlineare Optimierung mit Nebenbedingungen. Ein notwendiges Optimalitätskriterium erster Ordnung ist z.B. die Bedingung von Karush-Kuhn-Tucker (KKT) [56]. KKT-Punkte stellen mögliche Lösungen dar. Der Satz von Karush-Kuhn-Tucker besagt, dass, wenn der Punkt $\overrightarrow{x}^*$ das lokale Minimum des Optimierungsproblems darstellt und bestimmte Regularitätsbedingungen erfüllt, es Lagrange-Multiplikatoren $\overrightarrow{\lambda}^* \in \mathbb{R}^m$ und $\overrightarrow{\mu}^* \in \mathbb{R}^p$ gibt, für die folgende Gleichungen gelten. Bei diesem Punkt handelt es sich um einen KKT-Punkt [56].

$$\nabla f(\overrightarrow{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\overrightarrow{x}^*) + \sum_{j=1}^{p} \mu_j^* \nabla h_j(\overrightarrow{x}^*) = \overrightarrow{0} \tag{3.26}$$

$$g_i(\overrightarrow{x}^*) \leq 0, i = 1, 2, ...m \tag{3.27}$$

$$h_j(\overrightarrow{x}^*) = 0, i = 1, 2, ...p \tag{3.28}$$

$$\lambda_i^* \geq 0, i = 1, 2, ...m \tag{3.29}$$

$$\lambda_i^* g_i(\overrightarrow{x}^*) = 0, i = 1, 2, ...m \tag{3.30}$$

Dabei beschreibt h Gleichungen und g Ungleichungen für die zulässige Richtung. Lagrange-Multiplikatoren wurden bereits im Abschnitt 3.4.2 behandelt. Bei konvexen Problemen entspricht das notwendige Kriterium auch dem hinreichenden Kriterium.

Verfahren des Reduzierten Gradienten

Die Nebenbedingungen müssen in Form von Gleichungen vorliegen, um dieses Verfahren anwenden zu können. Aus diesen kann dann die Gütefunktion berechnet werden. Mit dieser Funktion und dem reduzierten Gradienten kann mit den Verfahren aus Kapitel 3.4.3 das Optimum gefunden werden [53].

3.4.5. Heuristische Optimierung

Zur heuristischen Optimierung gehören u.a. die naturanalogen Optimierungsverfahren. Sie orientieren sich stark an natürlich vorkommenden Verfahren und werden in den folgenden Kapiteln erläutert. Weitere Verfahren, die im Folgenden nicht aufgeführt werden, sind unter anderem die Schwarmintelligenten Algorithmen, die künstlichen Bienenvölker oder auch der Ameisenalgorithmus, die sich einer sehr großen Beliebtheit erfreuen [57].

Simulierte Abkühlung

Die Simulierte Abkühlung orientiert sich am Abkühlprozess von der Schmelze zum Festkörper, wobei die Schwelle die Temperatur darstellt, die nach Erreichen unter bestimmten Bedingungen wieder verlassen werden kann. Auf diese Weise kann das globale Optimum gefunden werden, indem lokale Optima übersprungen werden. Die Zielfunktion stellt das Energieniveau in Form von bewegten Molekülen dar. Die Methode versucht, ein thermisches Gleichgewicht innerhalb des Abkühlprozesses zu erhalten [51]. Ähnlich wie die Simulierte Abkühlung funktioniert auch die **Schwellenakzeptanzmethode**. Der Unterschied liegt in der Behandlung der Schwellwerte, da im Gegensatz zur Simulierten Abkühlung alle Verschlechterungen akzeptiert werden, solange sie einen Grenzwert nicht überschreiten [68]. In Abbildung 11 ist die Simulierte Abbildung veranschaulicht. Dabei werden die Punkte neu berechnet und lokale Minima können übersprungen werden.

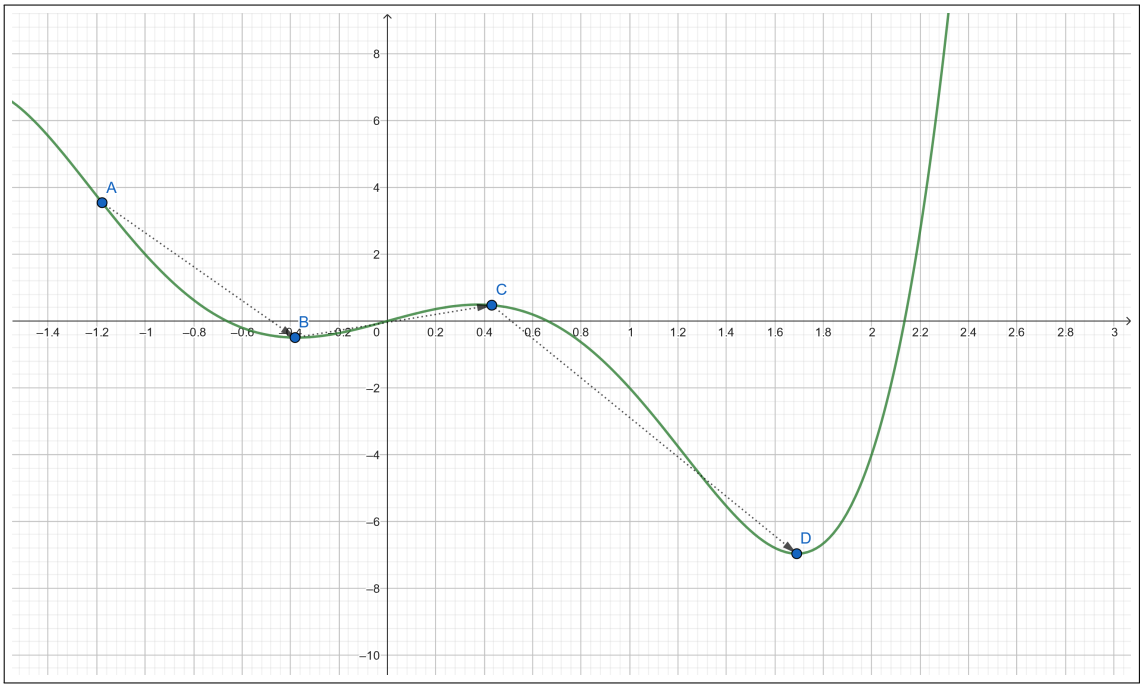


Abb. 11.: Veranschaulichung der Simulierten Abkühlung mit schrittweiser Neuberechnung der Punkte (blau) (Grafik inspiriert von Ghasemalizadeh et al. [23])

Bergsteigeralgorithmus

Der Bergsteigeralgorithmus ist ein recht triviales Verfahren, bei dem iterativ versucht wird, den nächstbesten Punkt zu finden. Auf diese Weise konvergieren die Punkte zu einem Optimum [68]. Die Situation ist vergleichbar mit einem Bergsteiger, der bei schlechter Sicht versucht, einen Berg zu erklimmen und sich daran orientiert möglichst steil bergauf zu gehen. Dieser Algorithmus ist nicht in der Lage lokale Minima zu übergehen [22].

Sintflutalgorithmus

Der Sintflutalgorithmus stellt eine vereinfachte Form der simulierten Abkühlung dar [51]. Beim Sintflutalgorithmus wird so lange nach einem neuen Punkt gesucht, bis der Schwellenwert erreicht ist. Mit jeder Akzeptanz eines neuen Punktes wird der Schwellenwert so lange erhöht, bis - wie beim Bergsteiger-Algorithmus - der Punkt gegen das Optimum konvergiert [68]. Um beim Beispiel des Bergsteigers zu bleiben, kann der Schwellenwert mit einer Überschwemmung verglichen werden, welcher der Bergsteiger entkommen möchte. Mit jedem Schritt, den er wählt, steigt der Wasserspiegel, so dass er gezwungen ist, einen Schritt nach oben in Richtung Berggipfel zu wählen. Im Gegensatz zum Bergsteigeralgorithmus können anfänglich lokale Minima überwunden werden. Das Verfahren ist veranschaulicht in Abbildung 12. Im 2-Dimensionalen

sieht es aus, als könnte das lokale Minimum nicht verlassen werden. Jedoch könnte es sich in einem mehrdimensionalen Raum auch um einen Sattelpunkt handeln [18].

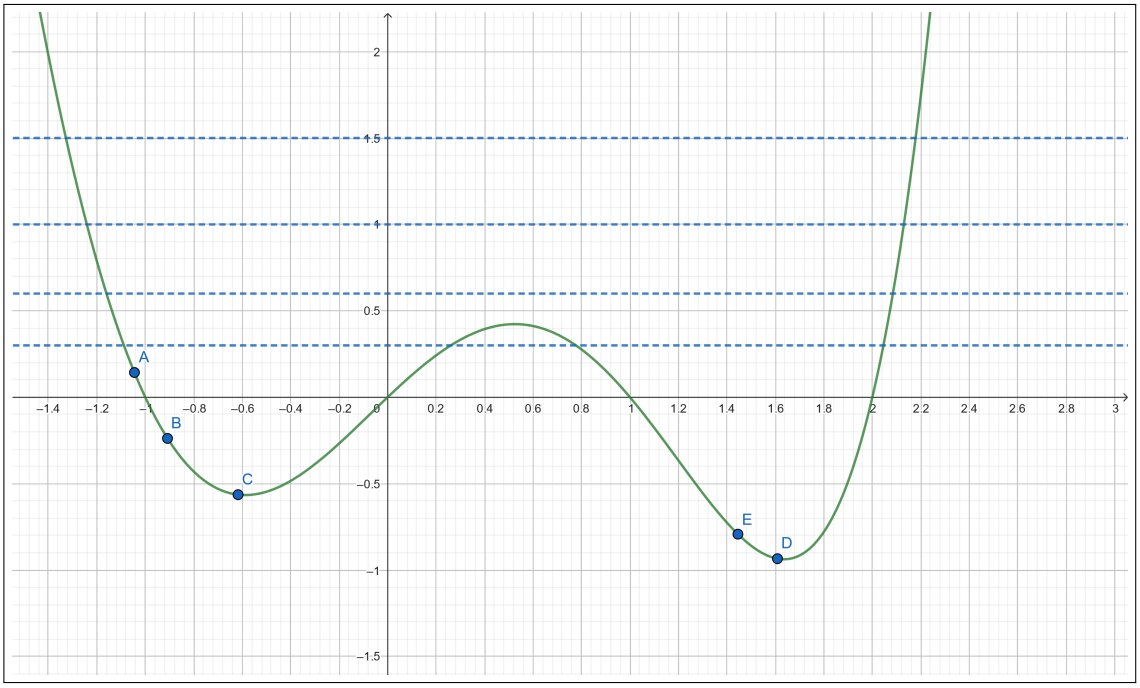


Abb. 12.: Veranschaulichung des Sintflutalgorithmus mit schrittweiser Neuberechnung der Punkte (blau). Dabei stellen die blauen Linien den steigenden Wasserspiegel dar. (Grafik inspiriert von Dittes [18])

Evolutionäre Algorithmen

Da sich evolutionäre Algorithmen am natürlichen Evolutionsprozess orientieren, gehören sie auch zu den naturanalogen Verfahren. Insbesondere wird versucht, die darwinistische Idee des Zusammenspiels von Variation und Selektion (englisch: *survival of the fittest*), die klassische Genetik sowie die moderne Populationsgenetik abzubilden [51, 69, S. 2].
Für globale Probleme sind sie eher ungeeignet, da keine Beweise abgeleitet werden können. Für hochdimensionale Suchräume sind diese Algorithmen jedoch besonders gut geeignet [51].
Der erste Schritt bei evolutionären Algorithmen ist die Initialisierung der Population (Menge an Individuen) und ihrer Ausprägungen. Danach erfolgt eine Bewertung jedes Individuums anhand des Fitnesswertes. Dieser Wert ordnet den Individuen eine Lösungsgüte zu. Folgende Schritte werden iterativ je Generation einer neuen Population durchgeführt bis ein Konvergenzkriterium erreicht wird. Zunächst findet die sogenannte stochastische Selektion statt, bei der alle Individuen gegeneinander antreten müssen und somit die mit den *besten* Fitnesswerten in den nächsten Schritt gelangen. Dann wird eine Replikation durchgeführt, bei der die Lösungsinformationen der Eltern kopiert werden. Eltern sind die zur Reproduktion ausgewählten Lösungen und die

Nachkommen sind aus den Eltern erzeugte Lösungen. Es findet eine Variation statt, welche entweder in Form einer Mutation oder eines Crossovers erfolgen kann. Beim Crossover werden Elemente der Individuen vermischt und bei die Mutation ist ein Suchoperator, bei der einzelne Individuen verändert werden [51]. Die Nachkommen werden bewertet und eine neue Population wird gebildet. [51]. In Abbildung 13 ist das Verfahren dargestellt [51].

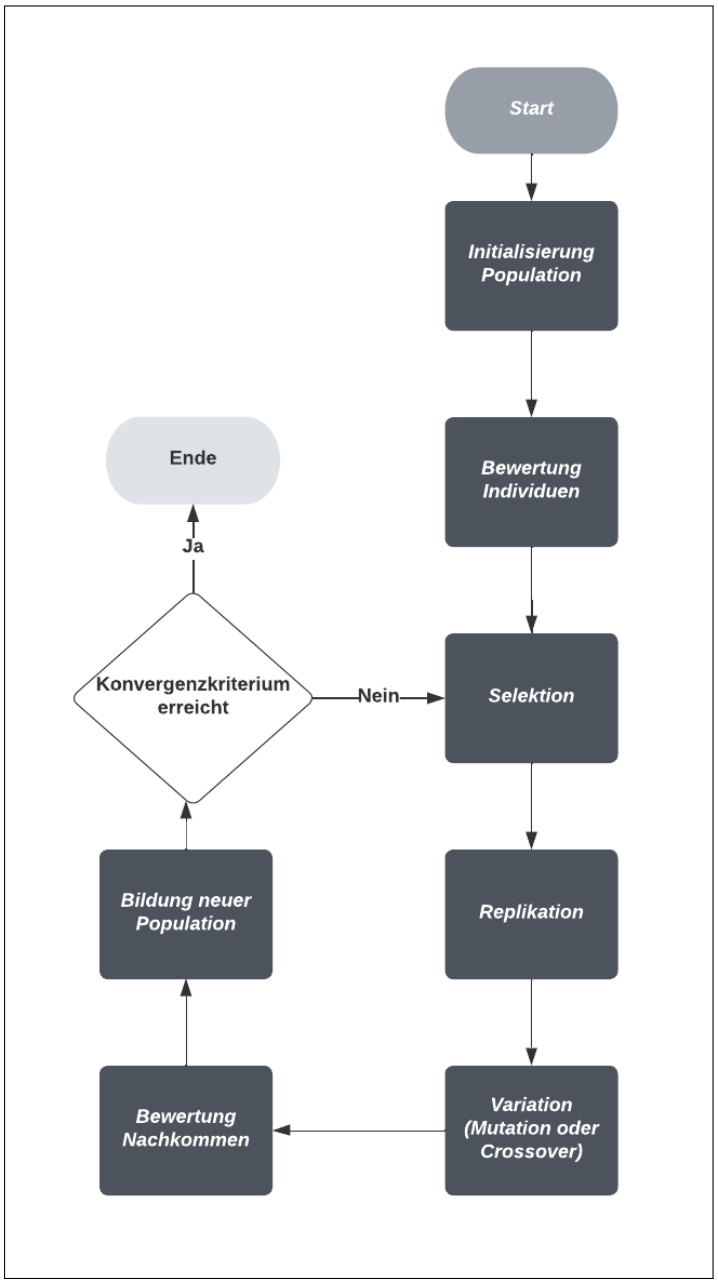


Abb. 13.: Flussdiagramm des Ablaufs eines evolutionären Algorithmus (Grafik inspiriert am Algorithmus von Nissen et al. [51])

Nissen et al. behandelten die Kombination von evolutionären Algorithmen mit Methoden des maschinellen Lernens oder auch Künstlichen Neuronalen Netzen. Evolutionäre Algorithmen können dabei sowohl zur Optimierung der Verbindungsgewichte als auch der Netzwerkarchitektur von KNNs oder von beidem gleichzeitig eingesetzt werden. Darüber hinaus können mit Hilfe von evolutionären Algorithmen auch die Trainingsdaten generiert oder die Parameter des Lernverfahrens von KNNs optimiert werden [51].

3.4.6. Multikriterielle Optimierung

Im Gegensatz zum unikriteriellen Optimierungsproblem werden bei multikriteriellen Optimierungsproblemen nicht nur ein, sondern mehrere Ausgaben in Form eines Vektors zurückgeliefert. Im Beispiel mit dem Ziel der Kreditwürdigkeitserlangung kann dies zusätzlich das Ziel einer hohen Tilgung sein. In diesem Fall können die Ziele übereinstimmen, in den meisten Fällen sind sie jedoch gegensätzlich. Im Gegensatz zu den bisher genannten Verfahren werden multikriterielle Probleme mathematisch und algorithmisch anders behandelt [14].
Mithilfe von bspw. der Gewichteten-Summen-Methode können diese multikriteriellen in unikriterielle Probleme überführt und anschließend mit den bereits genannten Methoden der nichtlinearen Optimierung gelöst werden. Darüberhinaus gibt es die Constraint-Methode, welche alle Ziele bis auf eins in geeignete Nebenbedingungen umwandelt, um diese dann ebenfalls mit den Methoden der nichtlinearen Optimierung lösen zu können. Weitere ähnliche Methoden sind die Benson- sowie die Kompromiss-Methode. Weitere Methoden werden auch von Marler et al. beschrieben [46, 69].
Im Folgenden werden sowohl die allgemeine Pareto-Optimierung als auch die Erweiterung der evolutionären multikriteriellen Optimierung erläutert.

3.4.7. Pareto-Optimierung

Die Pareto-Optimierung ist in folgenden Gleichungen definiert [62]. Dabei sind g(x), h(x) und i(x) Nebenbedingungen, die für die Optimierung erfüllt sein müssen.

$$\min_x \; J(x) := \min_x \; [J_1(x), J_2(x), ..., J_n(x)] \tag{3.31}$$

$$\text{u.d.N. } g(x) \leq 0 \tag{3.32}$$

$$\text{u.d.N. } h(x) \leq 0 \tag{3.33}$$

$$\text{u.d.N. } i(x) \leq 0 \tag{3.34}$$

g(x) aus Ungleichung 3.32 ist die Differenz zwischen den geänderten Variablen und der Vorgabe, wie viele Variablen maximal geändert werden dürfen. h(x) aus Ungleichung 3.33 ist die Differenz zwischen der Vorhersage und einer minimalen Zielklasse. Im Gegensatz zu h(x) ist i(x) die Differenz zwischen der Vorhersage und einer maximalen Zielklasse. Dies ist in Ungleichung 3.34 definiert. Bei diesen beiden Nebenbedingungen steht die Regressionsaufgabe im Vordergrund. Mit Hilfe einer Pareto-Front werden alle nicht dominierten bzw. dominierenden Lösungen innerhalb des zulässigen Lösungsraumes Z gefunden. Diese Lösungen werden auch als Pareto-optimale Lösungen bezeichnet.

Definition 7 (Nicht dominierte Lösungen) *Ein Vektor von Zielfunktionen, $J(x^*) \in Z$, ist dominant, wenn es keinen anderen Vektor $J(x) \in Z$ gibt, sodass $J(x) < J(x^*)$ mit mindestens einem $J_i(x) < J_i(x^*)$ [46].*

Da bei einem Durchlauf in der Regel nur ein Ergebnis ausgegeben wird, sind mehrere Durchläufe erforderlich, um die Pareto-Front zu erzeugen. In Abbildung 14 werden die dominierenden Lösungen, welche die Pareto-Front bilden, graphisch dargestellt.

