Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Entwurf vom
24. August 2022

Masterarbeit

Transparenzschaffende Maßnahmen für Verfahren des maschinellen Lernens

vorgelegt von

Lynn Oesterwind
Matrikelnummer 6919328
Studiengang Wirtschaftsinformatik

eingereicht am 24. August 2022

Betreuer: Joshua Stock
Erstgutachter: Prof. Dr.-Ing. Hannes Federrath
Zweitgutachter: Dr.-Ing. Daniel Demmler

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Forschungsziel und Umfang . . . 1

2 Theoretische Grundlagen **2**
2.1 Maschinelles Lernen . . . 2
2.1.1 Begriffserklärung . . . 2
2.1.2 Ablauf des maschinellen Lernens . . . 3
2.1.3 Typen des maschinellen Lernens . . . 4
2.1.4 Verfahren des maschinellen Lernens . . . 6
2.1.4.1 Entscheidungsbäume . . . 7
2.1.4.2 k-means Clustering . . . 7
2.1.4.3 Support Vector Machines . . . 8
2.1.4.4 Logistische Regression . . . 9
2.1.4.5 k-Nächste Nachbarn . . . 9
2.1.4.6 Bayessche Netze . . . 10
2.1.4.7 Künstliche neuronale Netze . . . 12
2.1.5 Erklärbares maschinelles Lernen . . . 14
2.1.5.1 Erkläransätze . . . 14
2.1.5.2 Erklärungen . . . 15
2.2 Transparenz . . . 16
2.2.1 Definition . . . 16
2.2.2 Relevanz von Transparenz bei Algorithmen des maschinellen Lernens . . . 17
2.3 Dokumentation von IT-Systemen . . . 18

3 Transparenzschaffende Maßnahmen aus der Literatur **19**
3.1 Vorgehen der Literaturrecherche . . . 19
3.1.1 Definition des Umfangs der Literaturrecherche . . . 20
3.1.2 Konzeptualisierung des Themas . . . 21
3.1.3 Literaturrecherche . . . 22
3.1.4 Literaturanalyse und -synthese . . . 23
3.1.5 Forschungsagenda . . . 24
3.2 Ergebnisse der Literaturrecherche . . . 24
3.2.1 Konzepte . . . 24
3.2.1.1 Transparentes ML . . . 24
3.2.1.2 Mensch und Maschine . . . 25
3.2.1.3 Mentale Modelle . . . 25
3.2.1.4 Facetten der Transparenz . . . 25
3.2.2 Was transparent machen? . . . 26
3.2.2.1 Inhaltlicher Kontext . . . 26
3.2.2.2 Rahmenbedingungen und sozio-technischer Kontext . . . 26

3.2.3 Daten . . . 28
3.2.3.1 Bias . . . 28
3.2.3.2 Dokumentation der Daten . . . 29
3.2.3.3 Dokumentation der Datenverarbeitung . . . 30
3.2.3.4 Transparenz über gleiche Datenformate . . . 30
3.2.4 Transparenz durch das gewählte Modell . . . 30
3.2.4.1 Transparente Modelle . . . 31
3.2.5 Modellauswahl . . . 32
3.2.6 Erklärungen . . . 32
3.2.7 Einordnung von XAI-Methoden . . . 32
3.2.8 XAI-Methoden für transparente Methoden . . . 33
3.2.8.1 Lineare Regression . . . 33
3.2.8.2 Entscheidungsbäume . . . 33
3.2.8.3 Random Forest . . . 33
3.2.8.4 Sonstiges . . . 33
3.2.9 Modellagnostische Methoden - lokal . . . 34
3.2.9.1 LIME . . . 34
3.2.9.2 SHAP . . . 36
3.2.9.3 Anchors . . . 38
3.2.9.4 Kontrafaktische Erklärungen . . . 40
3.2.9.5 Konzept . . . 40
3.2.10 Modellagnostische Methoden - global . . . 41
3.2.10.1 Partial Dependence Plot . . . 41
3.2.10.2 Individual Conditional Expectation (ICE) . . . 42
3.2.10.3 Global Surrogate . . . 42
3.2.10.4 Protoypen und Kritiken . . . 43
3.2.11 Modellspezifisch: Neuronale Netze . . . 43
3.2.11.1 Layer-wise Relevance Propagation . . . 44
3.2.11.2 Saliency Maps . . . 44
3.2.11.3 TCAV . . . 45
3.2.12 Weitere Methoden . . . 46
3.2.13 Präsentation von Erklärungen . . . 47
3.2.13.1 Integration von Domänenwissen und Kontext . . . 47
3.2.13.2 Form der Präsentation . . . 47
3.2.13.3 Interaktivität . . . 48
3.2.14 XAI-Plattformen . . . 50
3.2.15 Vorgehen bei Entwicklung . . . 51
3.2.15.1 Vorgehensmodell für transparentes ML . . . 51
3.2.15.2 Transparenz während der Implementation . . . 52
3.2.15.3 AutoML . . . 53
3.2.15.4 Verteiltes Lernen . . . 53
3.2.16 Dokumentation . . . 54
3.2.16.1 Dokumentation des ML-Algorithmus . . . 54
3.2.16.2 Dokumentation der Modellauswahl . . . 55
3.2.16.3 Ganzheitliche Dokumentation . . . 55
3.2.17 Probleme mit Transparenz . . . 56
3.2.17.1 Offene Fragen bei Transparenz . . . 57
3.2.17.2 Zeitaufwand . . . 57

3.2.17.3 Gefahren durch Transparenz 57

Literatur **60**

Todo list **74**

Aufgabenstellung

Nur Studien-, Bachelor-, Master- und Diplomarbeiten: Soweit eine ausformulierte Aufgabenstellung mit der Betreuerin bzw. dem Betreuer vereinbart wurde, diese bitte hier einfügen.

Da der Pipeline-Begriff schon vergeben ist (siehe z.B. z.B. [KKH22] oder https://docs.microsoft.com/de-de/azure/machine-learning/concept-ml-pipelines), hier daran denken, diesen in Datenfluss umzuändern :)

Zusammenfassung

Für die eilige Leserin bzw. den eiligen Leser sollen auf etwa einer halben, maximal einer Seite die wichtigsten Inhalte, Erkenntnisse, Neuerungen bzw. Ergebnisse der Arbeit beschrieben werden.

Durch eine solche Zusammenfassung (im Engl. auch Abstract genannt) am Anfang der Arbeit wird die Arbeit deutlich aufgewertet. Hier sollte vermittelt werden, warum man die Arbeit lesen sollte.

1 Einleitung

1.1 Motivation

1.2 Forschungsziel und Umfang

Auf Grundlage der Aufgabenstellung wurden folgende Forschungsfragen formuliert, welche mithilfe der Masterarbeit beantwortet werden sollen:

- *RQ1*: Wie können die unterschiedlichen Ebenen einer ML-Pipeline transparent gemacht werden?
- *RQ2*: Welche Ansätze existieren, um verschiedene ML-Algorithmen, wie z.B. Entscheidungsbäume oder auch neuronale Netzwerke zu erklären?
- *RQ3*: Inwieweit und auf welche Weise sollten ML-Algorithmen transparent gemacht werden, wenn diese Experten oder fachfremden Personen veranschaulicht werden?

2 Theoretische Grundlagen

Das vorliegende Kapitel führt in die theoretischen Grundlagen des MLs ein und stellt die Grundzüge der Transparenz von IT-Systemen dar.