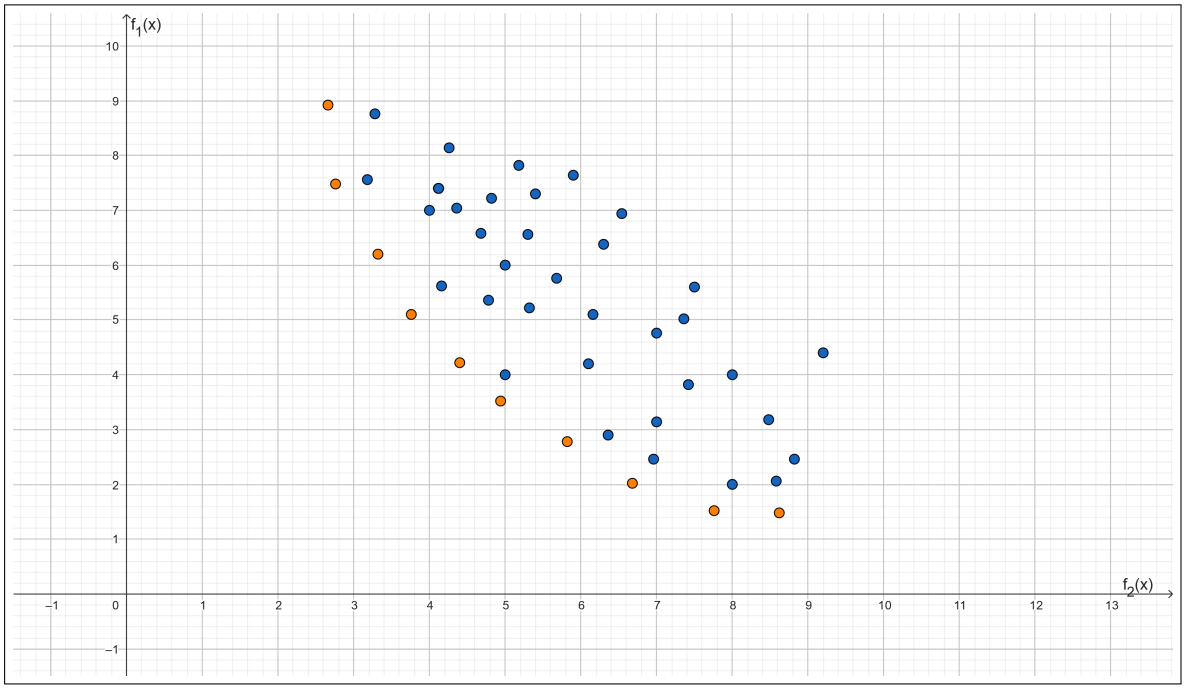


Abb. 14.: Pareto-Front mit dominierenden Lösungen (orange) und dominierten Lösungen (blau) (Grafik inspiriert von Reynoso et al. [57])

3.4.8. Evolutionäre Pareto-Optimierung

Evolutionäre Algorithmen sind bereits aus dem Kapitel 3.4.5 über die Einzeloptimierungsprobleme bekannt. Im Gegensatz zur einfachen Pareto-Optimierung findet der evolutionäre Al-

gorithmus in der Regel mehrere Ergebnisse innerhalb eines Durchlaufs [14]. Die evolutionäre Pareto-Optimierung basiert auf folgendem Verfahren. Es wird eine Anfangspopulation mit Individuen erzeugt. Anschließend wird die initiale Pareto-Menge approximiert. Solange das Abbruchkriterium nicht erreicht ist, wird die Population iterativ mit einem evolutionären Algorithmus aufgebaut und die neue Pareto-Menge mit Hilfe der neuen Population approximiert. Andernfalls wird die Approximation zurückgegeben [57].
Es gibt verschiedene Varianten des evolutionären Algorithmus, wie beispielsweise der Nichtdominierte Genetische Sortieralgorithmus (NSGA). Das NSGA-II-Verfahren ist eine Erweiterung des NSGA-Verfahrens. Die klassische Genetik wird in den **Genetischen Algorithmen** behandelt. Die beiden Hauptoperationen der Evolution sind Mutation (zufällige Veränderung) und Rekombination, um neue Phänotypen zu erzeugen [51, S. 238]. Die Attribute eines Individuums stellen dabei die Gene dar. Die Attribute werden so lange rekombiniert und mutiert, bis eine definierte Anzahl von Generationen (Verfahrensiterationen) erreicht ist oder sich die Leistung nicht mehr verbessert. Bei dem NSGA-II Verfahren werden zunächst alle Individuen einer Population nichtdominiert sortiert und ihnen somit ein Fitnesswert in Abhängigkeit von der Sortierung und der Populationsgröße zugewiesen. Dies wird solange wiederholt, bis alle Individuen klassifiziert sind. NSGA-II verwendet eine sogenannte Crowding-Distanz, die die Lösungsdichte von einen Punkt den durchschnittlichen Abstand zu den anderen Punkten berechnet. Bei der Auswahl der Punkte werden sowohl die Klassifikation als auch die Distanz zur Entscheidung herangezogen [14, 16]. In Abbildung 15 ist das Verfahren dargestellt [52].

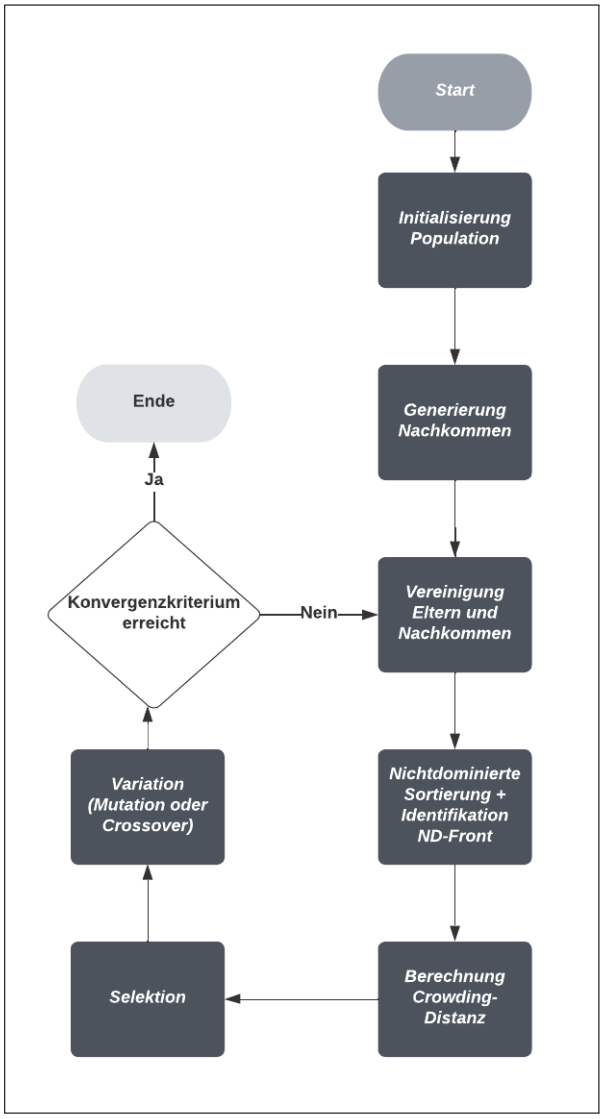


Abb. 15.: Flussdiagramm vom Ablauf des NSGA-II Algorithmus (vereinfachte Form) (Grafik inspiriert am Algorithmus von Nissen et al. [51])

Im Gegensatz zum NSGA-II-Verfahren wird beim NSGA-III-Verfahren versucht, eine höhere Lösungsvielfalt zu erreichen. Im RNSGA-II Ansatz werden vordefinierte Referenzpunkte mithilfe der Euklidischen Distanz, welche in Gleichung 4.5 definiert ist, berechnet. Diese werden verwendet, um die Pareto-Front zu unterteilen, da nicht jedes Individuum der Front überlebt [41]. UNSGA-III versucht wiederum eine bessere Performance zu erreichen, indem ein anderer Selektionsalgorithmus, der Tournament-Algorithmus, verwendet wird [62, 4]. Darüberhinaus gibt es noch viele weitere Verfahren welche innerhalb dieser Arbeit nicht weiter behandelt werden [14]. Grafik 16 veranschaulicht das Schemata von NSGA-II, wobei P_t die Eltern und Q_t die Nachkommen darstellt. Die generierten Fronten sind mit F_i bezeichnet und P_{t+1} stellt die neue Population dar [16].

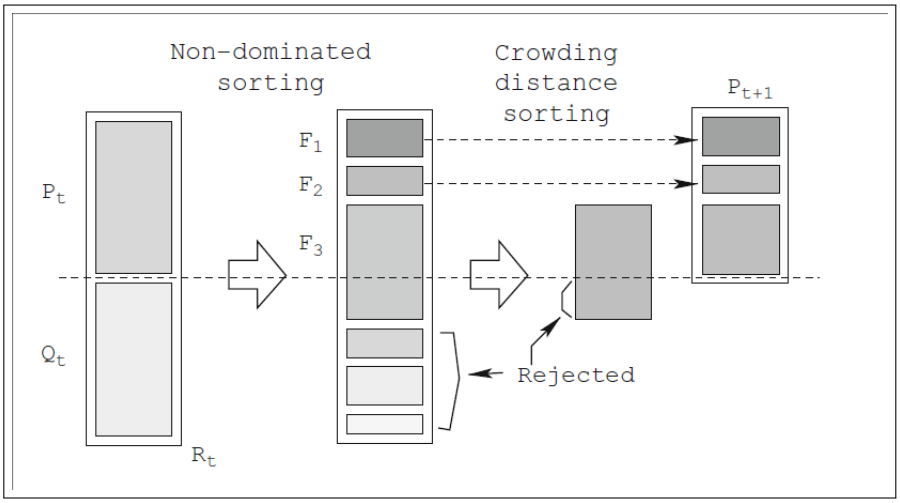


Abb. 16.: Konzept von NSGA-II (Quelle der Grafik von Deb et al. [16])

Vergleich der Optimierungsverfahren

Um einen besseren Überblick über die einzelnen genannten Verfahren zu geben, sind diese in Tabelle 1 noch einmal mit ihren wichtigsten Merkmalen aufgelistet. Handelt es sich um ein globales Optimum, so handelt es sich auch implizit um ein lokales Optimum.

Methode	global/ lokal	Funktionstyp	Neben- bedingungen	Problem	Kernidee
Simplex-Methode	global	linear	mit	unikriteriell	Abtastung der Ecken eines Polyeders
Innere Punkte Verfahren	global	linear/ nichtlinear	mit	unikriteriell	Abtastung durch das Polyeder selbst
Gradienten- verfahren	lokal	nichtlinear	ohne	unikriteriell	Mithilfe des Gradienten die Richtung des steilsten Anstieges finden
Downhill- Simplex- Verfahren	lokal	nichtlinear	ohne	unikriteriell	Eckpunkte des Simplex neu berechnen mit Refklexion, Expansion, Kontraktion und Multikontraktion
Goldener Schnitt	global	nichtlinear (unimodal)	ohne	unikriteriell	Intervallverkleinerung
Quasi-Newton- Verfahren	lokal	nichtlinear	ohne	unikriteriell	Suchrichtung mithilfe der Approximation der Hesseschen Matrix berechnen
Trust-Region- Verfahren	lokal	nichtlinear	ohne	unikriteriell	Quadratische Approximation mit Verkleinerung eines Vertrauensbereiches
Subgradienten- verfahren	global	konvex	ohne	unikriteriell	Mithilfe des Subgradienten die Suchrichtung bestimmen
Newton- Verfahren	lokal	nichtlinear	ohne	unikriteriell	mithilfe der Newton-Iteration solange neue Punkte berechnen, bis Folge konvergiert
Reduzierter Gradient	lokal	nichtlinear	mit	unikriteriell	Umwandlung der Nebenbedingungen zu Gleichungen + Gradientenverfahren

Simulierte Abkühlung	global	heuristisch	ohne	unikriteriell	Algorithmus orientiert an physikalischem und chemischen Verfahren der Abkühlung von Metallen
Bergsteiger-algorithmus	lokal	heuristisch	ohne	unikriteriell	Algorithmus orientiert an einem Bergsteiger, der Berg möglichst steil erklimmt
Sintflut-algorithmus	global	heuristisch	ohne	unikriteriell	Kombination aus Sintflut- und Bergsteiger-algorithmus
Pareto-Optimierung	global nah	heuristisch	mit	multikriteriell	mehrere Ziele mithilfe von Pareto-Front finden
Evolutionäre Pareto-Optimierung	global nah	heuristisch	mit	multikriteriell	Pareto-Optimierung in Kombination mit evolutionären Algorithmen

Tab. 1.: Übersicht verschiedener Optimierungsverfahren

3.5. Optimierung von Entscheidungsbäumen

Da neben den bereits erwähnten Entscheidungsbaumalgorithmen optimierte Verfahren in der Anwendung verwendet werden sollen, werden diese in diesem Kapitel behandelt.
Es gibt verschiedene Möglichkeiten, Optimierungen an Entscheidungsbaummodellen vorzunehmen. Neben allgemeinen Optimierungen an bestehenden Modellalgorithmen, die im Kapitel 3.5.3 beschrieben werden, können die Eingabeparameter der Algorithmen mit Hilfe der Hyperparameteroptimierung angepasst werden. Ein weiteres bekanntes Verfahren ist das Pruning, welches im folgenden Kapitel erläutert wird.

3.5.1. Pruning von Entscheidungsbäumen

Pruning gehört streng genommen nicht zu den Optimierungsmethoden, wird aber dennoch aufgeführt, weil damit eine Verbesserung der Modelle erreicht werden kann. Da Entscheidungsbäume zur Überanpassung neigen, wird Pruning eingesetzt, um die Überanpassung und damit die Fehlerrate bzw. den Verlust zu reduzieren. Weitere Gründe sind eine bessere Interpretierbarkeit sowie eine visuell kompaktere Darstellung der Entscheidungsbäume. In Abbildung 17 ist ein geprunter Entscheidungsbaum dargestellt.

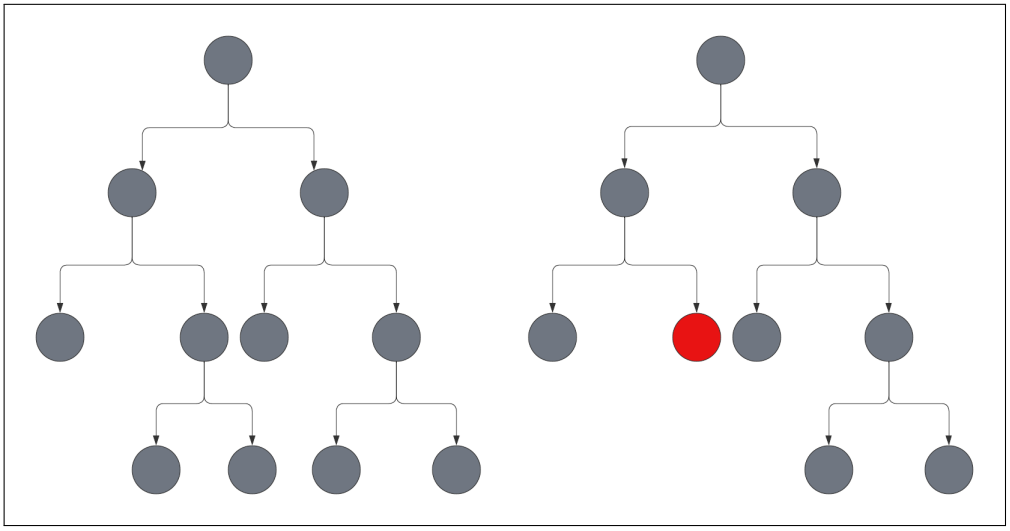


Abb. 17.: Links der originale und rechts der geprunte Entscheidungsbaum (Grafik inspiriert von Merenda et al. [47])

Es gibt viele verschiedene Methoden, die für unterschiedliche Modelle eingesetzt werden können. Generell lassen sie sich in Pre-Pruning- und Post-Pruning-Methoden unterteilen. Post-Pruning wird eingesetzt, nachdem das Modell erstellt wurde. Im Gegensatz dazu wird Pre-Pruning vor der Modellerstellung angewendet. In den folgenden Kapiteln werden einige bekannte Methoden erläutert [6].

Kostenkomplexitäts-Pruning

Kostenkomplexitäts-Pruning wird z.B. bei CART-Modellen angewendet. Dabei spielt der Parameter cpp_alpha eine wichtige Rolle. Je höher dieser Wert ist, desto geringer ist die Tiefe des Baumes und desto mehr neigt das Modell zur Überanpassung. Es wird daher versucht, den Parameter so einzustellen, dass die Genauigkeit der Testdaten möglichst hoch ist und somit ein guter Mittelweg zwischen Unter- und Überanpassung gefunden wird. Da mehrere Modelle mit unterschiedlichen Parametern erzeugt werden müssen, gehört diese Methode zu den Post-Pruning-Methoden.

Vorzeitiges Stoppen

Beim vorzeitigen Stopp wird während der Generierung des Modells die Weiterführung von einem Knoten zu weiteren Kindknoten gestoppt, wenn der Fehler bei der Kreuzvalidierung mit einem neuen Knoten nicht in ausreichendem Maße reduziert werden kann.

3.5.2. Hyperparameter-Optimierung

Die Daten, die in Modellen des maschinellen Lernens verwendet werden, können in zwei Arten unterteilt werden. Zum einen gibt es die Modellparameter, die während der Modellentwicklung erstellt werden. Dazu gehören zum Beispiel Gewichtungen. Zum anderen gibt es Hyperparameter, die vor der Modellentwicklung festgelegt werden können. Dazu gehören bei Entscheidungsbäumen und dem CART-Algorithmus bspw. die maximale Tiefe des Baumes oder die minimale Anzahl an Instanzen eines Knotens, um einen Knoten weiter unterteilen zu können. Die Hyperparameter variieren je nach verwendetem Algorithmus. Diese Art der Optimierung versucht die besten Parameter für das Modell und den übergebenen Datensatz zu finden.

Definition 8 (Hyperparameter-Tuning) *Hyperparameter-Tuning ist eine Form von Black-Box Optimierung, bei der versucht wird die optimalen Hyperparameter für einen Datensatz und Algorithmus zu finden und wird in Gleichung 3.35 definiert. H ist eine Menge an Hyperparameter-Konfigurationen, D ist der Datensatz und $a \in A$ ist ein Algorithmus [43].*

$$h* = \underset{h \in H}{argmax} f(a, D, h) \tag{3.35}$$

Neben den folgenden Methoden gibt es noch weitere, wie z.B. die Bayes'sche Optimierung, die jedoch für Black-Box Modelle verwendet wird [43].

Grid-Search

Bei Grid-Search wird die Kreuzvalidierung verwendet, um die optimale Kombination von Hyperparametern zu finden. Dabei werden die zu verändernden Parameter und die möglichen Optionen vorgegeben und alle Kombinationen getestet. Bei n verschiedenen Parametern mit einer Auswahl von jeweils m Werten gibt es also $n * m$ Kombinationen, die getestet werden [37].

Random-Search

Im Gegensatz zur Grid-Search werden bei der Random-Search zufällige Kombinationen getestet, was zwar die Performance deutlich erhöht, aber unter Umständen nicht das optimale Ergebnis liefert. Um dieses Problem zu umgehen, ist auch eine Kombination aus Vorauswahl durch Random-Search und Verfeinerung durch Grid-Search möglich und sinnvoll [37].

3.5.3. Optimierung von Modell-Algorithmen

Nach der Erläuterung allgemeiner Optimierungsverfahren werden in diesem Kapitel bekannte Optimierungsverfahren an Modellalgorithmen von Entscheidungsbäumen vorgestellt.

Alternierende Baum-Optimierung

Bei der iterativen Baumoptimierung (englisch: *Tree alternating optimization* (TAO)) wird ein Bottom-to-Top-Verfahren verwendet, bei dem für alle Blätter in aufsteigender Reihenfolge bis zu einer bestimmten Tiefe eine parallele Optimierung durchgeführt wird. Wenn ein Knoten keine Trainingspunkte erhält, wird dieser Knoten eliminiert. Dabei sind Trainingspunkte Instanzen der Trainingsmenge, welche zu Knoten zugeordet werden. Da der Baum nur kleiner werden kann, wird dieses Verfahren auch indirektes Pruning genannt. Dabei wird jeder Knoten separat mit einer Verlustfunktion optimiert. Dieses Verfahren verbessert die Qualität des Modells erheblich [8].

Hierarchische Schrumpfung

Im Gegensatz zur alternierenden Baum-Optimierung wird bei der Hierarchischen Schrumpfung (englisch: *Hierarchichal shrinkage* (HS)) die Baumstruktur nicht verändert. Es wird lediglich ein Regularisierungsmechanismus auf die Knoten angewendet, bei dem die Vorhersage auf den Mittelwert der Elternknoten reduziert wird. Insbesondere die Performance wird durch diese Methode verbessert [1].

Verallgemeinerte Optimale Spärliche Entscheidungsbäume

Die Verallgemeinerte Optimale Spärliche Entscheidungsbäume (englisch: *Generalized optimal sparse decisiontrees* (GOSDT)) konzentrieren sich auf zwei zentrale Punkte: die Behandlung unbalancierter Daten und die vollständige Optimierung mit kontinuierlichen Attributen. Dies wird durch die Optimierung mehrerer Ziele erreicht. Zu diesen Zielen gehören z.B. der Anteil korrekter Vorhersagen (Genauigkeit), der Mittelwert aus True-Positive- und True-Negativ-Raten (balancierte Genauigkeit), die gewichtete Genauigkeit mit einem Gewichtungswert sowie ein harmonischer Mittelwert aus Genauigkeit und Trefferquote (F-Score). Weitere Zielgrößen lassen sich aus den ROC-Kurven und der Fläche unter der Kurve (AUC) ableiten. Aus einer ROC-Kurve lassen sich die Falsch-positive Rate (Spezifizität) und die Wahr-positive Rate (Sensitivität) mithilfe von Testdaten des Modells ableiten. Abbildung 18 präsentiert eine solche ROC-Kurve.

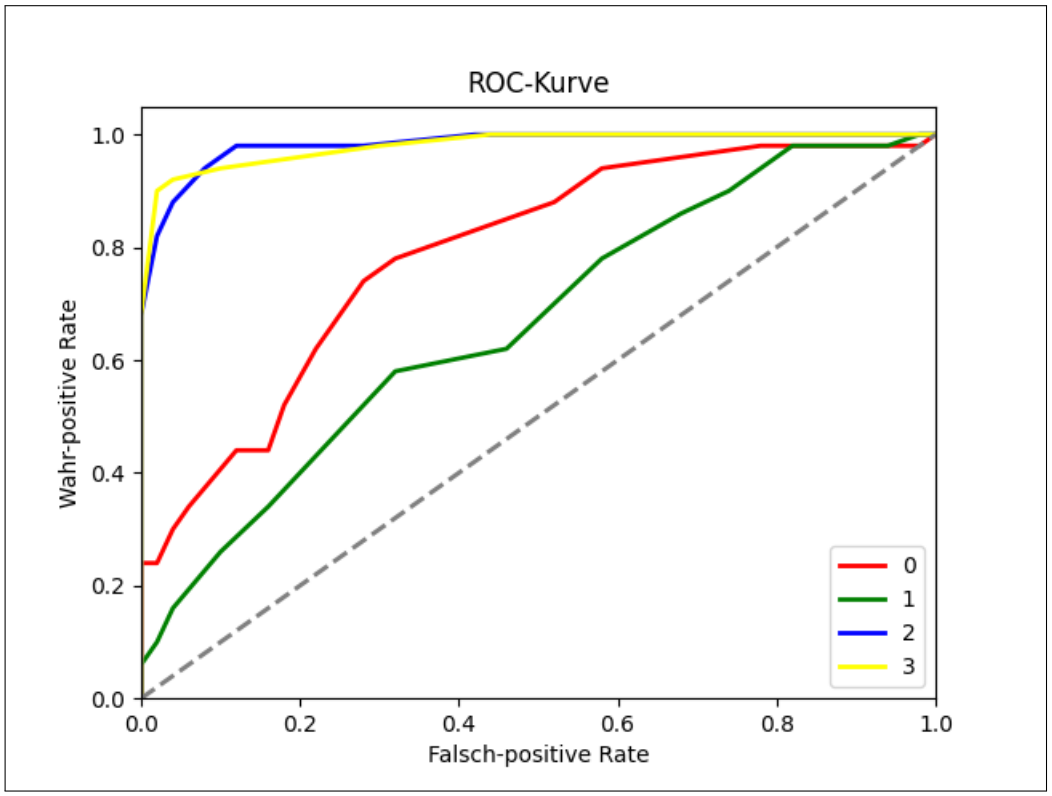


Abb. 18.: Vier verschiedene ROC-Kurven in rot, grün, blau und gelb. Dabei stellen die blaue und gelbe Linien die besten Lösungen dar.

Das Ganze basiert auf der Verlustfunktion, die aus den oben genannten Zielen abgeleitet werden kann, und einer Spärlichkeits-Regulierung, bei der die Anzahl der Blätter benachteiligt wird. Der Entscheidungsbaum wird mit Hilfe eines Branch-and-Bound-Ansatzes modifiziert [38].

4. Entwicklung eines interaktiven Werkzeuges zur Kontrafaktischen Analyse

Mit dem Hintergrundwissen aus den vorangegangenen Kapiteln werde in diesem Kapitel allgemeine Anforderungen an die Anwendung vorgestellt, die Nutzer definiert und nach der Definition der Anforderungen und Akzeptanzkriterien die Strategieentwicklung präsentiert.

4.1. Allgemeine Anforderungen an die Anwendung

Grundlegendes Ziel ist es, kontrafaktische Analysen in Entscheidungsbäumen durchführen zu können und Nebenbedingungen für Lösungen innerhalb des System berechnen zu können. Wie bereits im Kapitel 3.4 dargestellt, gibt es eine Reihe von Optimierungsverfahren. Ein Schritt ist die Findung eines geeingeten Verfahrens für die kontrafaktische Generierung. Ebenso müssen Verfahren zur Generierung von Entscheidungsbäumen ermittelt werden, die der Anwender bestenfalls auch vergleichen kann, um einen für seinen Anwendungsfall geeigneten Algorithmus auswählen zu können. Wie bereits im Kapitel 3.5.1 vorgestellt, gibt es die Möglichkeit des Prunings. Für verschiedene Algorithmen müssen geeignete Pruning-Verfahren ausgewählt werden können.
Darüber hinaus müssen die Berechnungen für den Anwender nachvollziehbar sein. Durch die natürliche Struktur der Entscheidungsbäume kann die Nachvollziehbarkeit über eine Benutzeroberfläche hergestellt werden. Besonders wichtig ist hier der Fokus auf eine gute Usability, um als Anwender möglichst intuitiv zum Ziel zu kommen, sowie auf eine gute User Experience, um auch einen positiven Eindruck von der Anwendung selbst zu erhalten. Wie bereits erwähnt, ist auch das Vertrauen in die Anwendung ein Indikator für eine bessere Erklärbarkeit der Anwendung.

4.2. Nutzer der Anwendung

Um die Anforderungen definieren zu können und den Bedürfnissen des Nutzers gerecht zu werden, muss der Nutzer der Anwendung definiert werden. Bei den Anwendern handelt es sich um Personen, die sich in ihrem Fachgebiet (Bankenwesen, Gesundheitswesen usw.) sehr gut auskennen, aber nicht unbedingt über Fachwissen im Bereich der kontrafaktischen Analyse verfügen. Sie müssen keine Experten in den zugrundeliegenden Algorithmen sein, obwohl dies hilfreich sein kann. Wichtig ist, dass sie in der Lage sind, die Ergebnisse der kontrafaktischen Analyse auszuwerten und zu interpretieren. Aus diesem Grund muss die Anwendung eine einfache Grundeinstellung ohne Eingabe von Expertenparametern bieten, diese aber um solche Parameter erweitern können. Der Anwender sollte sich die Inhalte der verschiedenen Ansichten nicht merken müssen. Wichtig ist eine übersichtliche Darstellung der Klassifikation und der Kontrafaktuale in der Visualisierung, damit der Anwender die Lösungen besser verstehen und vergleichen kann.

4.3. Spezifizieren der Anforderungen an die Anwendung

Aus der allgemeinen Anforderungsbeschreiben lassen sich genau vier Aufgaben an die Anwendung ableiten, die in weitere Teilaufgaben untergliedert werden können. Diese werden im Folgenden aufgeführt.

4.3.1. Anforderungen an die Generierung von Kontrafaktualen

Die Anwendung soll eine Funktion beinhalten, die bei Eingabe eines trainierten Modells und geltender Nebenbedingungen mindestens ein passendes Kontrafaktual zurückgibt, sofern dies realistisch ist. Dabei ist neben der Klassifikationsaufgabe auch die Regression erwünscht, aber nicht notwendig. Die Angabe folgender Nebenbedingungen soll realisiert werden.

Minimale Anzahl an Attributen

Eine Nebenbedingung ist die Berechnung der minimalen Lösung. Bei der Minimallösung darf sich für eine Instanz nur eine minimale Anzahl von Attributen ändern, um das gewünschte Ergebnis zu erzielen. Idealerweise sollte auch die Änderung des Attributs so gering wie möglich sein.

Toleranzbereich der Attribute

Es muss möglich sein, für numerische Attribute eine Änderungsdistanz anzugeben. Die Differenz der Änderung darf einen definierten Wert nicht überschreiten. Eine weitere Möglichkeit wäre die Angabe eines Intervalls, in dem das Ergebnis für ein Attribut liegen darf.

Unveränderliche Attribute

Ferner muss es möglich sein, Attribute von einer Veränderung auszuschließen. Diese sollen bei der kontrafaktischen Generierung nicht verändert werden.

4.3.2. Anforderungen an die Generierung von Entscheidungsbäumen

Um trainierte Modelle für die kontrafaktische Generierung übergeben zu können, muss die Anwendung eine Funktion zur Verfügung stellen, mit der ein Modell auf Basis eines ausgewählten Algorithmus mit einer ausgewählten Pruning-Methode sowie der Größe der Trainings- und Testdaten erstellt werden kann. Der Benutzer sollte zwischen Algorithmen wie CART, ID3 oder C4.5 wählen können. Optimierte Algorithmen und zu den einzelnen Algorithmen passende Pruning-Methoden sind erwünscht.

4.3.3. Anforderungen an das Erstellen einer Graphischen Oberfläche

Wie bereits erwähnt, spielt die Visualisierung eine wichtige Rolle. Aus diesem Grund muss der Anwender in der Lage sein, die Funktionen über eine Benutzeroberfläche zu bedienen. Die Visualisierung des Entscheidungsbaumes ist dabei von entscheidender Bedeutung. Die einzelnen Instanzen und Kontrafakturen sind im Baum farblich zu kennzeichnen. So können die Pfade identifiziert werden. Darüber hinaus ist die Möglichkeit vorgesehen, dass der Benutzer neue Vektoren eingeben kann, um das trainierte Modell Vorhersagen zu treffen. Eine weitere wichtige Funktion ist die Angabe von Nebenbedingungen für die kontrafaktische Generierung. Hilfreiche Elemente wie die Anpassung des Erscheinungsbildes des Entscheidungsbaumes (Rotation und Skalierung) sind erwünscht. Da die Größe der Entscheidungsbäume stark variieren kann, muss es eine Möglichkeit geben, die Darstellung anzupassen, zu zoomen und zu verschieben.

4.3.4. Anforderungen an das Ermöglichen der Datenspeicherung

Es sind Funktionen zum Speichern und Lesen der Modelle vorgesehen, um eine einfache Wiederverwendung zu ermöglichen. Neue Datensätze müssen importiert und die Entscheidungsbäume konkreter Datensätze und Modelle gespeichert und gelesen werden können.

4.4. Akzeptanzkriterien der Anwendung

Im Folgenden werden zu den Anforderungen entsprechende Akzeptanzkriterien vorgestellt.

1. Als Anwender möchte ich Datensätze in Form einer csv-Datei einlesen können.
2. Als Anwender möchte ich eine Visualisierung meiner Daten in Form eines Entscheidungsbaumes erhalten.
3. Als Anwender möchte ich verschiedene Algorithmen und Pruning-Verfahren auf meinen Datensatz vergleichen können.
4. Als Anwender möchte ich die Modellkonfigurationen speichern und wieder einlesen können.
5. Als Anwender möchte ich eine individuelle Instanz klassifizieren können.
6. Als Anwender möchte ich Nebenbedingungen und ein Ziel zur kontrafaktischen Analyse eingeben können.
7. Als Anwender möchte ich ein geeignetes Kontrafaktual finden können, wenn möglich.

4.5. Strategieentwicklung der Anwendung

Aus den Anwendungsanforderungen wird eine Umsetzungsstrategie abgeleitet. Die Generierung von Kontrafaktualen kann als Optimierungsproblem dargestellt werden. Da im Kapitel 3.4 einige Optimierungsverfahren vorgestellt wurden, stellt sich unter anderem die Frage, welche davon für die genannten Anforderungen in Frage kommen. Dabei wird auf bekanntes Wissen zurückgegriffen.

4.5.1. Verwandte Lösungen

Da sich diese Arbeit auf die Verwendung von Entscheidungsbäumen und somit nichtlineare Klassifizierer bzw. Regressoren beschränkt, handelt es sich mathematisch um ein nicht-konvexes und nicht-differenzierbares Problem. Bei der Betrachtung der Einzeloptimierungsprobleme werden die Grenzen der kontrafaktischen Generierung deutlich. Möglichkeiten wären z.B. die Reduktion der Differenz zwischen Ausgangs- und Zielwert **oder** zwischen Eingabeparametern und Parametern des Kontrafaktualen. Um beides abzudecken, ist die Anwendung der multikriteriellen Optimierung unumgänglich [62].

Grundlage einiger Methoden ist die von Wachter et al., die in Kapitel 4.5.1 näher erläutert wird. Eine weitere Methode ist die von Dandl et al. in Kapitel 4.5.1. Darauf aufbauend wird der Unterschied zur Methode von Monteiro et al. erläutert.
Außerdem besteht, wie bereits erwähnt, die Möglichkeit, das multikriterielle Problem in ein unikriterielles Problem zu transformieren. Dandl und Marler et al. betonen jedoch, dass diese Lösungen die Wichtigkeit der einzelnen Ziele nicht balancieren können. Aus diesem Grund wird sich auf die folgenden bewährten Lösungen eingeschränkt [15, 45].

Kontrafaktische Generierung nach Wachter et al

Wachter et al. lösen ein Optimierungsproblem durch Minimierung der Verlustfunktion wie in Gleichung 4.1. Dies wird mit dem Downhill-Simplex-Verfahren erreicht, welches in Kapitel 3.4.3 erläutert wurde. Dabei beschreibt x die Instanz und $\hat{f}(x)$ die zugehörige Vorhersage, y' die Zielklasse und x' das Kontrafaktual.

$$\underset{x'}{argmin}\underset{\lambda}{max}L(x, x', y', \lambda) \tag{4.1}$$

Die Verlustfunktion ist in Gleichung 4.2 definiert.

$$L(x, x', y', \lambda) = \lambda * (\hat{f}(x') - y')^2 + d(x, x') \tag{4.2}$$

Dabei ist d die Manhattan-Distanz gewichtet mit der inversen mittleren absoluten Abweichung vom Median zwischen Originalinstanz und Kontrafaktual. Sie ist in Gleichung 4.6 definiert. λ gewichtet den ersten Term gegen die Distanzmetrik aus.

$$d(x, x') = \sum_{j=1}^{p} \frac{|x_j - x'_j|}{MAD_j} \tag{4.3}$$

Definition 9 (Manhattan-Distanz) *Die Manhattan-Distanz oder L1-Norm wird zur Bestimmung des Abstands zwischen Vektoren verwendet und ist in Gleichung 4.4 definiert [35].*

$$||v||_1 = \sum_{i=1}^{n} |v_i| \tag{4.4}$$

Anstelle der Manhattan-Distanz kann auch die Euklidische Distanz verwendet werden.

Definition 10 (Euklidische Distanz) *Die Euklidische Distanz, auch L2-Norm genannt, ist in Gleichung 4.5 definiert [35].*

$$||v||_2 = [\sum_{i=1}^{n} |\overrightarrow{v_i}|^2]^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{n} |\overrightarrow{v_i}|^2} \tag{4.5}$$

Definition 11 (Mittlere absolute Abweichung vom Median (MAD)) *Die mittlere absolute Abweichung vom Median ist in Gleichung 4.6 definiert.*

$$MAD_j = median_{i \in 1,\ldots,n}(|x_{i,j} - median_{l \in 1,\ldots,n}(x_l, j)|) \tag{4.6}$$

Da auch dieses Verfahren Defizite aufweist, wird im Folgenden das Verfahren von Dandl et al. vorgestellt, das diese Probleme behebt. Zu den Defiziten gehört z.B., dass kategoriale Attribute nicht behandelt werden können [15].

Kontrafaktische Generierung nach Dandl et al

Im Gegensatz zu dem Verfahren von Wachter et al. versuchen Dandl et al. vier verschiedene Ziele mit Hilfe einer einzigen Funktion zu minimieren. Die Funktion ist in Gleichung 4.7 definiert. X^{obs} sind dabei die beobachteten Trainingsdaten.

$$L(x, x', y', X^{obs}) = (\sigma_1(\hat{f}(x'), y'), \sigma_2(x, x'), \sigma_3(x, x'), \sigma_4(x', X^{obs})) \tag{4.7}$$

Das erste Ziel besteht darin, die Abweichung der Zielklasse von der Vorhersage wie in Gleichung 4.8 zu minimieren.

$$\sigma_1(\hat{f}(x'), y') = \begin{cases} 0 & \text{wenn } \hat{f}(x') \in y' \\ \underset{y' \in y'}{inf} |f(x') - y'| & \text{sonst} \end{cases} \tag{4.8}$$

Darüber hinaus soll der Abstand zwischen den Werten der Originalinstanz und der Kontrafaktuale wie in Gleichung 4.9 verringert werden. Dies kann mit Hilfe der Gower-Distanz erreicht werden, welche in Gleichung 4.10 definiert ist.

$$\sigma_2(x, x') = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x'_j) \in [0, 1] \tag{4.9}$$

In der folgenden Funktion muss zwischen kategorialen und numerischen Attributen unterschieden werden. Dabei ist $\hat{R}_2$ der Wertebereich des Attributs und $\mathbb{I}_{x_j} \neq x'_j$ die Anzahl geänderter Attribute.

$$\delta_G(x, x') = \begin{cases} \frac{1}{\hat{R}_2} |x_j - x'_j| & \text{wenn } x_j \text{ numerisch ist} \\ \mathbb{I}_{x_j} \neq x'_j & \text{wenn } x_j \text{ kategorial ist} \end{cases} \tag{4.10}$$

Anschließend wird versucht, die Anzahl der geänderten Attribute zu reduzieren. Dies wird mithilfe der Gleichung 4.11 durchgeführt.

$$\sigma_3(x, x') = ||x - x'||_0 = \sum_{j=1}^{p} \mathbb{I}_{x_j \neq x'_j} \tag{4.11}$$

Zu guter Letzt wird versucht die gewichtete Gower-Distanz zu reduzieren. Dies wird in Gleichung 4.12 definiert und dabei ist $w^{[i]}$ die Gewichtung.

$$\sigma_4(x, X^{obs}) = \sum_{i=1}^{k} w^{[i]} \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x_j^{[i]}) \in [0, 1], wobei \sum_{i=1}^{k} w^{[i]} = 1 \tag{4.12}$$

Als Optimierungsverfahren wird das NSGA-II verwendet, das jedoch speziell an die Problemstellung angepasst wurde, indem z.B. eine andere Crowding-Distanz verwendet wird.

Kontrafaktische Generierung nach Monteiro et al.

Monteiro et al. haben eine eigene Lösung realisiert, bei der die folgenden drei Ziele in Anlehnung an das Verfahren von Dandl et al. angelehnt optimiert werden sollen.

1. Differenz zwischen dem Ausgabe- und dem Zielwert

2. Gower-Distanz zwischen der Original-Instanz und dem Kontrafaktual
3. Anzahl geänderter Variablen

Zusätzlich werden in dieser Lösung folgende Nebenbedingungen für die Zielfunktion definiert.

$$g(x) \leq 0 \tag{4.13}$$

$$h(x) \leq 0 \tag{4.14}$$

$$i(x) \leq 0 \tag{4.15}$$

$g(x)$ definiert eine Grenze für die Anzahl der Variablen, die verändert werden dürfen. $h(x)$ und $i(x)$ beschreiben die Ober- bzw. Untergrenze des Zielwertes bei Regressionsaufgaben. Bei Klassifikationsaufgaben entsprechen sie einem vorgegebenen Wert.
Der gesamte Verarbeitungsprozess wird auch als Mehrzieloptimierung bezeichnet. Es besteht aus genau drei Schritten. Dazu gehören die Zieldefinition, die Suche und der Entscheidungsprozess. Eine solche Strategie heißt Generate First Choose Later (GFCL) [57].
Im Gegensatz zu Dandl et al. sind vier verschiedene Methoden als Optimierungsverfahren im Einsatz. Diese sind NSGA-II, NSGA-III, RNSGA-II und UNSGA-II. Für jeden Algorithmus wird eine Paretofront erzeugt, ein Dominanzfilter eingesetzt und eine gemeinsame Menge gebildet. Wenn dies für alle Algorithmen geschehen ist, wird der Dominanzfilter ein letztes Mal auf alle Werte angewendet. Neben dem Dominanzfilter ist noch ein weiteres Verfahren zur Filterung der Kontrafaktuale in Verwendung. Dabei wird eine Methode namens TOPSIS (Technique for the Order of Preference based on Similarity to the Ideal Solution) genutzt, um die Lösungen zu ordnen und die n besten Lösungen zurückzugeben.

4.5.2. Strategieentwicklung auf Basis verwandter Arbeiten

Im folgenden Abschnitt wird die Entwicklungsstrategie erläutert, wobei zunächst auf die Generierung von Entscheidungsbäumen, die kontrafaktische Generierung und anschließend auf die Visualisierung eingegangen wird.

Generierung von Entscheidungsbäumen

Klassische Entscheidungsbaumalgorithmen existieren bereits in vielen Varianten. Auch lässt sich ein Pruning auf bestimmten Algorithmen anwenden. Ermöglicht wird das aber wiederum nicht für alle. Dies liegt zum einen am Algorithmus selbst, z.B. sind erweiterte Algorithmen wie TAO oder GOSTD bereits optimiert oder für Basisvarianten wie ID3 sind noch keine Pruning-Methoden vorgesehen. Die klassischen Methoden ID3, C4.5, CART sollen daher bereitgestellt werden. Darüber hinaus werden weitere optimierte Algorithmen wie HS, GOSTD und TAO bereitgestellt. Für die Algorithmen GOSTD und TAO werden keine Pruning-Methoden und keine

maximale Baumtiefe bereitgestellt. Für ID3, C4.5 und HS kann nur die maximale Tiefe des Baumes konfiguriert werden. Für CART kann sowohl Random-Search als auch Kostenkomplexitätspruning verwendet werden. Zusätzlich kann bei diesem Algorithmus auch die Tiefe des Baumes konfiguriert werden. In Tabelle 2 werden die für die Anwendung auswählbaren Algorithmen mit ihrer Aufgabe, der Art der Eingabedaten, der Konstruktionsstrategie sowie Besonderheiten tabellarisch dargestellt.