2.1 Maschinelles Lernen

ML hat in den letzten Jahren Einzug in viele Bereiche des persönlichen und professionellen Lebens genommen. So wird es beispielsweise im Marketing und Vertrieb, aber auch in der Medizin eingesetzt. Mithilfe selbstlernender Algorithmen können beispielsweise Persönlichkeitsprofile erstellt oder Krankheiten erkannt werden [Str19; Wut22]. Es bestehen vielfältige Anwendugnsmöglichkeiten in nahezu allen Branchen. Technisch können ML-Algorithmen im Bereich der Computer Vision eingesetzt werden, um z.B. Objekte auf Bildern zu erkennen. Zusätzlich können Vorhersagen getroffen werden. Überdies sind ML-Algorithmen unter anderem in der Lage aus Texten Informationen zu ziehen oder diese sogar zu übersetzen [SS18].

Folgendes Kapitel bietet einen Überblick über ML, indem zunächst das Feld ML abgegrenzt wird. Darauffolgend wird auf die verschiedenen Lerntypen eingegangen und die Funktionsweise konkreter Algorithmen dargelegt.

2.1.1 Begriffserklärung

Im Zusammenhang mit dem Begriff des „Maschinellen Lernens“ tauchen häufig auch Konzepte wie „Künstliche Intelligenz“ (engl. Artifical Intelligence, Abk.: KI/AI), „Neuronale Netze“ oder „Deep Learning“ (DL) auf. Überdies werden diese Begriffe teilweise synonym verwendet, was jedoch nicht korrekt ist, obgleich eine gewisse Verwandtschaft besteht [Ker+20; SS18]. Um in Bezug auf die vorliegende Masterarbeit Klarheit zu schaffen, startet dieses Kapitel mit einer Abgrenzung dieser verschiedenen Begrifflichkeiten. Der Zusammenhang ist grob in Abbildung 2.1 dargestellt.

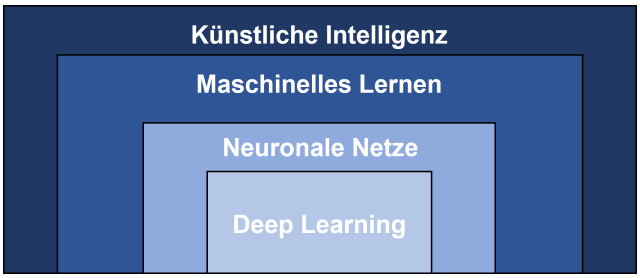


Abbildung 2.1: Feld des MLs, eigene Darstellung in Anlehnung an [SS18]

KI lässt sich definieren als die Befähigung „einer Maschine, menschliche Fähigkeiten wie logisches Denken, Lernen, Planen und Kreativität zu imitieren“ [Par20]. Das ML dagegen ist ein Teilgebiet der KI und hat den automatisierten Lernprozess von Maschinen zum Gegenstand [Ker+20]. Mithilfe von ML ist es möglich, große Datenmengen zu verarbeiten und aus diesen

Entscheidungen oder Erkenntnisse abzuleiten. Abstrakter formuliert, kann ML als Berechnungsprozess verstanden werden, welcher Eingabedaten verwendet, ohne dass durch Programmcode exakt vorgegeben ist, wie schlussendlich das Ergebnis gebildet wird [EM15]. Neuronale Netze bilden wiederum ein Teilgebiet von ML [Ker+20]. Künstliche neuronale Netze bilden computerbasiert die Gehirnstrukturen von Menschen und Tieren nach, indem einzelne Knoten über Kanten verknüpft werden [Roj01]. Neben den neuronalen Netzen existiert noch der Begriff des DLs. Deep Learning-Modelle sind neuronale Netze mit tieferen Strukturen, jedoch ist umstritten wie viele Schichten ein künstliches neuronales Netz zu einem „Deep Neuronal Network“ (DNN) machen, und wie somit eine präzise Abgrenzung der Begriffe möglich ist [Ker+20].

Die folgende Masterarbeit konzertiert sich auf das Forschungsfeld des MLs. Während der Literaturrecherche werden aber auch Quellen, welche KI, neuronalen Netzen oder Deep Learning zum Gegenstand haben, nicht ausgeschlossen, da diese wie gezeigt mit ML eng vermascht sind.

2.1.2 Ablauf des maschinellen Lernens

Der Ablauf beim ML kann auf unterschiedlichen Abstraktionsebenen beschrieben werden. Auf hohger Abstraktionsebene lässt sich der technische Ablauf vereinfacht in Abbildung 2.2 darstellen. Im Kontext dieser Masterarbeit wird der technische Ablauf des MLs von hier an als *Datenfluss* bezeichnet. Ausgangspunkt für diesen Datenfluss sind Beispieldaten, welche die Grundlage für einen Algorithmus bieten, Muster und Zusammenhänge zu erkennen. Bei diesen Beispieldaten wird auch von Trainingsdaten gesprochen [JZH21]. Um diese Daten für den Algorithmus effizient nutzbar zu machen, beginnt der Datenfluss mit der *Feature Extraction*. Hier werden relevante Merkmale aus den Daten ausgewählt, zusammengefasst oder in ein Format überführt, welches für Maschinen leichter zu interpretieren ist [JZH21]. Ein Beispiel für Feature Extraction ist im Umfeld der Textklassifikation die TF-IDF-Methode [DZ11]. TF-IDF steht für *term frequency-inverse document frequency* und diese gibt an, wie wichtig ein Wort in einem Text ist. Hier wird für ein Wort ein numerischer Wert zurückgegeben, mit dem ein Algorithmus besser umgehen kann. Je häufiger ein Wort vorkommt, desto höher ist der TF-IDF-Wert, wobei dieser Wert durch das Betrachten anderer Texte in einem gesamten Korpus gesenkt werden kann, um zu vermeiden, dass Wörter, welche grundsätzlich oft vorkommen, eine fälschlicherweise hohe Wichtigkeit zugesprochen bekommen [BPV16]. Haben die Trainingsdaten die Feature-Extraction-Phase des Datenflusses passiert, beginnt der Prozess des "Model Buildings", welcher auch Training genannt wird [Cad17; BÖ14]. Ergebnis des Trainings ist ein Modell, welches Zusammenhänge innerhalb der Trainingsdaten beschreibt. Das erstelle Modell kann z.B. ein Entscheidungsbaum oder ein neuronales Netzwerk sein (vgl. Kapitel 2.1.4 [CL20]. Schlussendlich soll das erstellte Modell den zweck erfüllen, Vorhersagen zu treffen, wie z.B. das Klassifizieren von unbekannten Datensätzen [Wut22].

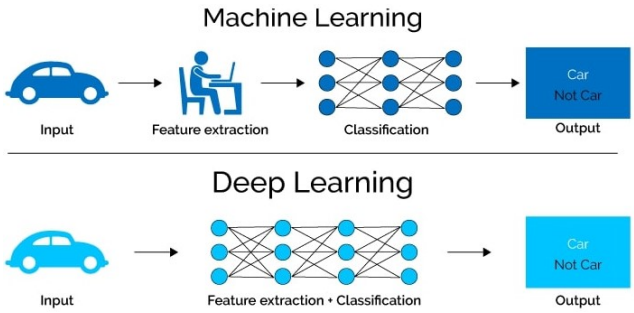


Abbildung 2.2: Prozess des MLs/DLs, entnommen aus [Wol22]

Neben dem eher technisch orientierten Datenfluss, können prozessual allerdings noch weitere organisatorische und inhaltliche Aspekte in den Vordergrund rücken. Grundsätzlich muss zu Beginn die Art des Problems identifiziert werden [Ver20]. Weiter muss sich für einen konkreten Algorithmus (vgl. Kapitel 2.1.4 entschieden werden [Ayo10]. Organisatorisch steht besonders bei ML-Projekten im Vordergrund, ob, wo und in welchem Format Trainingsdaten vorliegen. Weiter müssen diese Daten unter Umständen noch bereinigt oder auf ihre notwendigen Datenfelder herunter gebrochen werden [Ver20]. Auch ist es möglich, dass Datensätze künstlich expandiert werden, was beispielsweise in der Bilderkennung häufig vorkommt. Hier werden Bilder aus den Trainingsdaten rotiert oder gespiegelt, um mehr Trainingsdaten für den Algorithmus zu generieren um somit z.B. Überanpassung zu vermeiden [MG18]. Nach der Feature Extraction und dem Training muss sich auch noch über die Auslieferung des Modells in den produktiven Betrieb Gedanken gemacht werden [Ver20]. Weiter kann während des Betriebs auch das sog "Liefelong Learning"berücksichtigt werden. Dieses Lernparadigma zählt darauf ab, dass ein Algorithmus neues lernt, während er dabei auf bestehenden Wissen aufbaut [CML18].