Algorithmus	Aufgabe	Typ der Eingabedaten	Konstruktions-Strategie	Besonderheiten
ID3	Klassifikation	kategorisch	Entropie + Informationsgewinn	kein Pruning, Tiefe des Baumes konfigurierbar
C4.5	Klassifikation	kategorisch / kontinuierlich	Entropie + Informationsgewinn	integriertes Pre-Pruning, Tiefe des Baumes konfigurierbar
CART	Klassifikation / Regression	kontinuierlich	Gini-Index oder Entropie als Teilungskriterium	Pre-Pruning + Post-Pruning + Tiefe des Baumes konfigurierbar
TAO	Klassifikation / Regression	kontinuierlich	Grundlage CART + Verbesserung mit TAO	kein Pruning
HS	Klassifikation / Regression	kontinuierlich	Grundlage CART + Regularisierungsmethode	kein Pruning
GOSDT	Klassifikation	kontinuierlich	Grundlage CART + GOSTD-Wrapper	kein Pruning

Tab. 2.: Überblick ausgewählter Entscheidungsbaumalgorithmen

Kontrafaktische Generierung

Das Verfahren von Monteiro et al. liefert mehrere Ergebnisse mit einer hohen Diversität, da verschiedene Algorithmen verwendet werden, die mehrere Punkte liefern. Außerdem ist die Leistung entgegen den Erwartungen besser als bei den anderen Algorithmen. Da die Diversität bei der schrittweisen Suche nach einem idealen Ergebnis sehr hilfreich sein kann, wird das Verfahren als Grundlage für die Generierung der Kontrafaktuale verwendet. Das Verfahren ist in einem Github-Repository in Python bereitgestellt und anhand verschiedener Datensätze demonstriert. Eine Benutzeroberfläche zur Steuerung der Eingabedaten, der Parameter und zur Visualisierung der Ergebnisse ist jedoch nicht vorhanden. Da insbesondere die Steuerung und Visualisierung

das Verständnis der generierten Kontrafaktualen erhöht, wird hier ein Schwerpunkt gesetzt. Teil der Anforderungen ist auch, verschiedene Modellalgorithmen anwenden und diese überprüfen zu können. Dies muss auch bei der Anwendung der kontrafaktischen Generierung berücksichtigt werden. In der Anwendung von Monteiro et al. sind nur bestimmte Algorithmen im Einsatz und kein Pruning auf diese Modelle angewendet. Da die meisten Modellierungsalgorithmen mittlerweile in verschiedenen Bibliotheken verfügbar sind, müssen diese nur in einem einheitlichen Format zur Verfügung gestellt werden, um einerseits die kontrafaktische Generierung darauf anwenden zu können und andererseits auch eine Visualisierung auf Basis dieses Formates durchführen zu können.

Wie bei der Lösung von Monteiro et al. wird eine ML-Pipeline verwendet, um die Eingabedaten aufzubereiten. In der Abbildung 19 besteht die ML-Pipeline aus zwei Schritten. Der erste Schritt ist die Kodierung von kategorialen Daten und die Imputation fehlender Werte für numerische oder kategoriale Daten. Anschließend werden die Daten skaliert und transformiert, wobei Maximalwerte und Einheitsvarianz verwendet werden. Da der Fokus der Implementierung auf der Verarbeitung numerischer Werte liegt, muss bei einer Erweiterung lediglich die Vorverarbeitung innerhalb der ML-Pipeline angepasst werden. Dabei sind die Elemente in den blau- und grünmarkierten Bereichen scikit-learn-Bibliotheken [54].

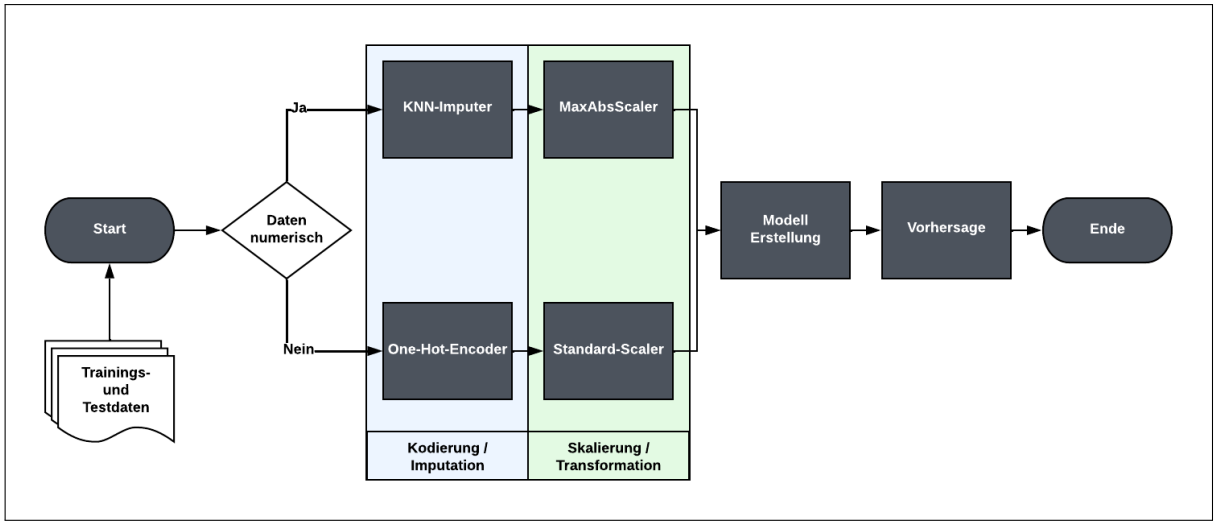


Abb. 19.: Flussdiagramm zur Vorbereitung und Verarbeitung der Daten

Visualisierung

Bereits 1987 haben Larkin et al. in ihrer Arbeit *Warum ein Diagramm (manchmal) so viel wert ist wie zehntausend Worte* auf die Bedeutung von Diagrammen hingewiesen. Für die Gestaltung der Visualisierung wurde insbesondere auf die Arbeit von Munzner et al. zurückgegriffen. Da es sich um eine problemorientierte Arbeit handelt, sprechen Munzner et al. auch von einer

Designstudie, die drei Fragen beantworten muss, um die Entwicklung einer Visualisierung zu unterstützen. In ihrer Arbeit wurde diese Beantwortung für jede Visualisierungsmethode durchgeführt, während sie hier für die gesamte Arbeit beantwortet wird [50].

Was? Da kontrafaktische Analysen in fast allen Bereichen durchgeführt werden können, gibt es kaum Einschränkungen. Statische zweidimensionale Datensätze müssen in Tabellenform eingelesen werden können, sofern sie mindestens ein Attribut und eine Klassifikation besitzen. Die Möglichkeit einer Erweiterung zur Verarbeitung unterschiedlicher Skalenniveaus ist erforderlich. In der ersten Version sollte die Verarbeitung kontinuierlicher Daten ausreichend sein. Ebenso sollte es leicht möglich sein, die Anwendung um die Regressionsaufgabe zu erweitern, da für den ersten Entwurf die Klassifikationsaufgabe vorgesehen ist.

Warum? Der Fokus liegt auf der Aufgabe selbst: Das Suchen und Analysieren von Kontrafaktualen, um lokale Erklärungen abzuleiten. Diese Aufgabe muss sehr intuitiv und effizient lösbar sein.

Wie? Aufgrund der natürlichen Struktur von Entscheidungsbäumen ist eine Visualisierung zwingend erforderlich, um die verschiedenen Algorithmen zu demonstrieren. Da Entscheidungsbäume jedoch nicht alle Informationen der einzelnen Instanzen abbilden und einzelne Datensätze analysiert werden müssen, ist auch eine tabellarische Darstellung erforderlich. Auch für die Erstellung neuer Instanzen bietet sich eine tabellarische Bearbeitungsoberfläche an. Durch die Markierung von Knoten und Verbindungen in verschiedenen Pfaden oder auch der Daten in Tabellen lassen sich Assoziationen einfach herstellen. Darüber hinaus haben Munzner et al. das Konzept des Eye-Tracking Memory eingeführt, in dem erläutert wird, dass das Wechseln zwischen mehreren Ansichten zu einer erhöhten kognitiven Belastung führen kann. Aus diesem Grund werden alle Komponenten in einer Ansicht zusammengefasst.

Die Umsetzung dieses Ansatzes wird in Kapitel 5 beschrieben. Da ein besonderer Schwerpunkt auf der Visualisierung liegt, wurde eine Nutzerstudie durchgeführt, deren Auswertung zur Bewertung und Weiterentwicklung der Anwendung herangezogen wird.

5. Umsetzung der Anwendung

Als Programmiersprache hat sich Python durchgesetzt, da es eine Vielzahl von Bibliotheken im Bereich des maschinellen Lernens bietet. Für die Entwicklung der Benutzeroberfläche eignet sich Flask als Webframework für Python sowie klassisches HTML und Javascript in Kombination mit CSS [25]. Für die DOM-Navigation wird die Javascript-Bibliothek jQuery und für die Visualisierung des Entscheidungsbaums die Bibliothek D3.js verwendet, da diese bereits fertige Komponenten dafür bereitstellt [5]. Zuerst wird das Backend und anschließend das Frontend der Anwendung beschrieben.

5.1. Beschreibung der Logik

Im Folgenden wird die in Python entwickelte Logik vorgestellt. Grundlage aller Anforderungen ist die Generierung der Entscheidungsbäume, anschließend wird die Generierung der Kontrafaktuale vorgestellt. Neben den Kernfunktionen, die im Folgenden beschrieben werden, mussten eine Reihe weiterer Funktionen bereitgestellt werden, wie z.B. die Klassifikations- bzw. Regressionsaufgabe oder auch triviale Funktionen, die für die Visualisierung notwendig sind.

5.1.1. Generierung von Entscheidungsbäumen

Da als Grundlage für die kontrafaktische Generierung die Bibliothek XMOAI verwendet wird und dort die Modelle von scikit-learn zur Anwendung kommen, müssen weitere Modellalgorithmen die gleiche Struktur wie die von scikit-learn aufweisen [62, 54]. Dazu gehören die Funktionen zum Trainieren des Modells und zur Klassifikation. Aus diesem Grund wurden ID3, C4.5 und HS von anderen Bibliotheken verwendet [71]. TAO und GOSTD sind optimierte Modellierungsalgorithmen, die ebenfalls von imodels zur Verfügung gestellt werden [71]. Damit wurde eine Funktion zur Verfügung gestellt, die für einen Algorithmus, eine zugehörige Pruning-Methode und die Trainingsgröße ein Modell erzeugt, das auf einen Datensatz angewendet werden kann. Da bei bestimmten Pruning-Methoden, wie z.B. dem Kostenkomplexitätspruning oder auch der Random- oder Grid-Search, der Datensatz benötigt wird, um die optimalen Modellparameter zu finden, gibt es hier auch eine zweite Funktion, bei der zusätzlich der Datensatz mitgegeben werden muss. Für die Generierung der ML-Pipeline wurde ebenfalls auf die scikit-learn-Bibliothek zurückgegriffen, da diese verschiedene Möglichkeiten zur Vorverarbeitung der Daten bietet [54].

5.1.2. Generierung von Kontrafaktualen

Die kontrafaktische Generierung erwartet eine Reihe von Parametern, die über eine Funktion bereitgestellt werden müssen. Dazu gehören ganz allgemeine Eingabedaten, wie das trainierte Modell, die Originalinstanz und die Zielklasse. Die Generierungsfunktion für Modelle, welche ein

Pruning mit der notwendige Eingabe eines Datensatzes anwenden, ist im Anhang A zur Verfügung gestellt. Des Weiteren gehören alle Parameter zu den Nebenbedingungen dazu, wie z.B. die unveränderlichen Variablen, eine Ober- und Untergrenze für die Variablen sowie die maximale Anzahl der zu verändernden Variablen. Darüber hinaus können weitere Parameter festgelegt werden, wie z.B. die Anzahl der generierten Kontrafaktuale pro Algorithmus, die Anzahl der Threads, die für die Generierung verwendet werden dürfen oder auch die Wahrscheinlichkeit, mit der sich die Individuen der evolutionären Algorithmen paaren. Da Monteiro et al. sowohl eine Klassifikations- als auch eine Regressionsmethode verwenden, muss die Aufgabe von der Anwendung automatisch erkannt werden. Die Ausgabe der Methode ist eine Liste von Instanzen, welche die Kontrafaktuale enthalten. Um wirklich sicher zu gehen, wurde eine weitere Vorhersage nach der Generierung eingebaut, ob die Kontrafaktuale auch wirklich der Zielklasse entsprechen. Die Generierungsfunktion ist im Anhang A zur Verfügung gestellt.

5.1.3. Datenpersistierung

Um die Entscheidungsbäume nicht immer wieder neu generieren zu müssen, sollen diese speicher- und lesbar sein. Das bedeutet, dass sie in einem Format gespeichert und zu einem späteren Zeitpunkt wieder eingelesen werden können. Hintergrund des Problems ist die Neugenerierung des exakt gleichen Modells. Wie bereits im Kapitel 5.1.1 erläutert, existieren Algorithmen bzw. Pruning-Methoden, welche an die Datensätze gebunden sind. Diese Modelle sind jedoch nicht speicherbar, da es nicht sinnvoll ist, die für diesen Datensatz spezifischen Parameter ggf. auf einen anderen Datensatz anwenden zu können. In diesen Fällen ist der Button zum Speichern deaktiviert.

5.2. Beschreibung der Benutzerschnittstelle

In diesem Kapitel liegt der Schwerpunkt auf dem Frontend der Anwendung. Das Frontend ist in Abbildung 20 dargestellt. Dieses ist unterteilt in die Visualisierung der Eingabeparameter, den Entscheidungsbaum selbst und die Konfigurationen für die kontrafaktische Analyse. Wie bereits erwähnt, führt diese Ansicht zu einer geringeren kognitiven Belastung, insbesondere beim Wechsel zwischen allen drei Komponenten. Auf diese Weise können alle Informationen angezeigt werden, ohne dass wichtige Details ausgeblendet werden müssen [50].

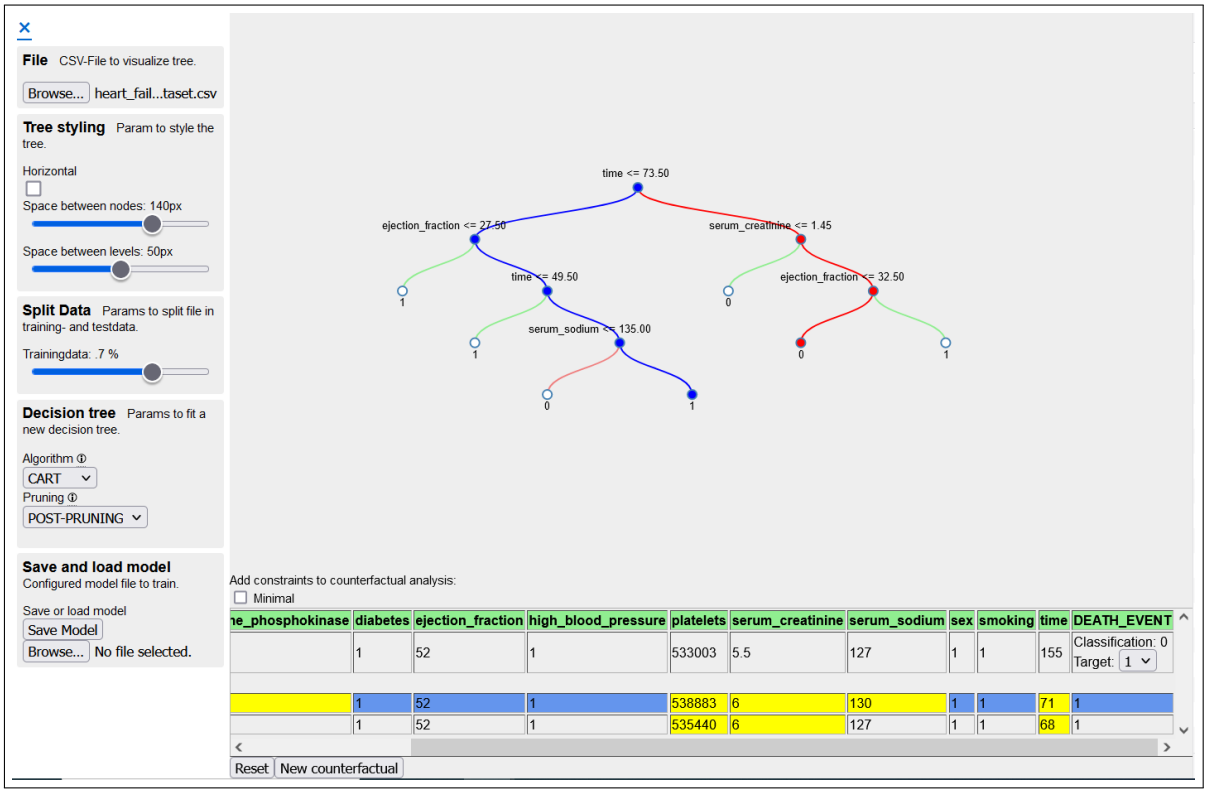


Abb. 20.: Layout der Anwendung mit Eingabeparameter (links), Modell-Visualisierung (mitte) und Parameter zur kontrafaktischen Analyse (unten)

5.2.1. Visualisierung der Eingabeparameter

In der linken Komponente kann der Datensatz hochgeladen, visuelle Anpassungen am Entscheidungsbaum vorgenommen und das Entscheidungsbaummodell konfiguriert werden. Die Komponente ist in Abbildung 21 dargestellt. Da die Werte bereits vorbelegt sind, genügt es, einen Datensatz in Form einer csv-Datei hochzuladen. Der Entscheidungsbaum wird sofort mit den Standardwerten initialisiert und angezeigt. Zu den visuellen Einstellungsmöglichkeiten gehören die Ausrichtung des Baumes und die Abstände zwischen den einzelnen Knoten. Zu den Einstellungen des Entscheidungsbaums gehören, wie bereits erwähnt, die Aufteilung des Datensatzes in Trainings- und Testdaten, der Algorithmus und das verwendete Pruning. Beim Pruning kann der Benutzer auch die maximale Tiefe des Baumes einstellen. Wenn diese Option ausgewählt wird, erscheint eine sogenannter Slider, mit dem die Tiefe konfiguriert werden kann. Außerdem kann das verwendete Modell in dieser Komponente gespeichert und externe Modelle geladen werden.

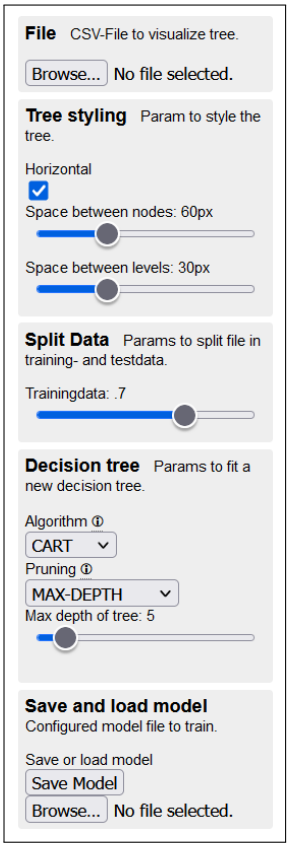


Abb. 21.: Visualisierung der Eingabeparameter aufgeteilt in das Einlesen von Datensätzen (File), optische Anpassungen des Baumes (Tree styling), Aufteilung der Daten in Trainings- und Testdaten (Split Data), Auswahl von Entscheidungsbaumalgorithmen und Pruning (Decision tree) sowie das Speichern und Laden von Modellen (Save and load model)

5.2.2. Visualisierung des Entscheidungsbaumes

Um die Entscheidungsbäume verschiedener Modelle visualisieren zu können, muss jedes Modell in ein einheitliches Format gebracht werden. Dazu wurde eine einfache Transformation in Entscheidungsregeln gewählt, die als JSON-Datei an das Frontend übergeben werden. Ein Auszug der Entscheidungsregeln ist im Anhang A zu finden. Dabei wird der *name* am Knoten angezeigt und *value* für die Auswertertung genutzt. Die *impurity* stellt die Reinheit eines Knotens dar und *samples* die Anzahl an Instanzen, welche innerhalb der Trainingsdaten dem Knoten zugeordnet sind. Zu guter Letzt ist p die Wahrscheinlichkeit des Knotens und *children* die Kindknoten eines Knoten. Auf diese Weise kann mit Hilfe von D3 und dem Baum-Layout über Knoten und Links der Baum von der Wurzel bis zu den Blättern generiert werden [5]. Drei verschiedene Entscheidungbäume sind in den Abbildungen 22, 23 und 24 dargestellt. Die dazugehörigen Datensätze werden in Kapitel 6.1.2 erläutert.