Ein Kapitel zu Über- und Unteranpassung und Bias?

Bei der Herausforderung inhaltliche und organisatorische Ebenen von ML-Projekten zu überblicken helfen in der Praxis Vorgehensmodelle. Bekannte Beispiele hierfür sind der *Cross-Industry Standard Process for Data-Mining* (CRISP-DM) und der *Knowledge Discovery in Databases*-Prozess (KDD) [Web+19]. Das Prozessmodell CRISP-DM wurde von dem amerikanischen IT-Unternehmen *IBM* entwickelt. CRISP-DM bezieht in mehreren Phasen unter anderem das Schaffen von Verständnis für die Domäne und die Daten mit ein. Ferner legt es Vorgaben für die Datenvorbereitung, Modellierung und Evaluation des Modells fest. Daneben wird auch der produktive Einsatz des generierten ML-Modells mit betrachtet [IBM]. Das KDD-Prozess kann auch Anwendung bei ML-Projekten finden, setzt hierbei allerdings einen eher technischen Fokus [Web+19; FPS96]. Dieses Modell legt in den ersten vier Phasen den Fokus auf die Datenauswahl, -vorbereitung, -transformation und Mustererkennung. Die letzte Phase beinhaltet die Interpretation und das Schaffen von Wissen [FPS96]. Weber et al. [Web+19] merken jedoch an, dass diese Modelle trotz ihres vielfachen Einsatzes in der Praxis auch Schwächen aufweisen. Dies begründen die Autoren damit, dass die experimentelle und die operative Phase des MLs nicht genügend Aufmerksamkeit bekommt [Web+19]. Geschlossen wird diese Lücke vom Konzept des *Machine Learning Operations* (MLOps). Kreuzberger et al. [KKH22] stellen heraus, welche Prozesse und Prinzipien dabei helfen ML-Projekte ganzheitlich bis hin zum Betrieb zu betrachten. Hierbei werden neben technischen Kompontenten oder Rollen, wie z.B. die eines Software Engineers auch prozessuale Elemente wie Automatisierung mitbedacht [KKH22].

2.1.3 Typen des maschinellen Lernens

Einzelne Algorithmen des MLs lassen sich zumeist einer von mehreren Klassen zuordnen. Hauptunterscheidungsmerkmal dieser Kategorien besteht in der Art des Lernens sowie in der Beschaffenheit der Ein- und Ausgabedaten. Die drei am häufigsten genannten Klassen sind überwachtes Lernen (engl.: *supervised learning*), unüberwachtes Lernen (engl.: *unsupervised learning*) und bestärkendes Lernen (engl.: *reinforcement learning*), wobei auch noch Mischformen wie teilüberwachtes Lernen (engl.: *semi-supervised learning*) existieren [Alp19; Wut22; Ayo10].

Überwachtes Lernen

Überwachtes Lernen ist der am häufigsten genutzte Typ des maschinellen Lernens. Hierbei lernt ein Computerprogramm, indem es sich bestehende Daten zu Nutze macht [Ver20]. Diese zu Beginn vorhandenen Datensätze enthalten immer einen Input und einen Output [Alp19; Cho+20]. Ein Algorithmus soll lernen, wie Merkmale, welche sich in den Daten befinden, mit den Zielvariablen zusammenhängen [Wut22]. Es soll also eine Funktion $f : x \rightarrow y$ entstehen, welche folgende Input-Paare $\{(x_1, y_1), ..., (x_i, y_i)\}$ aufeinander abbildet. Schlussendlich sollen Zielvariablen (y-Werte) für neue, unbekannte x-Werte mit der Funktion vorhergesagt werden [Cho+20].

Typisch für Probleme, welche mithilfe von überwachtem Lernen angegangen werden, sind Klassifikations- und Regressionsprobleme. Klassifikationsalgorithmen beispielweise ermöglichen es auf Bildern zu erkennen, ob es sich um eine Katze oder einen Hund handelt. Die im vorherigen Abschnitt angesprochene Zielvariable y wäre in diesem Fall ein Wert, welcher einer Kategorie, also Hund oder Katze, entspräche. Ein Beispiel für ein Regressionsproblem ist die Vorhersage von Regentagen auf Grundlage historischer Niderschlagsmengen [Cho+20].

Später überdenken: eigenes Kapitel für Anwendungsfälle sinnvoll?

Unüberwachtes Lernen

Im Gegensatz zum überwachten Lernen stehen beim unüberwachtem Lernen keine Trainingsdaten mit Zielvariablen zur Verfügung [Ayo10]. Das Ziel bleibt jedoch, dass eine Funktion erstellt werden soll, die Eingabe- auf Ausgabewerte $f : x \rightarrow y$ abbildet, jedoch ohne Kenntnisse über y [Alp19]. Algorithmen des unüberwachten Lernens funktionieren so, dass sie den Daten Informationen extrahieren und folglich Muster in den Daten entdecken [Cho+20; Ver20].

Beispiele für die Anwendung von unüberwachtem Lernen sind Clusteranalysen oder Assoziationsalanysen [Wut22]. Bei einer Clusteranalyse werden Daten aufgrund ähnlicher Muster oder gleicher Attribute gruppiert. Muster können dadurch entstehen, dass z.B. bei Entitäten eines Clusters die gleichen Merkmale vorhanden sind oder fehlen. Als praxisnaher Anwendungsfall sei hier die Kundensegmentierung genannt, welche es einem Unternehmen ermöglicht verschiedene Marketingstrategien passend für unterschiedliche Kundensegemente zu entwickeln. Merkmale, die im Rahmen einer Clusteranalyse ins Gewicht fallen, wären z.B. das Alter, der Umsatz oder das Online/Offline-Verhalten [Ver20]. Assoziationsanalyse als weiteres Beispiel des unüberwachten Lernens haben zum Gegenstand Assoziationsregeln zu finden, die Zusammenhänge zwischen verschiedenen Attributen darstellen, wie es z.B. in der Warenkorbanalyse der Fall ist. Hier wird versucht, anhand von bestimmten im Warenkorb befindlichen Produkten auf andere Produkte zu schließen [CL20]. Eine Regel, welche ein Algorithmus der Assoziatitonsanalyse aufstellen würde, könnte beispielsweise lauten: *Wer Müsli kauft, der kauft auch Milch*.

Teilüberwachtes Lernen

Das teilüberwachte Lernen bildet die Mischform des über- und unüberwachten Lernens, denn hier werden sowohl Daten mit Zielvariablen als auch Daten ohne Zielvariablen zum Trainieren des Algorithmus verwendet [Ayo10]. Teilüberwachtes Lernen ist meistens dann sinnvoll, wenn einige Daten mit Zielvariable, jedoch gleichzeitig auch viele ohne Zielvariable vorhanden sind und es sehr teuer oder aufwendig wäre, die fehlenden Zielvariablen zu erheben [Wut22]. Die Grundidee des teilüberwachten Lernens ist, dass Daten, welche sich im gleichen Cluster befinden, wahrscheinlich auch zur gleichen Klasse gehören. Diese

Erkenntnis kann genutzt werden, um eine bessere Datenbasis zu generieren. Somit können beide ML-Ansätze kombiniert werden [Ver20].