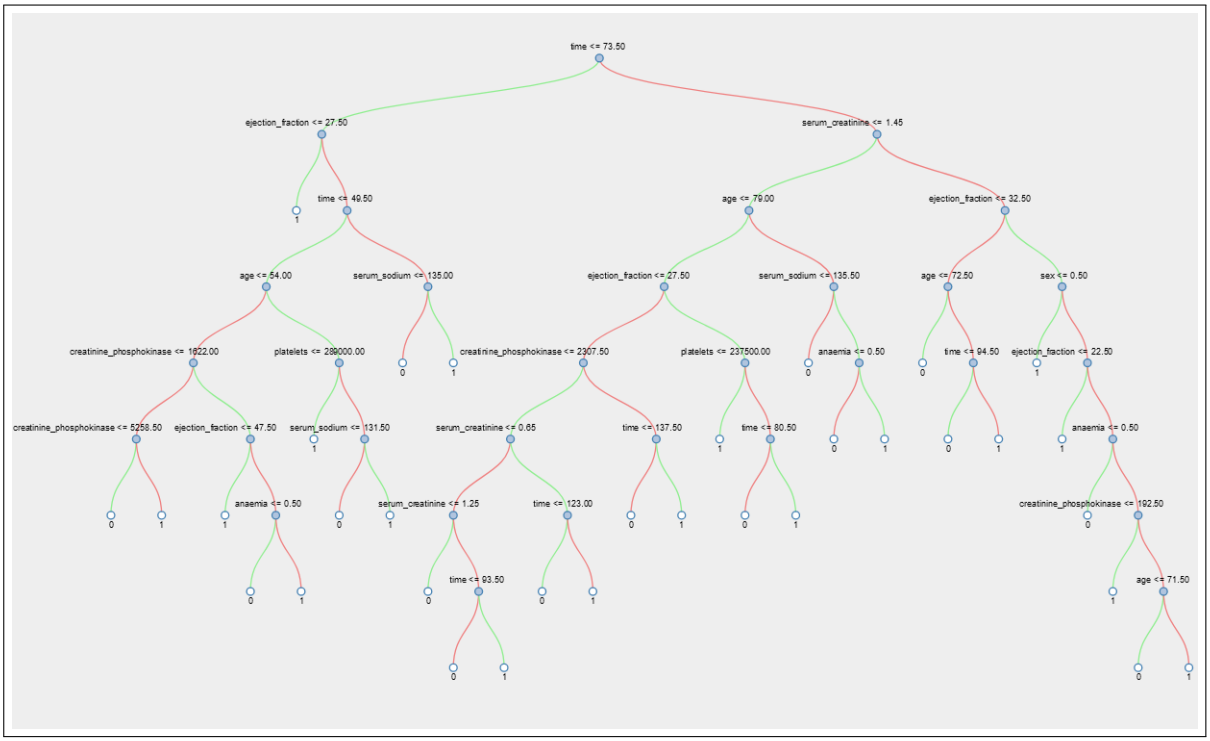


Abb. 22.: Visualisierung des Entscheidungsbaumes auf Basis des Datensatzes *Klinische Aufzeichnungen zu Herzinsuffizienz*

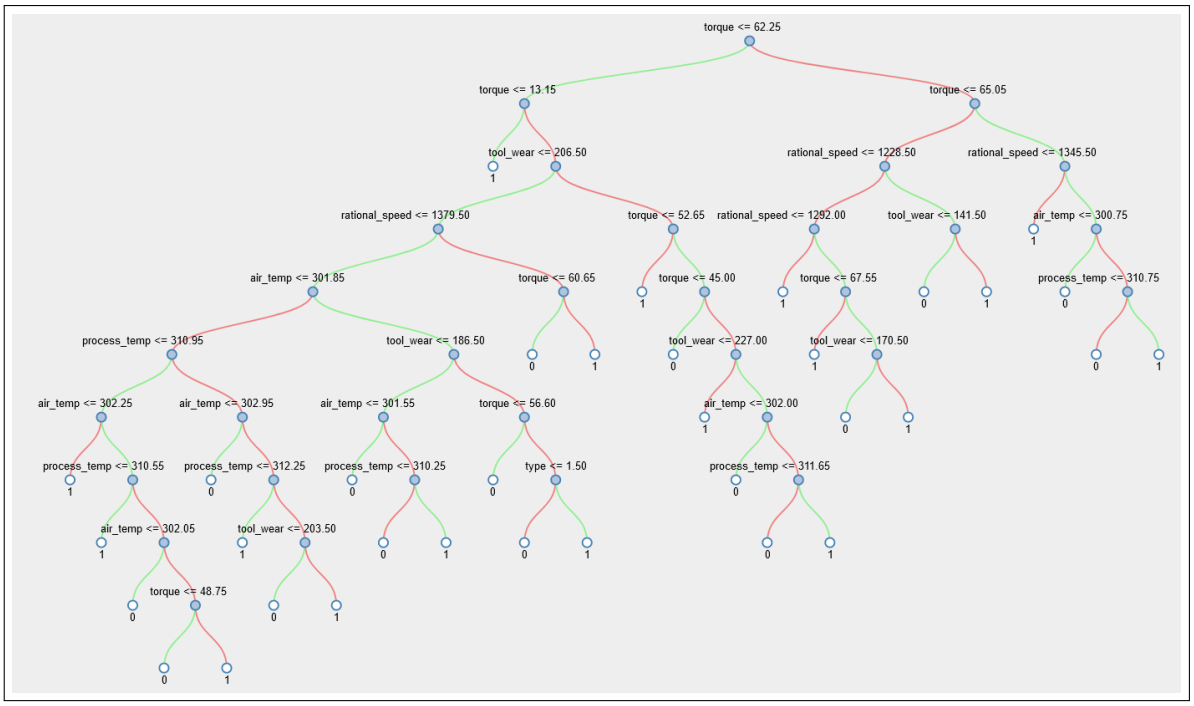


Abb. 23.: Visualisierung des Entscheidungsbaumes auf Basis des Datensatzes *Maschinen-Instandhaltung*

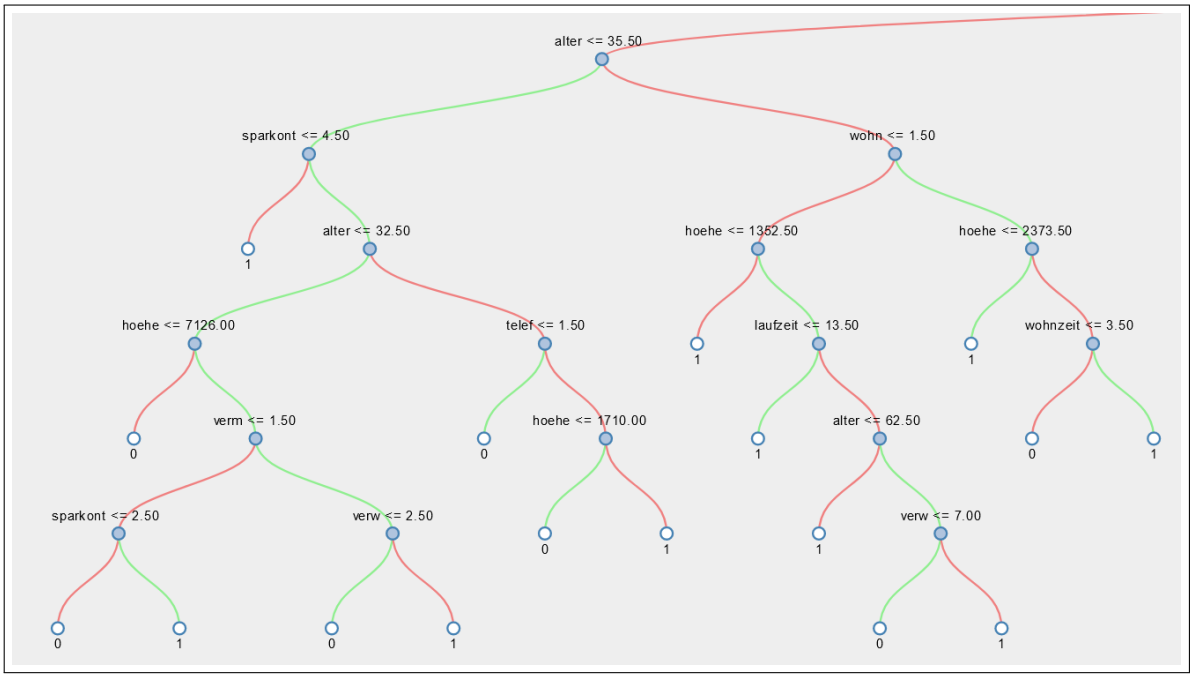


Abb. 24.: Visualisierung des Entscheidungsbaumes auf Basis des Datensatzes *Kreditwürdigkeit in Süddeutschland* (Auszug)

Es wurde eine Funktion zur farblichen Markierung der Pfade entwickelt, die z.B. beim Anklicken eines Knotens ausgelöst wird. Zusätzlich wird beim Anklicken der Wurzel die kontrafaktische Analyse gestartet und die darunterliegende Komponente angezeigt. Zur besseren Unterscheidung sind alle Blattknoten farblich anders dargestellt als die inneren Knoten des Baumes. Wenn der Mauszeiger auf einen Knoten zeigt, werden weitere Informationen wie die Anzahl der Instanzen oder die Reinheit des Knotens angezeigt. Dies ist die schnellste Methode, um zusätzliche Informationen einzublenden, die nicht permanent angezeigt werden sollen. Das Ganze ist in Abbildung 25 veranschaulicht.

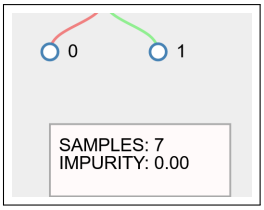


Abb. 25.: Zusätzliche Informationen eines Knotens

5.2.3. Visualisierung der Kontrafaktischen Analyse

Die kontrafaktische Analyse erfolgt in mehreren Schritten. Zunächst müssen die Attribute der gewünschten Instanz in einer Tabelle mit Schiebereglern konfiguriert und die Prognose für diese

Instanz erstellt werden. Dieser Schritt ist in Abbildung 26 veranschaulicht.

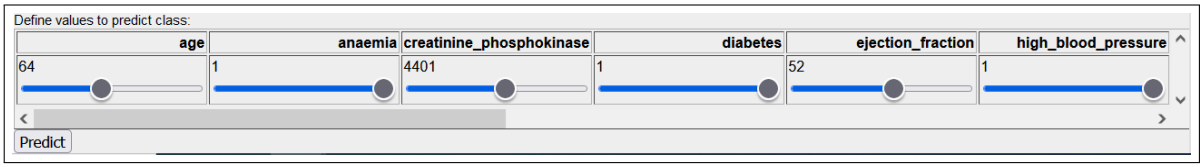


Abb. 26.: Erster Schritt der kontrafaktischen Analyse zur Konfiguration einer neuen Instanz

Nach der Vorhersage der Instanz wird deren Pfad im Entscheidungsbaum farblich (rot) markiert. Im nächsten Schritt kann das neue Ziel ausgewählt werden, wobei in einer Auswahlliste alle Klassifikationen angezeigt werden und bis auf die vorhergesagte Klasse auch alle auswählbar sind. Zusätzlich können die oben genannten Nebenbedingungen eingegeben werden. Zur Eingabe der Ober- und Untergrenze ist ein Klick auf die jeweilige Variable erforderlich. Auch hier werden die Werte über Slider eingegeben. Um Änderungen an bestimmten Variablen zu verhindern, muss ein Klick auf den Variablentitel ausgeführt werden, so dass sich dieser einfärbt. Die minimale Anzahl von Variablenänderungen wird erreicht, wenn die Check-Box *Minimum* aktiviert wird. Der zweite Schritt der kontrafaktischen Analyse ist in Abbildung 27 visualisiert.

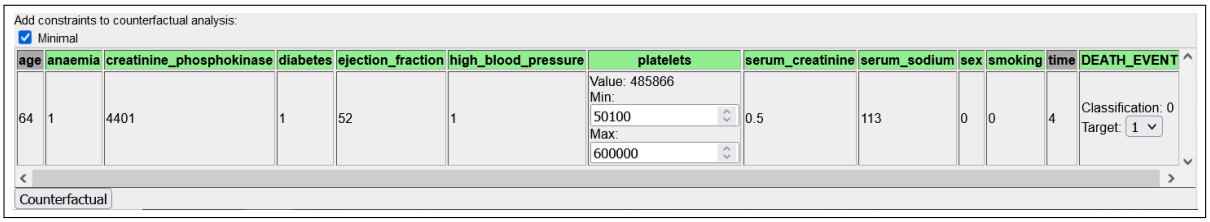


Abb. 27.: Zweiter Schritt der kontrafaktischen Analyse zur Eingabe von Nebenbedingungen

Abschließend können die Kontrafaktuale generiert und die Nebenbedingungen so lange angepasst werden, bis das gewünschte Ergebnis erreicht ist. Diese werden in einer Liste von neu nach alt sortiert angezeigt. In Abbildung 29 ist das Ergebnis anhand der blau markierten Zeile zu erkennen. Bei Auswahl eines Kontrafaktuals wird der entsprechende Pfad farblich (blau) markiert. Ein Beispiel solcher Markierungen wird in Abbildung 28 veranschaulicht.

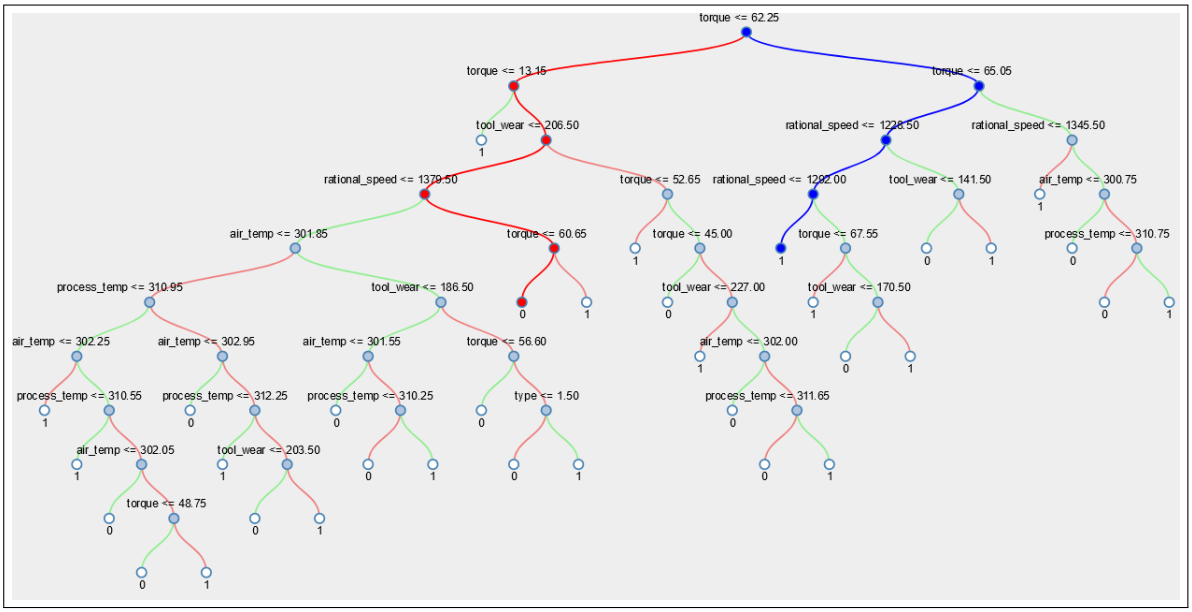


Abb. 28.: Pfadmarkierung im Entscheidungsbaum der Originalinstanz (rot) und des Kontrafaktuals (blau)

Beim Laden eines neuen Kontrafaktuals wird automatisch der oberste Datensatz ausgewählt. Es kann vorkommen, dass Kontrafaktuale nicht generiert werden können, da das Ziel mit den Variablen und Nebenbedingungen nicht erreicht werden kann. In diesem Fall wird dem Benutzer eine Fehlermeldung angezeigt. Da die Generierung von Kontrafaktualen je nach Fall eine unterschiedliche Rechenzeit beansprucht, wird während des Ladevorgangs ein Lade-Icon angezeigt und die Ansicht deaktiviert, so dass in dieser Zeit keine weiteren Aktionen durchgeführt werden können.

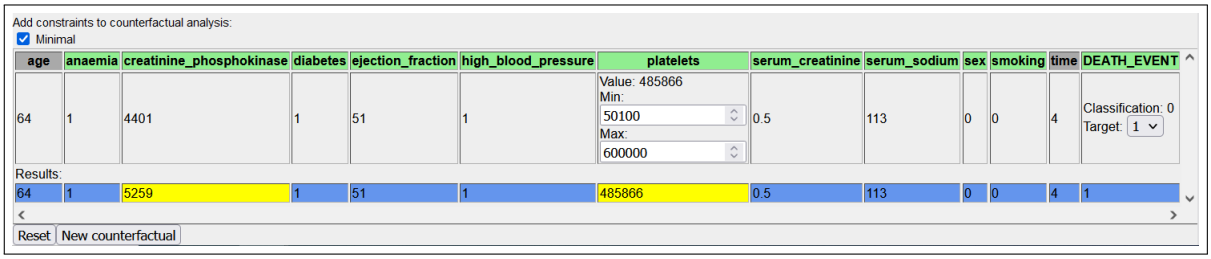


Abb. 29.: Dritter Schritt der kontrafaktischen Analyse zur Generierung von Kontrafaktualen

6. Evaluierung

Im Rahmen der Masterarbeit soll eine Benutzerstudie mit Studenten der Ostfalia Hochschule Wolfenbüttel durchgeführt werden. Die Probanden müssen das Programm in Form eines benutzerorientierten Benchmark-Tests mit retrospektivem Testen ausführen und anschließend anhand eines Fragebogens subjektiv und objektiv bewerten [27]. Zunächst wird das Studiendesign erläutert, anschließend werden die Ergebnisse vorgestellt. Die Nutzerstudie soll helfen, Bedienungsprobleme und Fehler des Programms zu identifizieren und die Gebrauchstauglichkeit zu beurteilen. Auch die subjektive Wahrnehmung der Nutzer spielt eine wichtige Rolle, um die User-Experience bewerten und verbessern zu können. Mit Hilfe dieser Ergebnisse können Potenziale des Programms und Verbesserungsmaßnahmen identifiziert werden. Der Schwerpunkt liegt auf einer qualitativen Analyse.

6.1. Beschreibung des Untersuchungsaufbaus

Im Folgenden werden zunächst die an der Nutzerstudie teilnehmenden Probanden vorgestellt. Im nächsten Schritt werden die für die Auswertung zur Verfügung stehenden Datensätze erläutert. Danach wird der genaue Ablauf der Studie dargelegt und anschließend werden die Artefakte der Studie vorgestellt.

6.1.1. Beschreibung der Probanden

Die Probanden selbst sind Studierende der Ostfalia Hochschule für angewandte Wissenschaften in den Bachelor- und Masterstudiengängen Informatik. Die zehn Probanden haben durch Vorlesungen im Bereich des maschinellen Lernens bereits Erfahrung mit den grundlegenden Konzepten dieser Arbeit. Da es sich um Personen handelt, die sich schnell in die Thematik einarbeiten können, aber keine Experten innerhalb der Domäne sind, wird die Untersuchung unter Beobachtung durchgeführt. Das bedeutet, dass die Teilnehmenden jederzeit Fragen zur Problemstellung oder zur Anwendung stellen können. Fragen und Anmerkungen werden während der Befragung notiert.

6.1.2. Beschreibung der Datensätze

Eine Literaturrecherche wurde im UC Irvine Machine Learning Repository durchgeführt [20]. Die folgenden drei Datensätze aus unterschiedlichen Bereichen werden für die kontrafaktische Analyse verwendet, da sie leicht verständlich sind. Vor jeder Studie wird dem jeweiligen Probanden ein Datensatz zugewiesen, mit dem er sich anhand einer Beschreibung vertraut machen kann.

Klinische Aufzeichnungen zu Herzinsuffizienz

Dieser Datensatz enthält nur 299 Fälle und besteht aus Daten von Patientinnen und Patienten mit und ohne Herzfehler [13]. Er enthält 12 nominale Attribute, keine fehlenden Werte und eine binäre Zielvariable. Zu den Attributen gehören z.B. Alter, Geschlecht und die Information, ob der Patient Raucher, Diabetiker oder Hypertoniker ist [13]. In weiteren Publikationen wurde dieser Datensatz genutzt, um herauszufinden, welche Merkmale maßgeblich an einem Tod durch Herzerkrankung beteiligt sind. Bei der kontrafaktischen Analyse geht es jedoch darum herauszufinden, welche Faktoren verändert werden müssen, um nicht an einer Herzerkrankung zu versterben.

Maschinen-Instandhaltung

Es handelt sich um einen synthetischen Datensatz, der aus 10.000 Instanzen mit 8 Attributen besteht [20]. Die Attribute umfassen Maschineninformationen wie Lufttemperatur, Prozesstemperatur und Werkzeugverschleiß [20]. Die Zielklasse entspricht dem Ausfall der Produktionsmaschine. Ziel ist es zu analysieren, welcher Faktor verändert werden muss, damit die Maschine in einem Szenario nicht ausfällt.

Kreditwürdigkeit in Süddeutschland

Dieser Datensatz besteht aus 1 000 Instanzen mit 21 Attributen [26]. Die Zielklasse definiert, ob einem Kreditnehmer ein Kredit gewährt wurde. Dabei werden Informationen wie Alter, Familienstand, Wohndauer und auch Kreditdaten wie Kredithöhe, Kreditzweck oder auch Kreditrate berücksichtigt [26]. Wie bereits an einem Beispiel erläutert, ist das Ziel der kontrafaktischen Analyse herauszufinden, welche Parameter verändert werden müssten, um eine Kreditvergabe zu ermöglichen, wenn dies nicht der Fall wäre.

6.1.3. Ablauf der Nutzerstudie

Im ersten Schritt werden Dauer und Ziel des Tests erläutert und eine allgemeine Einführung in das Thema gegeben. Anschließend wird die Testumgebung beschrieben, die die Bearbeitung der Aufgaben am Computer und die anschließende Beantwortung eines Fragebogens umfasst. Fragen dazu können jederzeit beantwortet werden. Anschließend werden die einzelnen Aufgaben sowie die drei auswählbaren Datensätze einmal erklärt. Wenn alles verstanden ist, hat der Nutzer zehn Minuten Zeit, um alle Aufgaben zu bearbeiten. Es ist jedoch damit zu rechnen, dass aufgrund des unterschiedlichen Schwierigkeitsgrades nicht jede Aufgabe zwei Minuten in Anspruch nimmt. Während der Bearbeitung werden anwendungsspezifische Fragen dokumentiert. Für jede Aufga-

be wird die Zeit gestoppt, um sie später auswerten zu können. Anschließend füllt der Proband einen Fragebogen aus, für dessen Beantwortung ebenfalls 10 Minuten zur Verfügung stehen.