Bestärkendes Lernen

Das bestärkende Lernen beschäftigt sich damit, wie mehrere Aktionen aufeinander folgen sollen. Ein Algorithmus lernt basierend auf Belohnungen oder Bestrafungen, welche durch seine Umwelt signalisiert werden, wie er zu handeln hat [Lor20]. Dieses Feedback ist zwingend erforderlich, um selbstständig lernen zu können [Wut22]. Ziel ist es mit zukünftigen Aktionsreihenfolgen die Belohnungen zu maximieren [Lor20]. Weiter ist nicht eine einzige Aktionen, sondern mehrere aufeinander folgende Aktionen zusammengefasst als Taktik, entscheidend für die angemessene Funktionsweise des Algorithmus [Alp19].

Ein prominentes Beispiel für bestärkendes Lernen ist die von Google entwickelte Software *AlphaGO*, welche mittels Datensammlungen über von Menschen gespielter Partien, geeignete Taktiken entwickeln konnte, um gegen den erfolgreichsten menschlichen Spieler im Brettspiel GO gewinnen zu können [Wut22].

Je nach dem, welcher Typ zur Anwendung kommt, entstehen Differenzen bzgl. des technischen Ablauf des MLs, welcher im vorherigen Kapitel 2.1.2 dargestellt wurde. Beim überwachten Lernen wird dem Lernprozess die Teilung der Beispieldaten vorangestellt. Der Datensatz wird hier in einen Trainings- und ein Testdatensatz aufgeteilt. Diese Testdaten werden genutzt, um das erstellte Modell auf seine Funktionsweise und inhaltliche Zuverlässigkeit zu testen [KZP+07]. Es ist für die Funktionsweise des Algorithmus und die Performance während des Trainings jedoch nicht unerheblich, wie groß diese Datensätze im Verhältnis sind [Ngu+21]. Auch beim bestärkenden Lernen kommen Testdaten zum Einsatz [Lor20]. Da die Trainingsdaten, welche beim unüberwachten Lernen vorliegen, über keine Zielvariablen verfügen, kann die Überprüfung des erzeugten Modells nicht wie beim überwachten Lernen mithilfe von Testdaten erfolgen [Ayo10]. Alternativen, um das Modell bei diesen Algorithmen zu überprüfen, bestehen beispielsweise in Metriken, welche überprüfen, wie kompakt die verschiedenen Cluster erstellt wurden, oder in Kreuzvalidierung [HBV01; Per09].

An dieser Stelle sei angemerkt, dass neben der Funktionweise auch andere Qualitätsmerkmal bei ML eine Rolle spielen können. So kann die Performance oder Nutzung des Hardwarespeichers auch entscheidend bei der Implementierung eines ML-Algorithmus sein [Col+19]. Daneben existieren noch weitere Methoden zur Evaluation im ML, welche es z.B. ermöglichen, verschiedene Instanzen des gleichen Algorithmus oder auch unterschiedliche Algorithmen untereinander zu vergleichen [Ras18].

2.1.4 Verfahren des maschinellen Lernens

Es existieren viele konkrete Algorithmen des MLs, welche sich in eine oder mehrere der vorher genannten Typen einordnen lassen. Folgend werden die theoretischen Grundlagen der jeweiligen Algorithmen erläutert. Die in der Literatur und Praxis häufig genannten ML-Verfahren sind: Entscheidungsbäume, logistische Regression, Bayessche Modelle, Support Vector Machines, k-means Clustering, k-nearest Neighbours sowie neuronale Netze [ANK18; Mic22; Döb+18; Sar21]. Um den Umfang nicht zu sprengen, wird sich auf diese häufig genannten konzentriert.

Diese Auswahl am Ende noch einmal überdenken/anpassen

2.1.4.1 Entscheidungsbäume

Entscheidungsbäume können zur Klassifikation oder Regression eingesetzt werden [Wut22]. Entscheidungsbäume, welche zur Klassifikation eingesetzt werden, teilen eine Menge immer wieder in unterschiedliche Teilmengen auf. Die letzte Unterteilung entspricht der Klasse zu der eine Entität gehört [Bre+17]. Bei dieser schrittweisen Aufteilung spricht man auch von rekursiver Partitionierung [NS18]. Soll ein Entscheidungsbaum konstruiert werden, kann wie folgt vorgegangen werden: man starte zunächst mit der Wurzel, die ein erstes Entscheidungskriterium beinhaltet. Von dieser Wurzel aus können mehrere Kanten ausgehen, welche zu anderen Knoten führen. An diesen Knoten können sich wiederum Entscheidungskriterien mit neuen Kanten befinden. Knoten ohne Nachfolger bilden die Blätter und somit auch die Kategorien des zugrundeliegenden Klassifikationsproblems [Wut22]. Ein Beispiel für einen Entscheidungsbaum ist in Abbildung 2.3 zu sehen.

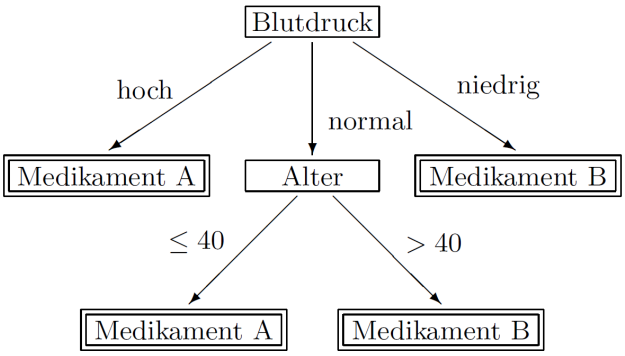


Abbildung 2.3: Beispiel für einen Entscheidungsbaum [BK98]

Entscheidungsbäume haben jedoch einige Nachteile, so können geringe Änderungen im Trainingsdatensatz große Auswirkungen im Design des Baumes nach sich ziehen. Überdies können unglücklich ausgewählte binäre Entscheidungskriterien am Baumknoten ungenaue Endergebnisse hervorrufen. Abhilfe kann eine mögliche Abwandlung von Entscheidungsbäumen schaffen, welche die Komplexität des Algorithmus jedoch erhöht. Dieses Verfahren heißt „Random Forest“, da dort verschiedene, zufällig erzeugte Entscheidungsbäume zur gleichen Zeit arbeiten. Am Ende werden über die Ergebnisse aller Bäume ein Mittelwert gebildet oder auf andere Weise zu einer endgültigen Antwort zusammengefasst [NS18].

2.1.4.2 k-means Clustering

Das *k-means Clustering*-Verfahren ist ein Algorithmus des unüberwachten Lernens und kann bspw. dazu eingesetzt werden gezielte Werbestrategien zu entwickeln [Mat21; NS18]. So können Kunden, die bestimmte Merkmale, z.B. Einkommen oder Persönlichkeitszüge wie Offenheit gemeinsam haben, die gleichen Produkte vorgeschlagen werden [NS18]. Das *k-means Clustering*-Verfahren unterstützt dies, indem es Kunden (oder universell formuliert Datenpunkte) einem von k Clustern zuordnet [Ayo10]. k muss bei diesem Verfahren vorgegeben werden. Im Fokus des Verfahrens steht die Ermittlung der Zentren (engl.: *Centroide*) eines jeden Clusters [CL20].

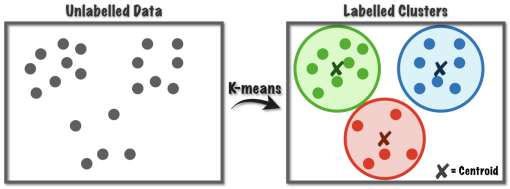


Abbildung 2.4: Input und Output des *k-means Clustering*-Verfahrens, entnommen aus [Fio20]

Die Funktionsweise des Verfahrens lässt sich in etwa wie folgt umreißen: zu Beginn werden (zufällig) die Standorte der k Zentren gewählt [CL20]. Als nächstes wird für jeden Datenpunkt das ihm am nächsten liegende Zentrum gefunden. Ist dies für jeden Datenpunkt erfolgt, werden neue Zentren ermittelt, indem die Mittelpunkte aller einem Zentrum zugehörigen Datenpunkte berechnet werden. Als nächstes folgt eine erneute Zuordnung aller Datenpunkte zu den neuen Zentren. Diese Zuordnung und Zentrenbildung wird so lange wiederholt bis sich keine neuen Zentren mehr auffinden lassen [NS18]. Abbildung 2.4 visualisiert das Endergebnis des *k-means Clustering*-Verfahrens beispielhaft.

k-means Clustering ist aufgrund seiner geringen Komplexität während der Implementierung beliebt. Vereinfacht gesagt, müssten bei einer Realisierung des Algorithmus nur immer wieder neue Abstände berechnet werden und draufhin die Datenpunkten (erneut) den Clusterzentren zugeordnet werden [CL20]. Neben dieser Einfachheit existieren jedoch auch einige Schwächen, wie z.B. die Form der Cluster. Durch die iterative Anwendung entstehen kreisförmige Cluster. In der Realität ist es jedoch nicht selten, dass die realen Cluster ein anderes, z.B. elliptisches Erscheinungsbild besitzen [NS18]. Weiter ist der Algorithmus anfällig in Bezug auf Rauschen und Ausreißer [CL20]. Als Ausreißer bezeichnet man Datenpunkte, welche sich von den anderen stark unterscheiden. Ein Beispiel hierfür könnten Kundendaten sein, welche das Alter der Personen beinhalten. Wären alle Kunden 20-30 Jahre alt, aber gäbe es einen Kunden, welcher 80 Jahre als ist, wäre dies der Ausreißer [CL20]. Rauschen in Daten kann sich in Form verschiedenster Anomalien bemerkbar machen. Beispielsweise kann es zu Messfehlern bei der Datenbeschaffung gekommen sein, wohingegen auch die Betrachtung zu vieler Attribute beim Lernprozess ein Rauschen verursachen kann [Alp19]. Daneben ist es nicht vorgesehen, dass sich Cluster gegenseitig überlappen [NS18]. Weiter sei angemerkt, dass die Ermittlung von k, also die Anzahl der Cluster, nicht durch das Verfahren gelöst wird und diese Entscheidung durch den Anwender getroffen werden muss [CL20].

2.1.4.3 Support Vector Machines

Support Vector Machines (SVM) können zur Klassifikation eingesetzt werden und gehören zu den Algorithmen des überwachten Lernens [Ver20]. Ziel des Algorithmus ist die Ermittlung einer Grenzlinie (engl.: *Hyperplane*), welche die Datenpunkte zweier oder mehrerer Klassen separiert [NS18; Nob06]. Im zwei-dimensionalen Raum kann man sich die Separierung der Daten als Gerade vorstellen [Ver20]. Zur Ermittlung der Grenzlinie ist es erforderlich „diejenigen Datenpunkte am Rand ihrer jeweiligen Gruppe zu finden, die am nächsten zu den Punkten der jeweils anderen Gruppe gelegen sind“ [NS18]. Diese Randpunkte werden als *Support-Vektoren* bezeichnet. Werden die Randpunkte der jeweiligen Klassen verbunden, entstehen zwei Linien. In der Mitte dieser zwei Linien, liegt nun die Hyperplane, was in Abbildung 2.5 visualisiert ist [NS18]. Der SVM-Algorithmus hat folglich zur Aufgabe die Abstände zwischen den *Support-Vektoren* zu maximieren, um die optimale Grenzlinie zu ermitteln, welche die Klassifikation unbekannter Datensätze ermöglicht [Mat21].

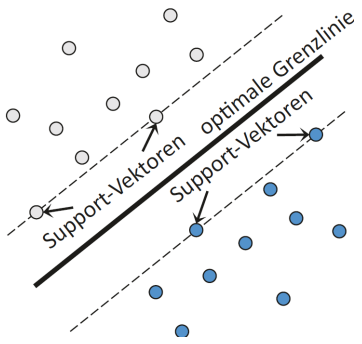


Abbildung 2.5: SVM [NS18]

SVM zeichnet ihre Flexibilität in der Separierung aus. Während in Kapitel 2.1.4.1 angemerkt wurde, dass das *k-means Clustering*-Verfahren die gefundenen Cluster nur kreisförmig darstellen kann, ermöglicht es die *Kernel*-Taktik hingegen mit SVM auch andere Formen darzustellen. Hierbei kommt es zu einer leichten Abwandlung des Verfahrens, sodass der Algorithmus in höheren Dimensionen angewendet wird und somit z.B. elliptische Abgrenzungen anstatt Geraden ermöglicht werden [NS18]. Weiter können auch Klassifikationen im n-dimensionalen Raum mithilfe einer Ebene, welche die Dimension $n-1$ besitzt, realisiert werden [CL20]. Ein weiterer Vorteil der SVM besteht in der Rigorosität gegenüber Ausreißern durch bestimmte Anpassungen des Verfahrens. So besteht die Möglichkeit Daten, die sowohl nah an der einen als auch an der anderen Gruppe liegen, lediglich mit gewissen Wahrscheinlichkeit einer Gruppe zuzuordnen. Diese Wahrscheinlichkeiten lassen sich durch die Betrachtung des Abstandes zur Grenzlinie berechnen [NS18].

2.1.4.4 Logistische Regression

Die logistische Regression verfolgt das Ziel die Wahrscheinlichkeit des Auftretens einer binären Zielgröße vorherzusagen. Diese Vorhersage kann abhängig von bestimmten Merkmalen getroffen werden. Ein beispielhaftes Szenario, mit dem sich logistische Regression beschäftigen könnte, besteht darin auf Grundlage von Geschlecht und Blutwerten vorherzusagen, wie wahrscheinlich es ist, dass eine Person erkrankt oder nicht erkrankt [KM21]. Anders als der Name des Verfahrens logistische „Regression“ vermuten lässt, geht es hier weniger um Regression, sondern eher um Klassifikation [KL19]. Zur Vorhersage der Wahrscheinlichkeiten wird eine logistische Funktion erstellt. Anders als es in der linearen Regression, in der versucht wird konkrete Werte mithilfe einer Geraden darzustellen, gefordert, ermöglicht es die logistische Funktion Wahrscheinlichkeiten abzubilden [Beh15]. Durch ihr s-förmiges Erscheinungsbild ist diese Funktion in der Lage „reelwertige Variablen“ $[+\infty, -\infty]$ in den Wertebereich von 0 bis 1 zu transformieren [Bac16]. Die entsprechenden Funktionsparameter der gewünschten logistischen Zielfunktion lassen sich mit der sogenannten *Maximum-Likelihood*-Methode finden [Bac16]. Diese Methode nähert sich iterativ den Parametern an und sucht diejenige Konstellation der Funktionsparameter, bei der es am wahrscheinlichsten ist, dass die im Trainingsdatensatz vorhandenen Beobachtungen auftreten [Beh15].