Beschreibung der Anwendung

Die Testpersonen haben die Wahl zwischen einer mündlichen Erläuterung der Anwendung oder dem Lesen einer Zusammenfassung der Anwendung. Das Handout enthält neben der Zusammenfassung auch die Aufgabenstellung und die Beschreibung der Datensätze. Bei der Beschreibung des Programms ist darauf zu achten, dass die Probanden das Ziel der Anwendung verstehen und die Ergebnisse nachvollziehen können. Die zugrundeliegenden Algorithmen und Pruning-Verfahren werden nicht im Detail erläutert, können aber bei Bedarf erklärt werden. Ein grobes Verständnis des Datensatzes sollte ebenfalls vorhanden sein, um die Ziele der kontrafaktischen Analyse zu verstehen.

Beschreibung der Aufgabenstellung

Um eine kontrafaktische Analyse durchführen zu können, ist der gesamte Arbeitsablauf in fünf Einzelaufgaben unterteilt. Die erste Aufgabe ist einfach und dient eher dazu, sich mit der ersten Ansicht vertraut zu machen. Dabei wird einer der vorgestellten Datensätze ausgewählt und über die Anwendung eingelesen. Ist dies erfolgreich geschehen, soll der Benutzer eine Rückmeldung erhalten, indem der Dateiname im Eingabefeld angezeigt wird und der Entscheidungsbaum mit den Standardparametern visualisiert wird. Im nächsten Schritt muss der Benutzer einen für ihn geeigneten Modellalgorithmus und ein Pruning auswählen. Dabei kann er alle Kombinationen ausprobieren, bis er einen für ihn geeigneten Entscheidungsbaum gefunden hat. Anschließend kann der zuvor generierte Entscheidungsbaum visuell angepasst werden, sodass alle Informationen für den Benutzer sichtbar sind. Die letzten beiden Aufgaben sind deutlich anspruchsvoller. Die Konfiguration und Vorhersage einer Instanz muss durch einen Klick auf die Wurzel des Entscheidungsbaums initialisiert werden. Wurde eine Instanz erfolgreich vorhergesagt, können abschließend die Nebenbedingungen für die Generierung von Kontrafaktualen eingegeben und Kontrafaktuale generiert werden, bis das gewünschte Ergebnis erreicht ist.

1. Einlesen eines Datensatzes
2. Auswahl eines Modellalgorithmus und Pruning
3. Justieren des Entscheidungsbaumes
4. Konfiguration und Vorhersage einer Instanz
5. Generierung von Kontrafaktualen mit Eingabe von Nebenbedingungen

Fragebogen der Nutzerstudie

Mit dem Fragebogen sollen mehrere Ziele erreicht werden. Unter anderem sollen die User-Experience sowie die Usability der Anwendung ermittelt werden. Des Weiteren sollen potentielle Probleme der Anwendung identifiziert und Verbesserungsvorschläge aufgenommen werden. Der Fragebogen befindet sich in Abbildung 6.1.3.

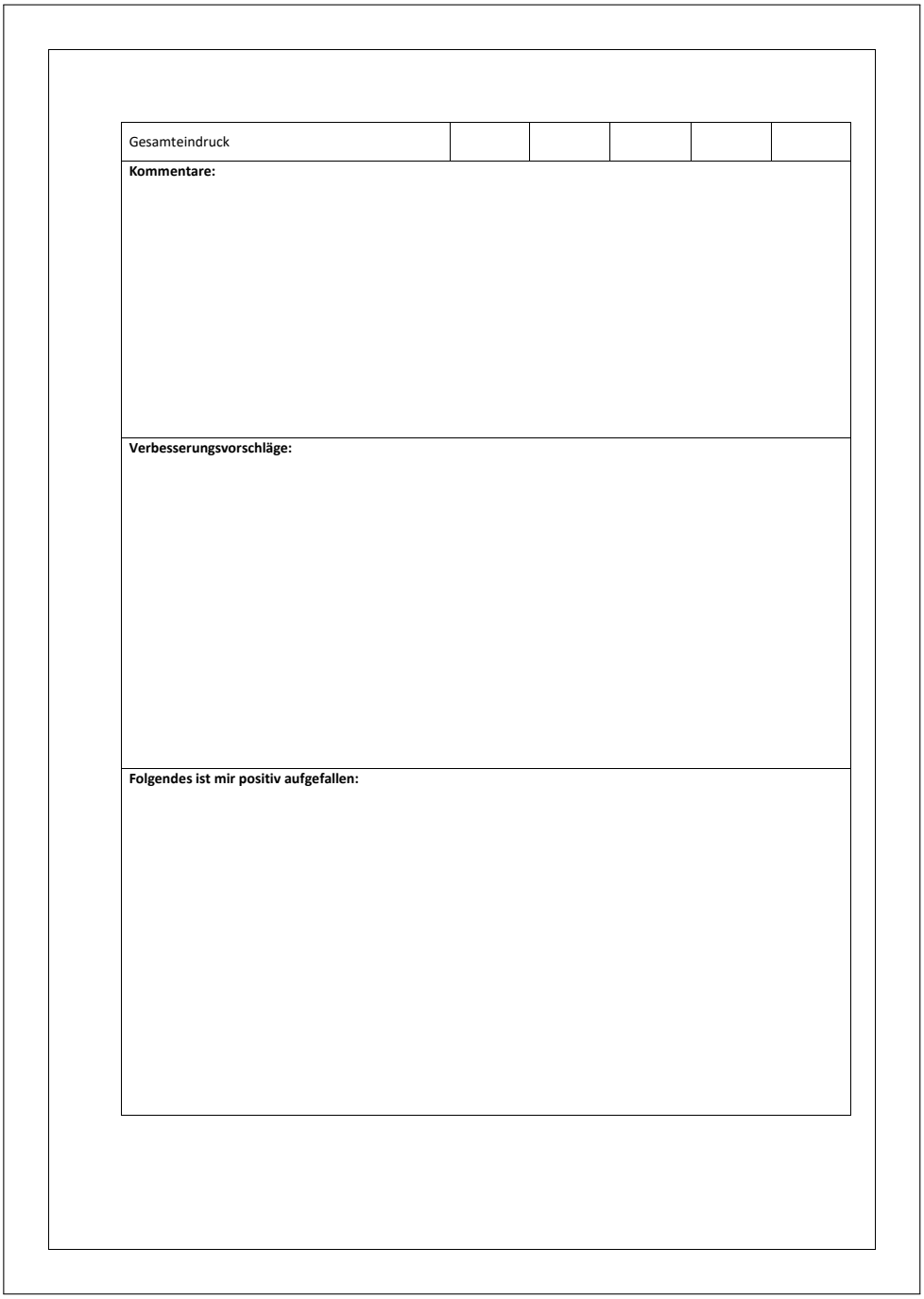
Fragebogen zur Umsetzung einer Anwendung für kontrafaktische Analyse

Nutzerstudie zur Masterthesis

Damit die Anwendung für wissenschaftliche Zwecke verbessert werden kann, bitte ich Sie, diesen Fragebogen auszufüllen. Die Bearbeitung des Fragebogens dauert ca. 5-10 Minuten. Herzlichen Dank für Ihre Mithilfe!

Teilnehmer Nr.	
Datensatz	

	Stimme völlig zu	**Stimme zu**	**Neutral**	**Lehne ab**	**Lehne völlig ab**
Aufgaben					
Ich habe Aufgabe Nr. 1 erfolgreich absolviert.					
Ich habe Aufgabe Nr. 2 erfolgreich absolviert.					
Ich habe Aufgabe Nr. 3 erfolgreich absolviert.					
Ich habe Aufgabe Nr. 4 erfolgreich absolviert.					
Ich habe Aufgabe Nr. 5 erfolgreich absolviert.					
Ich habe keine Unterstützung beim Durchführen der Aufgaben benötigt.					
Ich wusste immer, was der nächste Schritt bei Bedienung der Anwendung ist.					
Usability					
Ich finde, die visuelle Gestaltung trägt zum Verständnis der Funktionen bei.					
Die Anwendung reagiert schnell auf Eingaben.					
Die Anwendung gibt immer Rückmeldung auf Aktionen des Anwenders.					
Die Anwendung liefert verständliche Fehlermeldungen.					
Zu keinem Zeitpunkt hat die Anwendung ein unerwartetes oder unlogisches Verhalten gezeigt.					
	++	**+**	**+-**	**-**	**--**
User-Experience					
Die Ergebnisse der Anwendung konnte ich nachvollziehen.					
Ich denke die Anwendung ist für verschiedene Bereiche einsetzbar.					
Ich vertraue den Ergebnissen der Anwendung.					



Gesamteindruck					
Kommentare:					
Verbesserungsvorschläge:					
Folgendes ist mir positiv aufgefallen:					

Abb. 30.: Fragebogen zur Nutzerstudie

6.2. Resultate der Nutzerstudie

Innerhalb dieses Kapitels werden zuerst die Ergebnisse der Auswahlfelder des Fragebogens, dann die Freitextfelder und zu guter Letzt die Bearbeitungsdauer der Probanden vorgestellt.

6.2.1. Resultate des Fragebogen

Das Ergebnis der Auswahlfelder des Fragebogens aller Probanden befindet sich als Balkendiagramm in Abbildung 6.2.1. Es zeigt alle Bewertungen der Fragen anhand der Kriterien.

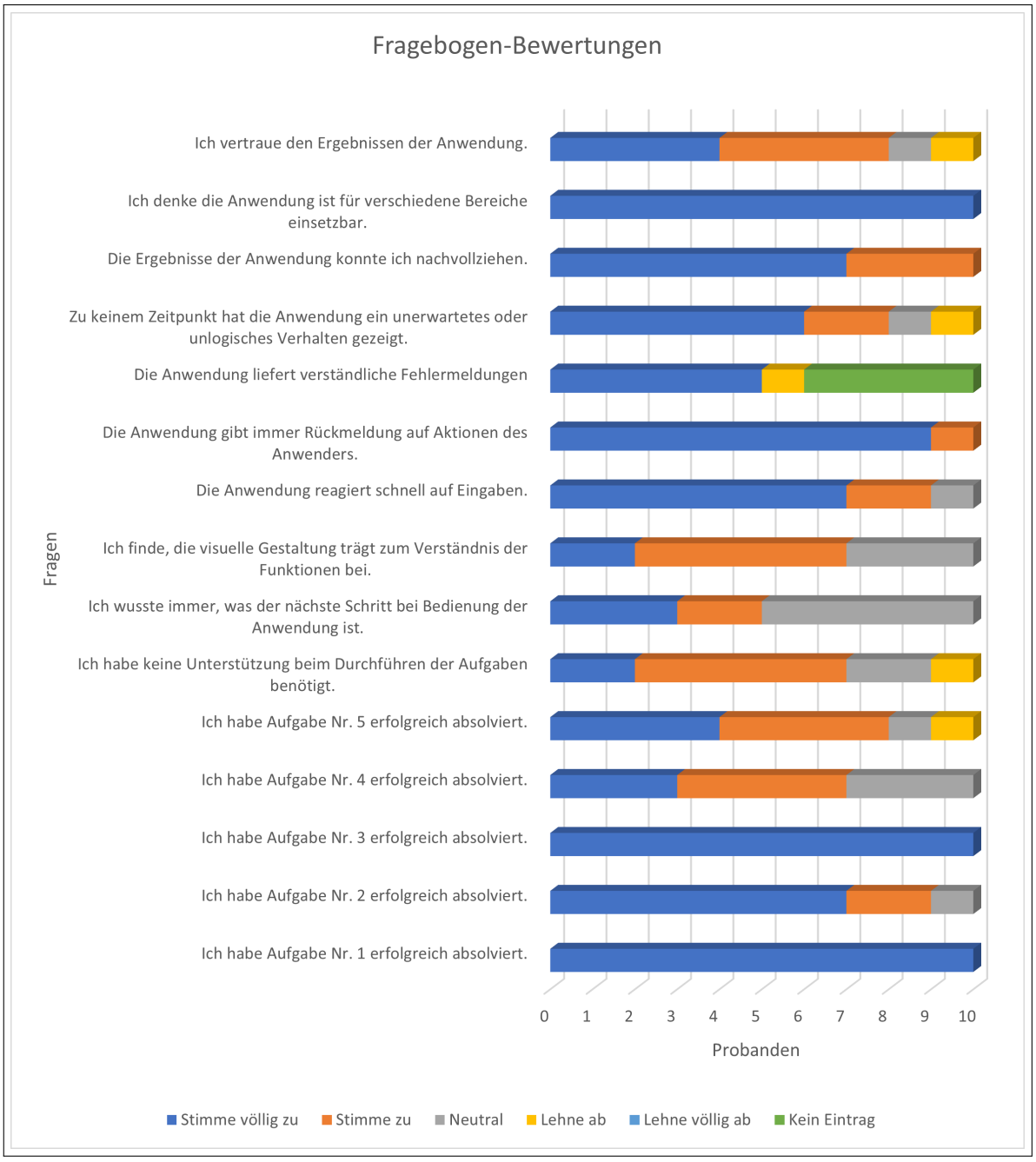


Abb. 31.: Balkendiagramm über Resultate des Fragebogens

Darüberhinaus wird in Abbildung 32 ein Balkendiagramm über die Auswertung des Gesamteindrucks gezeigt.

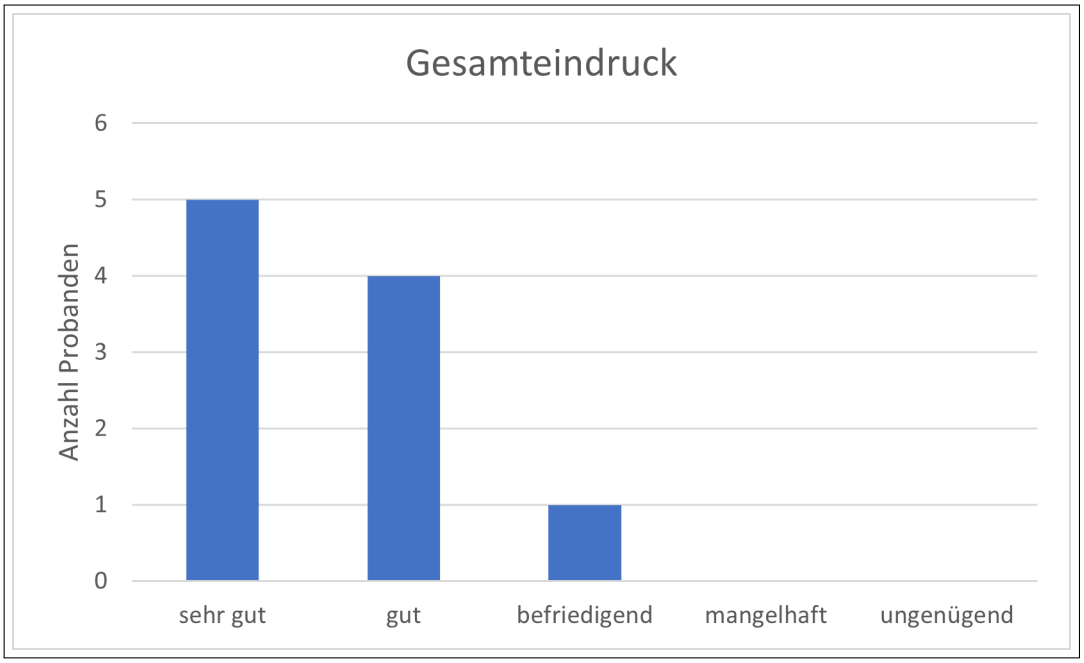


Abb. 32.: Balkendiagramm über Resultate zum Gesamteindruck

6.2.2. Kommentare der Probanden

Folgende Kommentare wurden dem Fragebogen unverändert beigefügt.

1. Wissen über Algorithmen fehlt
2. keine Fehlermeldungen sind aufgetreten
3. Vertrauenswürdigkeit hängt stark mit der Menge und der Qualität der Trainingsdaten ab
4. Scrollen statt Zoomen ist gewöhnungsbedürftig
5. die Beschreibungen der Knoten des Entscheidungsbaumes sehen nicht gut aus
6. es war etwas unerwartet, dass die Konfigurationsmöglichkeit für Instanz nach Klick auf Knoten erscheint
7. Ergebnisse ausversehen durch Klick auf Baumknoten gelöscht
8. Solide Studie, um ein besseres Verständnis von kontrafaktualen zu erhalten
9. Leicht interpretierbar / Intuitiv
10. Nachvollziehbarkeit gewährleistet
11. Informationen zu Formatierung der Daten wären interessant

6.2.3. Verbesserungsvorschläge der Probanden

Für die Anwendung wurden einige Verbesserungen vorgeschlagen. Diese werden im Folgenden aufgelistet.

1. Visualisierung in 3D, Verwendung von Shadern
2. Blending zwischen Parametern
3. Statistiken anzeigen, z.B. Bayes Faktor
4. Genauigkeit des Modells besser darstellen
5. Legende für GUI/Datentabelle
6. Klarere Bezeichnungen der Knoten im Baum
7. Zentrieren-Knopf hinzufügen
8. Algorithmen sollen besser erklärt werden
9. Split Data erklären
10. Knoten sehen besser als Button aus
11. ROC-Kurve zu Modell anzeigen
12. Übersetzungen

6.2.4. Positives Feedback der Probanden

Folgende Punkte wurden im Freitext-Feld aufgezählt, die positiv aufgefallen sind.

1. intuitive GUI
2. Animationen
3. einfach zu benutzen
4. Funktionen sind verständlich und sinnvoll
5. effektiv und günstig
6. minimalistisches Frontend
7. Farben sind komfortabel für das Auge
8. Verständliche Einleitung mit einer kurzen Erklärung im Informationen-Dokument
9. Große Auswahl an Algorithmen; Zudem wurde sichergestellt, dass bestimmte Pruning-Verfahren bei bestimmten Algorithmen nicht verwendet werden können
10. Verschiedene Dateien, je nach Interessensgebiet

6.2.5. Bearbeitungsdauer der Aufgaben

Es wurde die durchschnittliche Bearbeitungzeit der Aufgaben berechnet, welche innerhalb des Balkendiagramms in Abbildung 33 dargestellt ist.

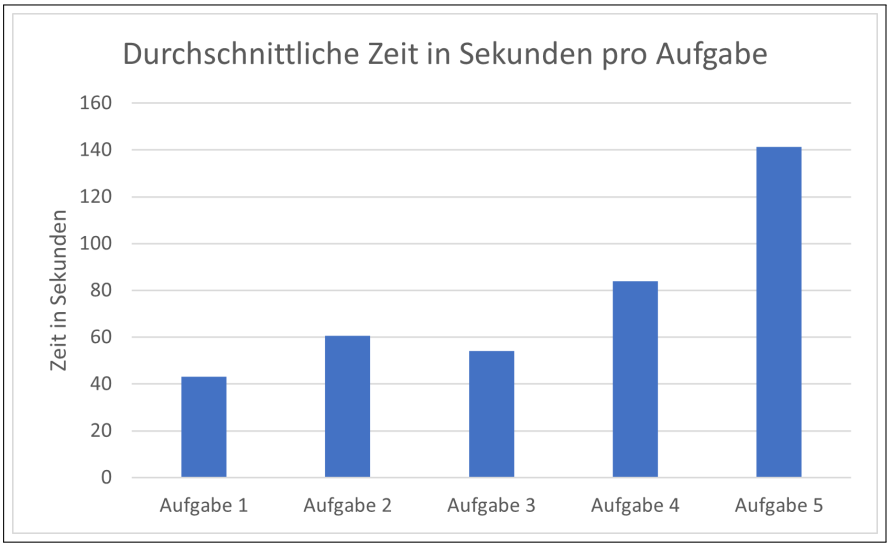


Abb. 33.: Balkendiagramm zur Bearbeitungszeit der Aufgaben durch die Probanden

Und darüberhinaus wird ein Boxplot-Diagramm der Zeitmessungen aller Probanden und Aufgaben in Abbildung 34 gezeigt, um die Verteilungen besser darstellen zu können.

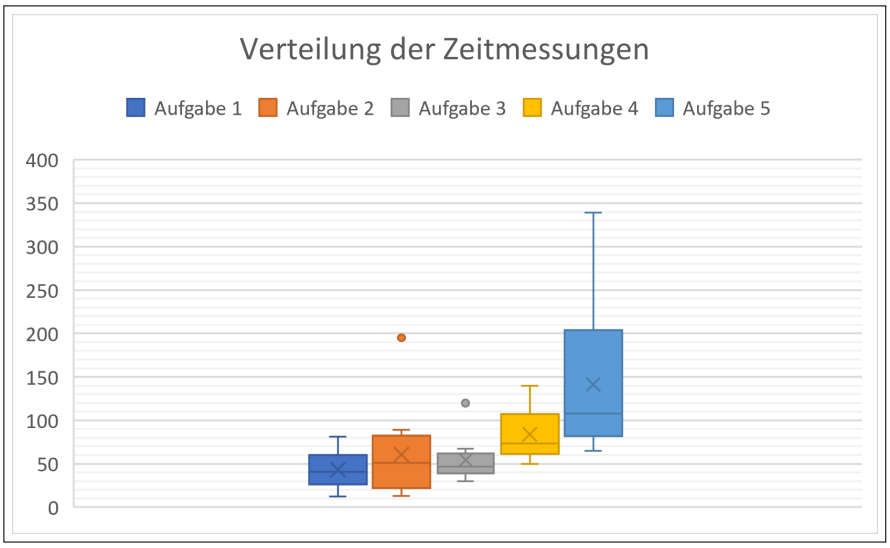


Abb. 34.: Boxplot-Diagramm zur Bearbeitungszeit der Aufgaben durch die Probanden

7. Diskussion

Innerhalb dieses Kapitels werden zuerst die Resultate der Nutzerstudie analysiert und realistische neue Anforderungen daraus abgeleitet. Das Resultat wird mit den Resultaten aus anderen Studien verglichen und es werden die Defizite in kontrafaktischen Erklärungen sowie die Grenzen der Erklärungen innerhalb dieser Arbeit erläutert.