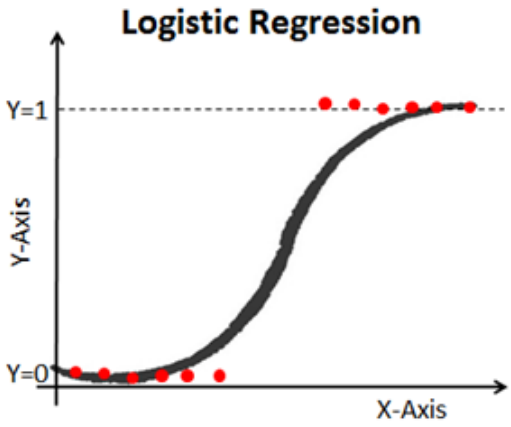


Abbildung 2.6: Logistische Regression [NS18]

2.1.4.5 k-Nächste Nachbarn

Das *k-nächste Nachbarn*-Verfahren (kNN) als ML-Algorithmus des überwachten Lernens kann zur Lösung von Klassifikationsproblemen herangezogen werden. Bei der Zuordnung unbekannter Entitäten zu einer Klasse macht kNN sich bestehende Ähnlichkeiten zu bereits zugeordneten Entitäten zu Nutze [Mat21]. Ein herausstellendes Merkmal dieses Verfahrens besteht darin, etwas von dem in Kapitel 2.1.2 beschrieben Datenfluss abgewichen wird. Bei diesem Algorithmus wird in der Trainingsphase kein Modell erstellt, welches später Zuordnungen vornehmen

kann, sondern die Berechnungen und die Klassifikation einzelner Datensätze finden zum selben Zeitpunkt statt [Ver20; Mat21]. Bei der Bestimmung der Klassenzugehörigkeit, prüft das kNN-Verfahren, wie weit dieser von anderen bereits klassifizierten Datensätzen entfernt ist [Ver20]. Der Parameter k in kNN steht für die Anzahl der Nachbarn, welche betrachtet werden sollen. Wird $k = 1$ gewählt, bestimmt lediglich der nächste Nachbar das Klassenlabel. Wird sich bei der Implementierung für $k = n$ entschieden, fließen die n nächsten Nachbarn in die Entscheidungsbildung mit ein [Mat21]. In Abbildung 2.7 ist die Funktionsweise des Verfahrens für $k = 3$ beispielhaft abgebildet. An dieser Stelle sei angemerkt, dass die Bestimmung von k wichtig für eine angemessene Funktionsweise des Verfahrens ist. Wird k ein kleiner Wert zugewiesen, wird sich bei der Klassenbildung nur auf wenige Daten gestützt, was besonders häufig zu Fehlklassifikation führen kann. Haben jedoch zu viele nächste Nachbarn an der Entscheidungsfindung ihren Anteil, werden Muster unter Umständen nicht präzise genug abgebildet. Neben der Festsetzung des Parameters k besteht eine wichtige Entscheidung darin, wie bei $k \geq 2$ entschieden wird, sollten Nachbarn unterschiedliche Klassenlabels besitzen. Eine Möglichkeit besteht darin, die Klassen aller k-nächsten Nachbarn in die Entscheidung miteinzubeziehen, indem der Mittelwert aller betrachteten Klassen gebildet wird. Alternativ dazu kann es jedoch auch als sinnvoll erachtet werden, solchen Nachbarn, für die eine geringere Distanz gemessen wurde, eine größere Wichtigkeit zuzusprechen [NS18].

Neben seiner leichten Implementierung hat das kNN-Verfahren den Vorteil, dass es zur Erkennung von Ausreißern genutzt werden kann [NS18; Ver20]. Eine Schwierigkeit, welche dieses Verfahren jedoch mit sich bringt, ist die Abhängigkeit vom Wert k. Dieser trägt viel zu der Präzision der Vorhersagen bei, wohingegen er gleichzeitig nicht leicht zu bestimmen ist. Ferner können sich überlappende Klassen mithilfe von kNN nicht berücksichtigt werden. Dazu ist der Algorithmus langsam und rechenintensiv, da für jeden Punkt zunächst alle Abstände berechnet werden müssen und zusätzlich ein (gewichtetes) Mittel bestimmt werden muss [Ver20].

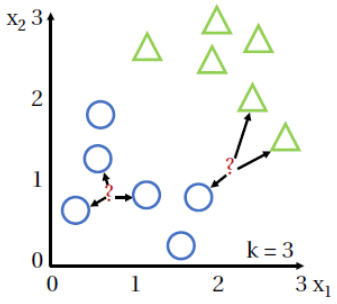


Abbildung 2.7: Beispiel für kNN mit k=3 [Mat21]

2.1.4.6 Bayessche Netze

Bayessche Netze können Prognosen für Größen, das Eintreffen bestimmter Ereignisse oder die Zugehörigkeit zu einer Klasse abgeben [Dör18; DF12]. Ein bayessches Netz betrachtet Variablen und deren Abhängigkeiten zueinander, um Wahrscheinlichkeitsaussagen zu treffen. Bayessche Netze basieren auf dem Satz von Bayes [Döb+18]. Dieser Satz trifft Aussagen darüber, wie wahrscheinlich es ist, dass ein bestimmtes Ereignis eintritt, unter der Bedingung, dass ein anderes Ereignis eingetreten ist [Dör18]. Bayessche Netze sind gerichtete Graphen, welche diesen Satz anwenden [Dör18; Döb+18]. In solchen Graphen stellen die Knoten Ereignisse oder Größen dar, während die Kanten die Abhängigkeiten der verschiedenen Knoten abbilden. Jeder Knoten besitzt mindestens zwei mögliche Zustände, wie z.B. wahr oder falsch. Darüber hinaus werden den Knoten Wahrscheinlichkeitswerte in tabellarischer Form zugewiesen. In diesen Netzen können bestimmte Arten von Beziehungen existieren. So kann ein Ereignis ein anderes bedingen. Auch möglich ist, dass mehrere vorgeschaltete Ereignisse in Kombination Auswirkungen auf ein weiteres haben [DF12]. In der Abbildung 2.8 ist ein simples bayessches Netz dargestellt. Bei

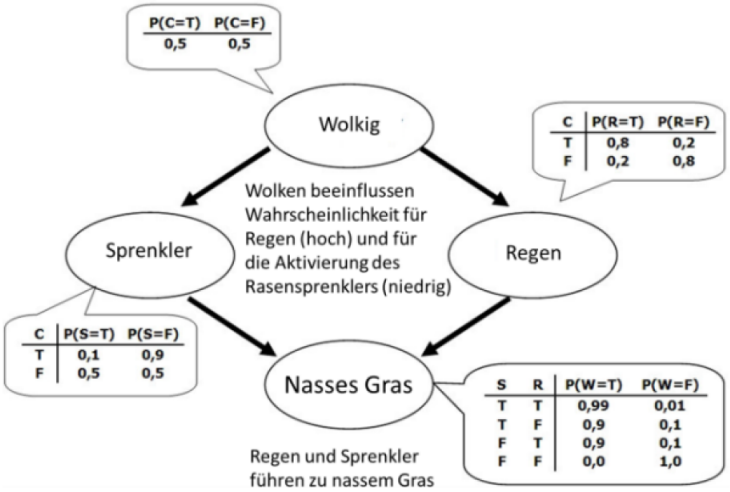
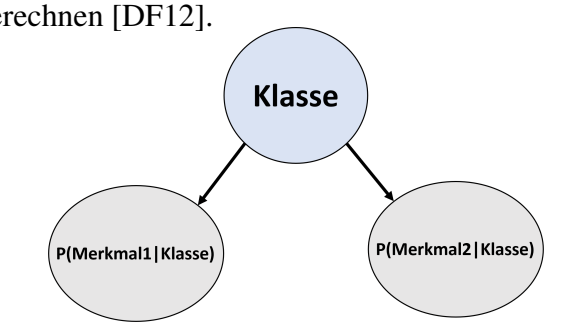


Abbildung 2.8: Beispiel für Bayessche Netze, entnommen aus [Döb+18]

bewölkten Wetter ist es sehr wahrscheinlich, dass es regnet, während es weniger wahrscheinlich ist, dass ein Rasensprenkler gestartet wird. Regen und ein Rasensprenkler in Kombination bedingen mit hoher Wahrscheinlichkeit, dass der Rasen nass wird [Döb+18]. Auf Grundlage von (kausalen) Beziehungen kann also die Wahrscheinlichkeit für das Eintreten eins bestimmten Ereignisses (Nasses Gras) berechnet werden, wenngleich die Wahrscheinlichkeit für genau dieses Ereignis - „Nasses Gras“ - im Vorhinein nicht bekannt war [Döb+18; DF12]. Die entsprechenden Werte lassen sich mithilfe des Satz von Bayes berechnen [DF12].