7.1. Resultate der Nutzerstudie

Im Folgenden werden die Resultate der Nutzerstudie diskutiert.

7.1.1. Bewertung der Auswahlfelder

Insgesamt zeigt das Balkendiagramm aller Fragestellungen in Anhang 6.2.1 positive Ergebnisse auf. Alle Probanden stimmten völlig zu, Aufgaben eins und drei erfolgreich absolviert zu haben. Darüber hinaus waren ebenfalls alle der Meinung, dass die Anwendung auch in verschiedenen Bereichen einsetzbar ist. Diese Meinung wurde sicherlich auch dadurch gestützt, dass verschiedene Datensätze für die Studie zur Verfügung gestellt und den Probanden kurz vorgestellt wurden. Auch bei Aufgabe zwei stimmen fast alle zu die Aufgabe erfolgreich absolviert zu haben. Bei den Aufgaben vier und fünf hatten die Probanden offensichtlich größere Probleme, jedoch haben immer noch 70-80% der Probanden zugestimmt die Augaben absolviert zu haben. Die Notizen unterstützen diese Aussage, da besonders das Öffnen der Komponente für die Instanz-Entwicklung sowie das Auswählen der Nebenbedingungen für die kontrafaktische Analyse die größten Herausforderungen dargestellt haben. Ein Proband lehnte berechtigt völlig ab, dass er keine Unterstützung bei Bearbeitung der Aufgaben benötigte. Weiter auffällig sind die Antworten auf die Frage bzgl. der Fehlermeldungen. Diese lässt sich damit beantworten, dass 50% der Anwender keine Fehler angezeigt bekommen haben und diese Aufgabe somit nicht beantworten konnten. Besonders wichtig ist, dass alle Probanden die Ergebnisse der Anwendung nachvollziehen konnten, weil es aussagt, dass sich die Ergebnisse der kontrafaktischen Analyse anhand des Entscheidungsbaumes interpretieren lassen. Ein weiteres bedeutendes Ergebnis ist, dass 80% der Probanden den Ergebnissen der Anwendung vertrauen. Ein Proband hat den Ergebnissen nicht vertraut. Dies ist darauf zurückzuführen, dass dieser Person weitere Statistiken über das Entscheidungsbaummodell selbst und damit Informationen über dessen Qualität fehlten.
Der Gesamteindruck spiegelt auch die Ergebnisse der anderen Felder des Fragebogens wider. 90% der Probanden finden die Anwendung gut bis sehr gut und lediglich ein Proband findet sie befriedigend. Die positiven Rückmeldungen spiegeln genau diesen Eindruck wieder. In den nächsten Kapiteln werden einzelne Probleme und Lösungsvorschläge präsentiert, um daraufhin neue Anforderungen zu definieren.

7.1.2. Bewertung der Kommentaren

Das fehlende Wissen über die Entscheidungbaumalgorithmen war schon bei Studienbeginn zu erwarten. Eine kurze Beschreibung je Algoritmus ist dem Info-Feld beigefügt. Eine weitere Abhilfe könnte wie im Kapitel 7.1.3 die Anzeige von weiteren Statistiken schaffen. Das würde zwar das Wissen über die Algorithmen selbst nicht verbessern, aber der Anwender hätte eine Möglichkeit diese miteinander zu vergleichen.

Dem Hinweis, dass Scrollen für die Zoom-Funktionalität gewöhnungsbedürftig ist, wird nicht ganz zugestimmt, da es eine bekannte Technik in vielen Anwendungen ist, wie z.B. MATLAB Simulink [19].

Die Beschreibungen der Knoten des Entscheidungsbaums sind manchmal nicht gut lesbar und könnten z. B. durch Buttons ersetzt werden, wie dies unter Punkt 10 vorgeschlagen wird. Je nach Länge des Textes wäre das eine einfache Lösung, um die Texte besser lesbar zu machen. Sollte die Länge des Textes eine bestimmte Grenze überschreiten, kann auch über eine Kürzung dieser nachgedacht werden.

Die Anmerkung bzgl. der Anzeige der Konfigurationsmöglichkeiten nach Klick auf den Wurzelknoten ist berechtigt, dennoch haben ein paar Teilnehmer intuitiv und ohne Unterstützung diesen Schritt ausführen können. Vermutlich wäre an dieser Stelle eine Unterstützung, wie ein Hinweis, eine farbliche Markierung oder eine bessere Dokumentation notwendig, um diese Aktion klarer hervorzuheben. Derselbe Proband hat darüberhinaus festgestellt, dass seine Ergebnisse ungeplant nach Klick auf den Entscheidungsbaum verloren gegangen sind. Auch hier wäre bei Anzeige des dritten Dialoges zumindest eine Bestätigung des Anwenders zum Schließen der aktuellen Analyse notwendig, da sonst ein unerwünschtes Verhalten auftreten kann.

7.1.3. Bewertung der Vorschlägen

Aus einer Visualisierung des Entscheidungsbaumes in 3D kann kein sichtbarer Mehrwert gezogen werden, da es sich um ein einfaches Modell handelt. Ganz nach den Prinzipien YAGNI (englisch: *You aren't gonna need it*) und KISS (englisch: *Keep it simple, stupid*), sollte weiterhin angestrebt werden, die Visualisierung so einfach wie möglich zu gestalten [24]. Bei dem Blending zwischen Parametern wurde sich ebenfalls auf die 3D-Visualisierung bezogen. Dies lässt sich jedoch z.B. auf das Pruning anwenden, indem ein Blending zwischen den Visualisierungen angezeigt wird, wenn diese geändert werden. Auf diese Weise wird ein angenehmerer Übergang zwischen den Modellen erreicht und Änderungen am Modell werden besser ersichtlich.

Von besonderem Interesse ist der Vorschlag, die Qualität des Modells in weiteren Statistiken auswerten zu können. Wird bspw. die False-Positive-Rate bzw. die False-Negative-Rate eines Modellalgorithmus und Datensatzes in einer ROC-Kurve abgebildet, so können diese einfach

mit anderen Modellalgorithmen verglichen werden. Außerdem kann das Risiko für einen Fehler, welcher durch die Wahl eines ungeeigneten Algorithmus auftritt, reduziert werden. Auch die Qualität des Datensatzes selbst sollte dabei beachtet werden, da es sich dabei i.d.R. um einen intrinsischen Faktor (Bayes Fehler) handelt, welcher nur in gewissem Maße behoben werden kann. Da die Anwendung die Verarbeitung verschiedenster Datensätze unterstützen muss, sollte hierbei auch das Risiko von Abtastfehlern oder auch, da sich auf einen Modelltypen beschränkt wird, Modellfehlern selbst beachtet werden. Dies würde auch mit Punkte 3 der Kommentare übereinstimmen.

Eine Legende zu den Datensätzen ist ebenfalls ein interessanter Vorschlag. In der Regel werden zu den einzelnen Attributen der Datensätze Beschreibungen in einem eigenen Dokument beigelegt. Besonders bei kategorialen Attributen kann die Beschreibung des Attributs bzw. seiner Belegungen sehr sinnvoll sein. Eine Möglichkeit zur Anzeige der Beschreibung würde dem Anwender helfen ein besseres Verständnis der Ergebnisse zu entwickeln, indem alle notwendigen Informationen an einer Stelle gesammelt werden. Eine Beobachtung der Studie war ebenfalls, dass viele Probanden stark zwischen der Bearbeitung der Aufgaben und dem Handout des Datensatzes gewechselt haben. Die Legende könnte entweder separat hinzugefügt oder bei Zeigen mit der Maus auf die jeweiligen Attribute angezeigt werden, um das ständige Wechseln zwischen den Dokumenten zu vermeiden.

Das Hinzufügen eines Zentrieren-Buttons ist eine einfache aber effektive Änderung dem Anwender die Möglichkeit einer automatisierten Zentrierung der Entscheidungsbaumvisualisierung zu bieten.

Eine wichtige Anmerkung eines Probanden war, dass die Aufteilung der Daten in Trainings- und Testdaten nicht verstanden worden ist. Das könnte zum Einen an fehlendem Wissen liegen, aber auch an den Variationsmöglichkeiten von Trainings- und Testdaten, da es neben der normalen Aufteilung auch weitere Möglichkeiten gibt. Dazu gehören z.B. die Kreuzvalidierung, welche bereits in Kapitel 4 beschrieben worden ist oder auch das Bootstrapping, bei dem die Standardabweichung für jeden Parameter anhand der Trainings-und Testdaten verwendet wird. Da die meisten Modellalgorithmen jedoch nur das Standardverhalten bieten, gibt es an dieser Stelle vorerst keinen Handlungsbedarf.

Bei der Anmerkung zu gewünschten Übersetzungen kann es sich zum Einen um die Anwendung selbst handeln, sodass diese bspw. auch andere Sprachen, wie Deutsch unterstützt. Andererseits kann es sich dabei aber auch um die Datensätze handeln, da diese größtenteils in englischer Sprache zur Verfügung gestellt worden sind. Ersteres kann sinnvoll sein, ist aber keine besonders wichtige Anforderung. Zweiteres kann der Anwender selbst beeinflussen, indem er seine Daten vorher sichtet und entsprechend bearbeitet.

7.1.4. Bewertung der Bearbeitungszeit

Die gemessenen Zeiten liefern nur bedingt Rückschlüsse zu der Anwendung, da verschiedene Datensätze mit unterschiedlichen Anzahlen an Attributen verwendet worden sind und alle Probanden so viel Zeit wie gewünscht zum Probieren aller Funktion hatten. Sowohl die durchschnittlichen Zeiten als auch das Boxplot spiegeln die Antworten auf den Fragebogen zu den Aufgaben wider. Dabei wurden die ersten drei Aufgaben am schnellsten durchgeführt. Bei Aufgabe zwei wird deutlich, dass manche Probanden deutlich mehr experimentiert haben als andere. Die letzte Aufgabe war zeitlich am intensivsten. Das ist vermutlich darauf zurückzuführen, dass die Anwender versucht haben mehrere Kontrafaktuale zu generieren, indem verschiedene Nebenbedingungen eingegeben worden sind. Desweiteren haben die Probanden innerhalb dieser Aufgabe die Ergebnisse analysiert und nachvollzogen.

7.1.5. Handlungsempfehlungen aus der Nutzerstudie

Im Folgenden werden die Handlungsempfehlungen zur Weiterentwicklung der Anwendung aus der Analyse gefiltert und von niedrig nach hoch priorisiert im Folgenden aufgelistet:

1. Um zum Ausgangszustand wieder zurückgelangen zu können, soll ein Button zum Zentrieren der Visualisierung hinzugefügt werden.
2. Um die Beschreibungen des Entscheidungsbaumes besser erkennen zu können, sollen sich diese besser an der Länge des Textes sowie der Größe des Baumes anpassen.
3. Um die Unterschiede zwischen verschiedenen Modellparametrisierungen besser erkennen zu können, soll ein Blending der Visualisierung hinzugefügt werden.
4. Um den Konfigurationsschritt einer Instanz intuitiver zu gestalten, soll dieser in der Anwendung besser gekennzeichnet werden.
5. Um den aktuellen Vorgang nicht versehentlich abzubrechen, sollen beim Wechsel von der kontrafaktischen Analyse zu anderen Vorgängen Bestätigungen des Anwenders erforderlich sein.
6. Um die Datensätze besser verstehen zu können und nicht zur Dokumentation dieser wechseln zu müssen, soll eine Legende zu diesen zur Verfügung gestellt werden, wenn die Informationen dazu bereitgestellt sind.
7. Um das Vertrauen in die Ergebnisse zu erhöhen und die Qualität der Modelle besser vergleichen zu können, sollen Statistiken der Modelle, z.B. in Form von ROC-Kurven, Konfusionsmatrix oder auch Abweichungen, wie die mittlere quadratische Abweichung, visualisiert werden.

7.2. Vergleich zu anderen Resultaten

Die Anwendung von Cheng et al. liegt in ihrem Ziel und der Verwendung dieser Anwendung am nächsten. In der Visualisierung können Instanzen mit Hilfe von Slidern erzeugt werden, wie es auch in dieser Anwendung der Fall ist [11]. Da es sich bei dieser Anwendung um eine neuartige Entwicklung handelt, lässt sich diese nur schwer mit anderen Ergebnissen quantitativ und qualitativ vergleichen. Es lassen sich wie bereits erläutert Parallelen zu diesen ziehen. Der entscheidende Vorteil zu allen anderen Anwendungen ist der gezielte Einsatz von natürlich interpretierbaren Modellen, um anhand dieser die Ergebnisse der kontrafaktischen Analyse durch die Visualisierung besser verstehen zu können. Die anderen Lösungen haben gezeigt, wie hilfreich zusätzliche Leistungsindikatoren für die Analyse des Modells und gezielte Interviews mit Fachexperten sein können.

7.3. Grenzen und Defizite

Im Folgenden werden zum Einen Grenzen der Anwendung selbst und Defizite innerhalb von kontrafaktischen Analysen aufgewiesen.

7.3.1. Skalierbarkeit der Entscheidungsbäume

Eines der größten Probleme bei der Visualisierung ist die Skalierbarkeit. Zwar impliziert eine große Datenmenge nicht direkt, dass das dazu erstellte Modell komplexer wird, kann jedoch dazu beitragen. Je größer und komplexer das Modell, desto weniger verständlich ist es für den Anwender. Ein starkes Pruning bewirkt wiederum leider das Gegenteil, eine zu starke Verallgemeinerung des Modells. Nicht nur die Komplexität des Modells kann zu Problemen führen. Eine hohe Anzahl an Attributen, welche der Anwender verstehen und überschauen muss kann die Übersichtlichkeit reduzieren und somit das Verständnis der Ergebnisse erschweren.

7.3.2. Beschränkter Einsatz des Modells

Nicht jedes Problem kann mit Entscheidungsbäumen abgedeckt werden. Besonders komplexe oder große Datensätze können den Einsatz anderer Modelle fordern, wie bspw. KNNs oder Support Vector Machines. Beispiel dafür wäre die Verarbeitung von Bild- oder Sprachdaten. Der Einsatz von Entscheidungsbäumen wäre an dieser Stelle zu ungenau und rechenaufwändig. Einen Ausblick zu einer möglichen Lösung für dieses Problem wird in Kapitel 8.1 geliefert.

Auch das Teilgebiet des Clusterings kann mit den in der Anwendung behandelten Entscheidungsbaumalgorithmen nicht abgedeckt werden, auch wenn kontrafaktische Analysen in diesem Bereich durchaus von Interesse sind.

7.3.3. Logische Erklärungen der kontrafaktischen Analyse

Wenn die Qualität es zugrundeliegenden Modells sehr hoch ist, kann es zu unlogischem Verhalten innerhalb der kontrafaktischen Analysen führen. Im Beispiel des Bankenwesens könnte dies z.B. in einem Kontrafaktual resultieren, welches besagt, dass der Kredit erhöht werden muss, um der Zielklasse zu entsprechen. Dieses Verhalten kann zwar in Form von Eingabe von Nebenbedingungen schnell behoben werden, erfordert aber immer die manuelle Nachbearbeitung. Ein vollständig automatisiertes Verfahren ist dabei nicht denkbar. Wie bei allen nicht automatisierten Verfahren kann es hier schneller zu Fehleingaben und somit auch falschen Schlussfolgerungen kommen.

7.4. Auswertung der Anwendung

Insgesamt konnte mit der Anwendung und der Nutzerstudie erfolgreich gezeigt werden, dass mit Hilfe von Entscheidungsbäumen als Lernalgorithmen und visuellen Komponenten zur Unterstützung kontrafaktische Erklärungen nachvollziehbar sind.
Obwohl die Probanden über wenig Vorwissen zum Thema verfügten, konnten sie sich durch die Anwendung schnell in die Problematik einarbeiten, die Ergebnisse nachvollziehen und ihnen zudem größtenteils vertrauen. Sie brachten auch viele Verbesserungsvorschläge ein, die teilweise in neue Anforderungen an die Anwendung umgesetzt wurden. Diese Arbeit bietet eine gute Grundlage für zukünftige Themen, die im Kapitel 8 behandelt werden und ein hohes Potential für den Einsatz in verschiedenen Domänen.
Rückblickend wäre die Durchführung der gleichen Evaluationsmetriken von Hoffmann et al. wie sie für die Anwendungen CX-ToM, LIME oder SHAP durchgeführt wurden, aufschlussreicher für die Evaluation gewesen [2, 58, 42, 29]. Dies würde einen besseren Vergleich zwischen den einzelnen Anwendungen ermöglichen. Dies schmälert jedoch in keiner Weise die Ergebnisse dieser Arbeit. Darüber hinaus wäre es wünschenswert, in einer nächsten Studie auch die Meinung von Fachexperten einzuholen, um die bisherigen Ergebnisse noch einmal bestätigen zu können.

8. Zukünftige Arbeit

In der Diskussion wurden einige Defizite aufgelistet, welche Anhaltspunkte für zukünftige Arbeiten darstellen. Einige Anregungen und weitere Referenzen für zukünftige Arbeiten werden im Folgenden aufgeführt.

8.1. Kontrafaktische Erklärungen bei Black-Box Modellen

Um Probleme zu lösen, welche nicht mit Entscheidungsbäumen abgedeckt werden können, gibt es die Möglichkeit der Anwendung von Surrogatmodellen. Ein Black-Box Modell kann in ein Surrogatmodell umgewandelt werden, um daraus Erklärungen abzuleiten, indem bspw. kontrafaktische Analysen darauf durchgeführt werden können. Eine Handlungsempfehlung wäre somit die Integration dieses Verfahrens in die Anwendung. Einen ähnlichen Ansatz haben Bertram et al. verfolgt, indem sie Black-Box Modelle, wie Random Forrests und Support Vektor Machines in ein Surrogatmodell umwandeln und kontrafaktische Analysen darauf ausführen können [63]. Bertram et al. haben einzelne Schritte extrahiert, welche notwendig sind für die Generierung von Kontrafaktualen auf Basis von Black-Box Modellen, um Erklärungen nicht natürlich interpretierbarer Modelle herzuleiten. Für die Generierung sind dabei folgende Schritte notwendig [81].

Phase 1: Finden der ersten Kontrafaktualen

Dies kann mit verschiedenen Methoden, wie z.B. anhand von zufälligen Stichproben erfolgen. Darüberhinaus kann an dieser Stelle auch eine Optimierung wie in 3.1 erfolgen.

Phase 2: Finden von unterstützenden Punkten

Hier können zufällige oder sogenannte magnetische Stichproben angewandt werden, um Instanzen mit dem Zielwert zu finden. Dabei wird ein Richtungsvektor, welcher mithilfe der euklidischen Norm ($|.|$) bestimmt wird sowie der Abstand zwischen der ersten Kontrafaktualen und der Originalinstanz berechnet. Anhand dieser wird um die Originalinstanz nach neuen unterstützende Punkte abgetastet.

Phase 3: Finden von Entscheidungsgrenzen

Anhand der Originalinstanz und unterstützenden Punkte kann nun eine Entscheidungsgrenze lokalisiert werden. Dafür können Methoden wie die lineare oder binäre Suche verwendet werden.

Phase 4: Trainieren des Surrogatmodells

Die Ergebnisse aus Phase 3 werden nun verwendet, um ein Surrogatmodell zu trainieren. Dafür wird als Optimierungsproblem versucht die Loss-Funktion zu minimieren und idealerweise ein Modell gewählt, welches auch visuell Erklärungen liefern kann.

Phase 5: Ableiten von Erklärungen

Anhand des Modells können nun die Merkmalsbedeutung und relative Differenzen abgeleitet werden. Das Ganze kann besonders bei Entscheidungsbäumen visuell durch Simulationen verstärkt werden.

8.2. Clustering im Bereich der Kontrafaktischen Analyse

Darüberhinaus sind kontrafaktische Analysen im Bereich des Clustering von hohem Interesse. Es gibt Entscheidungsbaumalgorithmen, welche Clustering ermöglichen, wie bspw. die Lösung von Lui et al. [40]. Bei diesem partitionellen Clustering wird ein Teile-und-Herrsche Verfahren angewandt, um die Daten zu partitionieren. Shichao et al. wenden hingegen ein hierarchisches Clustering Verfahren an [3]. Innerhalb dieses Verfahrens wird eine sogenannte Talerkennung anhand von Histogrammen und Segmentierung für die Partitionierung angewandt.