Der Naive-Bayes-Klassifikator stellt die simpelste Ausprägung der bayesschen Netze dar [Dör18]. Hierbei wird für einer Entität die wahrscheinlichste Klassenzugehörigkeit ermittelt. Ein Anwendungsbeispiel, welches in Abbildung 2.9 illustriert ist, besteht darin, festzustellen, ob es sich bei einer E-Mail um Spam oder eine legitime Nachricht handelt. Attribute der E-Mail, welche für die Zuordnung wesentlich sind, sind bestimmte Wörter oder Eigenschaften, wie das Fehlen einer Signatur. Zur Implementierung eines Naive-Bayes-Klassifikators würde ein Entwickler zunächst in vorliegenden Datenbanken mittels Durchzählen die entsprechenden Wahrscheinlichkeiten bzgl. der jeweiligen Merkmale ermitteln. Dieses Ergebnis ist Beispielhaft in den jeweiligen Tabellen dargestellt. Danach kann unter Zuhilfenahme des bayeschen Satzes die Wahrscheinlichkeit bzgl. der Klassenzugehörigkeit ermittelt werden. Welche Merkmale von der E-Mail erfüllt werden, bestimmen die Klasse. In diesem Kontext sei erwähnt, dass bei der Zuordnung davon ausgegangen wird, dass es sich (anders als es bei den bayesschen Netzen der Fall ist) bei den Variablen um voneinander unabhängige Merkmale handelt [Dör18].



Klasse	Merkmal 1: Wort „Passwort“ enthalten	
Spam	Ja	0,7
	Nein	0,3
Kein Spam	Ja	0,3
	Nein	0,7

Klasse	Merkmal 2: Signatur vorhanden	
Spam	Ja	0,4
	Nein	0,6
Kein Spam	Ja	0,8
	Nein	0,2

Abbildung 2.9: Naive-Bayes-Klassifikator, eigene Darstellung

Besonders bayessche Netze können es leisten, komplexe Beziehungen zu analysieren [DF12]. Das Ziel besteht darin, optimale Entscheidungen durch die Bestimmung von unbekannter Wahrscheinlichkeiten zu treffen [Dör18]. Hierbei wirkt sich das Fehlen von Informationen oder

Erfahrungen weniger stark auf das Ergebnis aus, da dies mittels bekannten Werten bzgl. anderer Merkmale ausgeglichen werden kann. Somit können auch Unsicherheiten umgangen werden [DF12]. Jedoch ist das Konstruieren eines bayesschen Netzes unter Umständen sehr komplex [Wit02]. Kritisch sei bzgl. des Naive-Bayes-Klassifikators angemerkt, dass dieser die Variablen unabhängig voneinander betrachtet und mit numerischen Variablen weniger gut umgehen kann [Ver20].

2.1.4.7 Künstliche neuronale Netze

Die Einsatzgebiete neuronaler Netze sind vielfältig, so können diese beispielsweise zur Klassifikation, Regression oder zur Bilderkennung eingesetzt werden [Ayo10; NS18]. Beim Lernprozess der neuronalen Netze wird sich am biologische Gehirn orientiert. Gehirne setzen sich vereinfacht gesagt aus Neuronen zusammen, welche mit Synapsen verbunden sind, welche für einen Informationsaustausch sorgen. Analog zum Gehirn stellen Neuronen in künstlichen neuronalen Netzen ein grundlegendes Element dar, welche häufig in verschiedenen Schichten (engl. *Layers*) angeordnet und verbunden sind [Cho+20]. Zu Beginn wird in diesem Unterkapitel auf Neuronen selbst eingegangen, worauf folgend die Grundlagen bzgl. der Strukturen und des Trainings neuronaler Netze vermittelt werden.

Das Neuron

Als Grundbestandteil eines neuronalen Netzes übernimmt ein einzelnes Neuron die Aufgabe, Input eines vorangestellten Neurons zu empfangen und seinen eigenen Output zu berechnen, den es wiederum an ein weiteres Neuron weiterleitet. Wie in Abbildung 2.10 illustriert, bildet ein Neuron seinen Output als Summe über alle Eingangswerte, wobei auch Gewichte der einzelnen Inputs beachtet werden und jeder Input ein anderes Gewicht haben kann [Ert21]. Diese Gesamtsumme der Inputs wird zusätzlich mit dem Bias-Term addiert, woraufhin das Gesamtergebnis in eine Aktiviierungsfunktion f gegeben wird [Ert21; Ver20]. Der Einsatz verschiedenster Aktivierungsfunktionen ist im Kontext neuronaler Netze denkbar, wobei zu berücksichtigen ist, dass diese nicht-linear sein sollte. Beispiele für Aktivierungsfunktionen sind die *Rectified Linear Unit*- (ReLU), die Sigmoid-Funktion oder die Softmax-Funktion, welche alle je nach Position innerhalb eines neuronalen Netzes und abhängig vom Problem, was es zu lösen gilt, unterschiedliche Vor- und Nachteile mit sich bringen [Cho+20]. Wurden innerhalb eines Neuron alle Inputdaten mithilfe der Aktivierungsfunktion bearbeitet, wird der erzeugte Output an das nächste Neuron weitergereicht und der Prozess wiederholt sich mit einem anderen Neuron [Ert21].

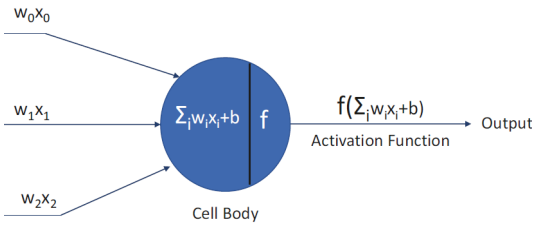


Abbildung 2.10: Neuron, entnommen aus [Ver20]

Strukturen neuronaler Netze

Werden mehrere Neuronen gruppiert und verbunden, erhält man ein neuronales Netz. In einem simplen neuronalen Netz bilden die ersten und letzten Neuronen die sogenannte sichtbare Schichten, während die dazwischen liegenden Schichten als unsichtbare Sichten bezeichnet werden [Cho+20]. Bestehen Verbindungen zwischen den Schichten nur in eine Richtung, spricht man von einem „feed-forward“-Netzwerk [Ert21]. Ein solches ist in Abbildung 2.11 zu sehen.