Literaturverzeichnis

[1] Agarwal, Abhineet ; Tan, Yan S. ; Ronen, Omer ; Singh, Chandan ; Yu, Bin: Hierarchical Shrinkage: improving the accuracy and interpretability of tree-based methods. In: *arXiv preprint arXiv:2202.00858* (2022)

[2] Akula, Arjun R. ; Wang, Keze ; Liu, Changsong ; Saba-Sadiya, Sari ; Lu, Hongjing ; Todorovic, Sinisa ; Chai, Joyce ; Zhu, Song-Chun: CX-ToM: Counterfactual explanations with theory-of-mind for enhancing human trust in image recognition models. In: *Iscience* 25 (2022), Nr. 1

[3] Basak, Jayanta ; Krishnapuram, Raghu: Interpretable hierarchical clustering by constructing an unsupervised decision tree. In: *IEEE transactions on knowledge and data engineering* 17 (2005), Nr. 1

[4] Blank, Julian ; Deb, Kalyanmoy: Pymoo: Multi-objective optimization in python. In: *IEEE Access* 8 (2020), S. 89497–89509

[5] Bostock, Mike: *D3.js - Data-Driven Documents*. 2012. – URL `http://d3js.org/`

[6] Bradford, Jeffrey P. ; Kunz, Clayton ; Kohavi, Ron ; Brunk, Cliff ; Brodley, Carla E.: Pruning decision trees with misclassification costs. In: Nédellec, Claire (Hrsg.) ; Rouveirol, Céline (Hrsg.): *Machine Learning: ECML-98*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1998. – ISBN 978-3-540-69781-7

[7] Burkart, Nadia ; Huber, Marco F.: A Survey on the Explainability of Supervised Machine Learning. In: *Journal of Artificial Intelligence Research* (January 2021)

[8] Carreira-Perpinán, Miguel A. ; Tavallali, Pooya: Alternating optimization of decision trees, with application to learning sparse oblique trees. In: *Advances in neural information processing systems* 31 (2018)

[9] Carreira-Perpiñán, Miguel ; Hada, Suryabhan S.: Counterfactual Explanations for Oblique Decision Trees:Exact, Efficient Algorithms. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021), May, Nr. 8. – URL `https://ojs.aaai.org/index.php/AAAI/article/view/16851`

[10] Ceneda, Nicolo: *Quantile Regression of High-Frequency Data Tail Dynamics via a Recurrent Neural Network*, Dissertation, 05 2020

[11] Cheng, Furui ; Ming, Yao ; Qu, Huamin: Dece: Decision explorer with counterfactual explanations for machine learning models. In: *IEEE Transactions on Visualization and Computer Graphics* 27 (2020), Nr. 2

[12] Cheng, Jade ; Mailund, Thomas: Ancestral population genomics using coalescence hidden Markov models and heuristic optimisation algorithms. In: *Computational Biology and*

Chemistry 360 (2015), 03

[13] Chicco, Davide ; Jurman, Giuseppe: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. In: *BMC medical informatics and decision making* 20 (2020), Nr. 1

[14] Coello, CA C.: Evolutionary multi-objective optimization: a historical view of the field. In: *IEEE computational intelligence magazine* 1 (2006), Nr. 1

[15] Dandl, Susanne ; Molnar, Christoph ; Binder, Martin ; Bischl, Bernd: Multi-Objective Counterfactual Explanations. (June 2020)

[16] Deb, Kalyanmoy: *Multi-objective optimisation using evolutionary algorithms: an introduction*. Springer, 2011

[17] DIN EN ISO 9241-110: *DIN EN ISO 9241-110:2008-09, Ergonomie der Mensch-System-Interaktion - Teil 110: Grundsätze der Dialoggestaltung (ISO 9241-110:2006); Deutsche Fassung EN ISO 9241-110:2006*. 09 2008. – URL `https://doi.org/10.31030%2F1464024`

[18] Dittes, Frank-Michael: Und er würfelt doch: Monte-Carlo-Verfahren der globalen Optimierung. In: *Optimierung: Wie man aus allem das Beste macht*. Springer, 2022

[19] Documentation, Simulink: *Simulation and Model-Based Design*. 2020. – URL `https://www.mathworks.com/products/simulink.html`

[20] Dua, Dheeru ; Graff, Casey: *UCI Machine Learning Repository*. 2017. – URL `http://archive.ics.uci.edu/ml`

[21] Dung Duong, Tri ; Li, Qian ; Xu, Guandong: Prototype-based Counterfactual Explanation for Causal Classification. (May 2021)

[22] Gerdes, Ingrid ; Klawonn, Frank ; Kruse, Rudolf ; Gerdes, Ingrid ; Klawonn, Frank ; Kruse, Rudolf: Optimierungsprobleme. In: *Evolutionäre Algorithmen: Genetische Algorithmen—Strategien und Optimierungsverfahren—Beispielanwendungen* (2004)

[23] Ghasemalizadeh, Omid ; Khaleghian, Meysam ; Taheri, Saied: A Review of Optimization Techniques in Artificial Networks. In: *International Journal of Advanced Research* 4 (2016), 09

[24] Goll, Joachim ; Goll, Joachim: Entwurfsprinzipien zur Vermeidung von Überflüssigem. In: *Entwurfsprinzipien und Konstruktionskonzepte der Softwaretechnik: Strategien für schwach gekoppelte, korrekte und stabile Software* (2018)

[25] Grinberg, Miguel: *Flask web development: developing web applications with python*. Ö'Reilly Media, Inc.", 2018

[26] Groemping, Ulrike: South German credit data: Correcting a widely used data set. In: *Rep. Math., Phys. Chem., Berlin, Germany, Tech. Rep* 4 (2019)

[27] Hegner, Marcus: Methoden zur Evaluation von Software. (2003)

[28] Hetzel, Leon ; Wangelik, Frederik: Künstliche neuronale Netze: Ein Nachbau unseres Gehirns? In: *Wie Maschinen lernen: Künstliche Intelligenz verständlich erklärt* (2019)

[29] Hoffman, Robert R. ; Mueller, Shane T. ; Klein, Gary ; Litman, Jordan: Metrics for Explainable AI: Challenges and Prospects. In: *CoRR* abs/1812.04608 (2018). – URL http://arxiv.org/abs/1812.04608

[30] Jarre, Florian ; Stoer, Josef: *Lineare Optimierung: Anwendungen, Netzwerke.* In: *Optimierung.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2004. – URL https://doi.org/10.1007/978-3-642-18785-8_5. – ISBN 978-3-642-18785-8

[31] Johansson, Fredrik ; Shalit, Uri ; Sontag, David: Learning representations for counterfactual inference. In: *International conference on machine learning* PMLR (Veranst.), 2016

[32] Keane, Mark T. ; Kenny, Eoin M. ; Delaney, Eoin ; Smyth, Barry: If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques. (February 2021)

[33] Klaise, Janis ; Looveren, Arnaud V. ; Vacanti, Giovanni ; Coca, Alexandru: Alibi Explain: Algorithms for Explaining Machine Learning Models. In: *Journal of Machine Learning Research* 22 (2021), Nr. 181. – URL http://jmlr.org/papers/v22/21-0017.html

[34] Klein, Kyle ; Neira, Julian: Nelder-Mead Simplex Optimization Routine for Large-Scale Problems: A Distributed Memory Implementation. In: *Computational Economics* 43 (2014), 04

[35] Kumar, Rohit: Analysis of shape alignment using Euclidean and Manhattan distance metrics. In: *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)* IEEE (Veranst.), 2017

[36] Lewis, Roger J.: An introduction to classification and regression tree (CART) analysis. In: *Annual meeting of the society for academic emergency medicine in San Francisco, California* Bd. 14 Citeseer (Veranst.), 2000

[37] Liashchynskyi, Petro ; Liashchynskyi, Pavlo: Grid search, random search, genetic algorithm: a big comparison for NAS. In: *arXiv preprint arXiv:1912.06059* (2019)

[38] Lin, Jimmy ; Zhong, Chudi ; Hu, Diane ; Rudin, Cynthia ; Seltzer, Margo: Generalized and Scalable Optimal Sparse Decision Trees. In: III, Hal D. (Hrsg.) ; Singh, Aarti (Hrsg.): *Proceedings of the 37th International Conference on Machine Learning* Bd. 119, PMLR, 13–18 Jul 2020. – URL https://proceedings.mlr.press/v119/lin20g.html

[39] Lindholm, Andreas ; Wahlström, Niklas ; Lindsten, Fredrik ; Schön, Thomas B.: Su-

pervised machine learning. In: *Department of Information Technology, Uppsala University: Uppsala, Sweden* (2019)

[40] Liu, Bing ; Xia, Yiyuan ; Yu, Philip S.: Clustering through decision tree construction. In: *Proceedings of the ninth international conference on Information and knowledge management*, 2000

[41] Lopes, Cristina: A Comparison Between Optimization Tools to Solve Sectorization Problem.

[42] Lundberg, Scott M. ; Nair, Bala ; Vavilala, Monica S. ; Horibe, Mayumi ; Eisses, Michael J. ; Adams, Trevor ; Liston, David E. ; Low, Daniel King-Wai ; Newman, Shu-Fang ; Kim, Jerry u. a.: Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. In: *Nature Biomedical Engineering* 2 (2018), Nr. 10

[43] Mantovani, Rafael G. ; Horváth, Tomáš ; Cerri, Ricardo ; Junior, Sylvio B. ; Vanschoren, Joaquin ; Carvalho, André Carlos Ponce de Leon F. de: An empirical study on hyperparameter tuning of decision trees. In: *arXiv preprint arXiv:1812.02207* (2018)

[44] Markus, Aniek F. ; Kors, Jan A. ; Rijnbeek, Peter R.: The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. In: *Journal of Biomedical Informatics* 113 (2021). – URL `https://www.sciencedirect.com/science/article/pii/S1532046420302835`. – ISSN 1532-0464

[45] Marler, R T. ; Arora, Jasbir S.: The weighted sum method for multi-objective optimization: new insights. In: *Structural and multidisciplinary optimization* 41 (2010)

[46] Marler, R.T. ; Arora, J.S.: *Survey of multi-objective optimization methods for engineering*. 2004. – URL `https://doi.org/10.1007/s00158-003-0368-6`

[47] Merenda, Massimo ; Porcaro, Carlo ; Iero, Demetrio: Edge Machine Learning for AI-Enabled IoT Devices: A Review. In: *Sensors* 20 (2020), 04

[48] Mishra, Shashi ; Chakraborty, Suvra ; Samei, Mohammad ; Ram, Bhagwat: A q-Polak-Ribière-Polyak conjugate gradient algorithm for unconstrained optimization problems. In: *Journal of Inequalities and Applications* 2021 (2021), 01

[49] Mothilal, Ramaravind K. ; Sharma, Amit ; Tan, Chenhao: Explaining machine learning classifiers through diverse counterfactual explanations. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020

[50] Munzner, Tamara: *Visualization analysis and design*. CRC press, 2014

[51] Nissen, Volker: *Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution*. Springer-Verlag, 2013

[52] Palaparthi, Anil ; Riede, Tobias ; Titze, Ingo: Combining Multi-objective Optimization and Cluster Analysis to Study Vocal Fold Functional Morphology. In: *IEEE transactions on bio-medical engineering* 61 (2014), 04

[53] Papageorgiou, Markos ; Leibold, Marion ; Buss, Martin: *Optimierung*. Bd. 4. Springer, 2015

[54] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[55] Phung ; Rhee: A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. In: *Applied Sciences* 9 (2019), 10

[56] Pieper, Martin: *Mathematische Optimierung*. Springer, 2017

[57] Reynoso-Meza, Gilberto ; Blasco, X. ; Sanchís, Javier ; Martínez, Miguel A.: Controller tuning using evolutionary multi-objective optimisation: Current trends and applications. In: *Control Engineering Practice* 28 (2014)

[58] Ribeiro, Marco T. ; Singh, Sameer ; Guestrin, Carlos: "Why should i trust you?" Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016

[59] Richter, Stefan: Statistisches und maschinelles Lernen. In: *Berlin, Heidelberg: Springer Berlin Heidelberg. DOI* 10 (2019)

[60] Rodrigo Monteiro, Wellington ; Reynoso-Meza, Gilberto: Counterfactual Generation Through Multi-objective Constrained Optimisation. (February 2022)

[61] Rodrigo Monteiro, Wellington ; Reynoso-Meza, Gilberto: Assessing the Predictions of Machine Learning Algorithms in an Industrial Application Through Counterfactual Generation. (June 2021)

[62] Rodrigo Monteiro, Wellington ; Reynoso-Meza, Gilberto: Explaining black-box classification and regression models with counterfactuals using multi-objective constrained optimization. (October 2020)

[63] Romashov, Piotr ; Gjoreski, Martin ; Sokol, Kacper ; Martinez, Maria V. ; Langheinrich, Marc: BayCon: Model-agnostic Bayesian Counterfactual Generator IJCAI (Veranst.), 2022

[64] S. Boyd, M. P. ; Vandenberghe, L.: Subgradients. In: *Notes for EE364b, Stanford*

University, Spring 2021-22 (2022)

[65] Salaschek, Martin ; Thielsch, Meinald T.: Toolbox zur kontinuierlichen Website-Evaluation Qualitätssicherung. In: Hess, Steffen (Hrsg.) ; Fischer, Holger (Hrsg.): *Mensch und Computer 2017 - Usability Professionals*. Regensburg : Gesellschaft für Informatik e.V., 2017

[66] Santiago, Facundo ; https://medium.com/ (Hrsg.): *Model interpretability — Making your model confesses: Surrogate models.* May 2020. – URL `https://santiagof.medium.com/model-interpretability-making-your-model-confesses-surrogate-models-3dbf72bee8e`. – [Online; posted 12-May-2020]

[67] Schaffranietz, Klaus ; Neumann, Fritz: *Wissensgenerierung aus Datenbanken.* In: Keuper, Frank (Hrsg.) ; Neumann, Fritz (Hrsg.): *Wissens- und Informationsmanagement: Strategien, Organisation und Prozesse.* Wiesbaden : Gabler Verlag, 2009. – URL `https://doi.org/10.1007/978-3-8349-6509-7_7`. – ISBN 978-3-8349-6509-7

[68] Schmied, Sabine: *Stochastische Losungsansatze zu mehrdimensionalen nichtlinearen Optimierungsproblemen*, München, Univ. der Bundeswehr, Diss., 2009, Dissertation, 2009

[69] Scholz, Daniel ; Scholz, Daniel: Multikriterielle Optimierung. In: *Optimierung interaktiv: Grundlagen verstehen, Modelle erforschen und Verfahren anwenden* (2018)

[70] Shikhman, Vladimir ; Shikhman, Vladimir: Newton-Verfahren. In: *Mathematik für Wirtschaftswissenschaftler: In 60 fachübergreifenden Vorlesungen präsentiert* (2019)

[71] Singh, Chandan ; Nasseri, Keyan ; Tan, Yan S. ; Tang, Tiffany ; Yu, Bin: *imodels: a python package for fitting interpretable models.* – URL `https://doi.org/10.21105/joss.03192`

[72] Swapna, S ; Niranjan, P ; Srinivas, B ; Swapna, R: Data cleaning for data quality. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* IEEE (Veranst.), 2016

[73] Verma, Sahil ; Boonsanong, Varich ; Hoang, Minh ; Hines, Keegan E. ; Dickerson, John P. ; Shah, Chirag: Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review. In: *arXiv preprint arXiv:2010.10596* (2020)

[74] Verordnung (EU) 2016/679: *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung) (Text von Bedeutung für den EWR).* 2016

[75] Wachter, Sandra ; Mittelstadt, Brent ; Russell, Chris: Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. (March 2018)

[76] Wang, Qi ; Ma, Yue ; Zhao, Kun ; Tian, Yingjie: A comprehensive survey of loss functions in machine learning. In: *Annals of Data Science* (2020)

[77] Weise, Thomas: *Global Optimization Algorithm: Theory and Application.* 01 2009

[78] Wexler, James ; Pushkarna, Mahima ; Bolukbasi, Tolga ; Wattenberg, Martin ; Viégas, Fernanda ; Wilson, Jimbo: The what-if tool: Interactive probing of machine learning models. In: *IEEE transactions on visualization and computer graphics* 26 (2019), Nr. 1

[79] Yao, Quanming ; Yang, Hansi ; Hu, En-Liang ; Kwok, James: Efficient Low-Rank Semidefinite Programming with Robust Loss Functions. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (2021), 06

[80] Zhang, Baochang: *Machine Learning and Visual Perception.* Berlin, Boston : De Gruyter, 2020. – URL `https://doi.org/10.1515/9783110595567`. – ISBN 9783110595567

[81] Zöller, Marc-Andre ; Huber, Marco F.: Explanation Framework for Intrusion Detection. (December 2020). ISBN 978-3-662-62745-7

[82] Zöller, Marc-Andre ; Huber, Marco F.: Benchmark and Survey of Automated Machine Learning Frameworks. In: *Journal of Artificial Intelligence Research* (January 2021)

Appendix

Anhang A:

Codebeispiele

```
def getModelForFile(file, algorithm, pruning, train_size, save):
  filename = "model_" + algorithm + "_" + pruning + "_" + str(train_size)
  df = getdataset(file)

  #Vorbereitung von Trainings- und Testdaten
  X_train, X_test, y_train, y_test, cat_columns, num_columns, cat_indexes,
  num_indexes = prepareTrainAndTestData(df, train_size)
  #Laden des Modells, falls bereits entwickelt
  dt = loadModel(filename)
  if (dt == 0):
      #Laden des Basis-Modells
      dt = getModel(algorithm, pruning, train_size, False)
      #Anwendung von Hyperparameter-Tuning (Random-Search) mit festen Parametern
      if (pruning == 'PRE-PRUNING'):
          grid_param = {"criterion": ["gini", "entropy"],
                        "splitter": ["best", "random"],
                        "max_depth": range(2, 50, 1),
                        "min_samples_leaf": range(1, 15, 1),
                        "min_samples_split": range(2, 20, 1)}
          grid_search = RandomizedSearchCV(DecisionTreeClassifier(), grid_param,
          n_iter=10, cv=9)
          grid_search.fit(X_train, y_train)
          dt = DecisionTreeClassifier(
              criterion=grid_search.best_params_['criterion'],
              max_depth=grid_search.best_params_['max_depth'],
              min_samples_leaf=grid_search.best_params_['min_samples_leaf']
              min_samples_split=grid_search.best_params_['min_samples_split']
              splitter=grid_search.best_params_['splitter'])
      #Anwendung von Kostenkomplexitäts-Pruning
      if (pruning == 'POST-PRUNING'):
          path = dt.cost_complexity_pruning_path(X_train, y_train)
          ccp_alphas, impurities = path.ccp_alphas, path.impurities
          clfs = []
          score = 0
```

```
35          for ccp_alpha in ccp_alphas:
36              clf = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
37              clf.fit(X_train, y_train)
38              clfs.append(clf)
39              newscore = accuracy_score(y_test, clf.predict(X_test))
40              if (newscore > score):
41                  dt = clf
42                  score = newscore
43  if save :
44      saveModelForFile(dt, file, filename)
45  return dt
```

Listing 1: Generierung eines Modells

```
def getcounterfactual(regression, label_encoders, X_test,
                      model, X_current, y_desired,immutable_column_indexes,
                      y_acceptable_range, upper_bounds, lower_bounds,
                      categorical_columns, num_indexes, max_changed_vars) :
  try:
      #Vorbereitung für Regressionsaufgabe
      if(regression):
          front, X_generated, algorithms = generate_counterfactuals_regression(
          model, X_current, y_desired, immutable_column_indexes,
          y_acceptable_range, upper_bounds, lower_bounds, categorical_columns,
          num_indexes,pop_size=None,max_changed_vars=max_changed_vars, n_gen=30,
          seed=0, prob_mating=0.7,verbose=False,method_name='predict_proba',
          n_jobs=-1)
      #Kontrafaktische Generierung auf Basis der Klassifikation
      else :
          front, X_generated, algorithms =
          generate_counterfactuals_classification_proba(model, X_current,
          y_desired, immutable_column_indexes, y_acceptable_range,
          upper_bounds, lower_bounds, categorical_columns, num_indexes,
          pop_size=None,  max_changed_vars=max_changed_vars, n_gen=30,
          seed=0, prob_mating=0.7,verbose=True,method_name='predict_proba',
          n_jobs=-1)
      df_xmoai = pd.DataFrame(X_generated.copy(), columns=X_test.columns)
      for col in label_encoders.keys():
          df_xmoai[col] = label_encoders[col]
          .inverse_transform(df_xmoai[col].astype(int))
      return df_xmoai
  except Exception:
      return pd.DataFrame()
```

Listing 2: Generieren von Kontrafaktualen

```
"name": "serum_creatinine <= 1.45",
"value": "serum_creatinine <= 1.449999988079071",
"impurity": 0.25409006577837745,
"samples": 154,
"left": 0,
"children": [
    {
        "name": "ejection_fraction <= 32.50",
        "value": "ejection_fraction <= 32.5",
        "impurity": 0.47750229568411384,
        "samples": 33,
        "left": 0,
        "children": [
            {
                "name": "0",
                "p": "13.0",
                "samples": 16,
                "impurity": 0.3046875,
                "left": 0
            },{
                "name": "1",
                "p": "10.0",
                "samples": 17,
                "impurity": 0.4844290657439446,
                "left": 1
            }
        ]
    },{
        "name": "0",
        "p": "111.0",
        "samples": 121,
        "impurity": 0.15162898709104566,
        "left": 1
    }
```

Listing 3: Entscheidungsregeln als JSON zum Datensatz *Klinische Aufzeichnungen zu Herzinsuffizienz* (Auszug)