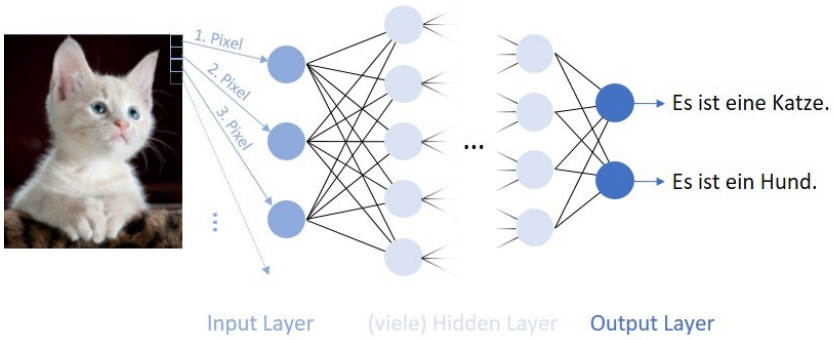


Abbildung 2.11: Neuronales Netz entnommen aus [Mül19]

Bei der ersten Schicht handelt es sich um die Input-Schicht, welche die (u.U. vorverarbeiteten) Daten der Problemstellung entgegen nimmt. In dem Beispiel aus Abbildung 2.11 fließen die einzelnen Pixel des Bildes als Input in das neuronale Netz. Hierbei würde die Input-Schicht aus genau so vielen Neuronen bestehen, wie das Bild Pixel besitzt. Auf die Input-Schicht folgen verdeckte Schichten, in denen die Input-Daten nacheinander mithilfe der Neuronen transformiert werden. Die Output-Schicht als letzter Layer des neuronalen Netzes liefert das Vorhersageergebnis. Das Erscheinungsbild dieses Layers, also z.B. die Anzahl der Neuronen, ist vom zugrundeliegenden Problem und dessen Lösungsraum abhängig. In Bezug auf die verdeckten Schichten, sei angemerkt, dass mit der Anzahl der Schichten häufig auch die Präzision des Vorhersageergebnisses steigt, was jedoch zu Lasten des Rechenaufwandes geht [NS18]. Bei sehr vielen verdeckten Schichten spricht man häufig von DL, wobei nicht klar ist, wie viele Schichten ein künstliches neuronales Netz zu einem DL-Netz machen (vgl. Kapitel 2.1.1).

Neben dieser oben erläuterten Struktur neuronaler Netze existieren noch Abwandlungen, wie z.B. *Convolutional Neural Network* (CNN) oder rekurrente neuronale Netzwerke [Cho+20]. Ein CNN lässt sich realisieren, indem dem neuronalen Netz nach der Input-Schicht eine oder mehrere Faltungsschichten (*Conolutional Layer*) sowie *Pooling*-Schichten hinzugefügt werden [NS18; Cho+20]. Die Faltungsschicht hat zum Ziel lokale Muster zu finden, während das Pooling dafür verantwortlich ist, die Datenfülle zu reduzieren, da hierbei jedes Neuron nur die relevantesten Information an ihre jeweiligen Nachbarn weitergibt. CNNs werden häufig zur Bilderverarbeitung eingesetzt, da die Information, welche Pixel in einem Bild der Daten nebeneinander liegen, genutzt werden kann. Daneben existieren noch rekurrente neuronale Netzwerke als Sonderform der neuronalen Netze, welche besonders gut für die Analyse von zeitlichen Abfolge, wie z.B. Wetterdaten verwendet werden können. Hier wird nicht der gesamte Input auf einmal, sondern elementweise im Netzwerk verarbeitet. Zusätzlich verfügt jedes Neuron über einen eigenen Speicher, um Informationen über das vorherige Element nicht zu vergessen [Cho+20].

Training neuronaler Netze

Beim Trainieren eines neuronalen Netzes steht die Anpassung der Gewichte und der Bias-Terme im Vordergrund. Da die Neuronen eigene Gewichte und eigenen Bias-Terme besitzen, besteht die besondere Herausforderung beim Trainieren darin, diese vielen, unterschiedlichen Parameter anzupassen. Zum Lernen wird eine Verlustfunktion (engl.: *loss function*) eingesetzt, welche messen kann, inwieweit die durch das Netzwerk errechnete Antwort von der korrekten, realen Antwort abweicht [Cho+20]. Der Prozess der Anpassung der Gewichte und Bias-Vektoren selbst wird als *Backpropagation* (dt.: Rückverfolgung) bezeichnet. Trifft das neuronale Netz

eine falsche Vorhersage in der Output-Schicht, erfolgt eine Rückverfolgung des Fehlers und die Parameter werden insoweit angepasst, dass sich der Wert der Kostenfunktion reduziert, was wiederum eine Reduktion des inhaltlichen Fehlers zur Folge hat. Dieses Training wird meist in mehreren Runden durchlaufen [NS18].

Das Potenzial der neuronalen Netze, welche das menschliche Gehirn abbilden, ist groß und bietet vielfältige Einsatzmöglichkeiten, während gleichzeitig bei der Implementation einige Herausforderungen bestehen. Analog zu den bereits betrachteten Verfahren, hängt die Funktionsweise stark von der Anzahl der vorhanden Trainingsdaten ab. Besonders, wenn das Netz komplexe Strukturen abbilden soll, und wenig Trainingsdaten vorhanden sind, neigt es zu Überanpassung . Weiter erfordert das Training neuronaler Netze einen hohen Rechenaufwand, der mit jedem zusätzlichen Neuron ansteigt. Ein weiterer Nachteil von neuronalen Netzen besteht darin, dass diese sich von Menschen nur schwer interpretieren lassen. Bei hundert Schichten mit mehreren Neuronen und individuellen Aktivierungsfunktionen sowie Parametern, lässt sich kaum nachvollziehen, wie ein neuronales Netzwerk zu seiner Vorhersage gelangt ist [NS18]. Eine Lösung für dieses Problem der Uninterpretierbarkeit bietet das erklärbare ML, das in folgenden Absätzen kurz dargestellt wird.

Überanpass ggf. einführen

2.1.5 Erklärbares maschinelles Lernen

Erklärbares maschinelles Lernen (engl.: *Explainable Artificial Intelligence*; Abkz.: XAI) hat zum Gegenstand Systeme künstlicher Intelligenz zu schaffen, welche Vorhersagen erklären können [Xu+19; SM18]. Unter dem Begriff „Erklären“ wird grundsätzlich das Abgeben eines Grundes oder einer Rechtfertigung für eine gewisse Handlung verstanden [DSB17]. Insbesondere bei selbstlernenden Systemen, welche in der Medizin oder bei der Klassifikation von Menschen eingesetzt werden (und dabei eventuell unethische Entscheidungen treffen könnten), ist es verständlich, dass Nutzer getroffene Systementscheidungen nachvollziehen wollen oder müssen [Ant+21; Xu+19]. Aber nicht nur für Nutzer ist es von Vorteil ML-Systeme zu verstehen; auch Entwicklern wird es leichter fallen, das Potenzial des Systems voll auszuschöpfen, wenn sie die Entscheidungen des ML-Algorithmus nachvollziehen können [Xu+19]. So stellt XAI speziell bei neuronalen Netzen, wie CNNs oder RNNs einen Mehrwert bereit, da diese durch ihre häufig vielen Neuronen und Verbindungen als besonders undurchdringbare Blackboxen wirken [Xu+19; NS18].

2.1.5.1 Erkläransätze

Um diese Blackboxen zu erklären, existieren mehrere grundlegende Ansätze, von denen sich einige eher an den Entwicklern, andere wiederum an den Nutzern eines Systems orientieren [Xu+19]. [KL21] untergliedern die Methoden der XAI in drei Kategorien: *Umfang*, *Stadium* und *Modell*. Steht der Umfang im Vordergrund, lassen sich lokale oder globale Methoden anwenden, um ein ML-System zu erklären. Globale Erklärmethoden agieren auf Makroebene und versuchen, die Vorhersagen des Gesamtmodells zu erklären, wobei vor allem ein grobes Verständnis über die Strukturen und Parameter geschaffen werden soll. Lokale Erklärungsansätze hingegen beziehen sich auf konkrete Vorhersagen, indem dargelegt wird, worauf eine bestimmte Vorhersage beruht. Dies könnte z.B. ein konkreter Gegenstand sein, welcher auf einem Bild zu sehen ist [KL21].

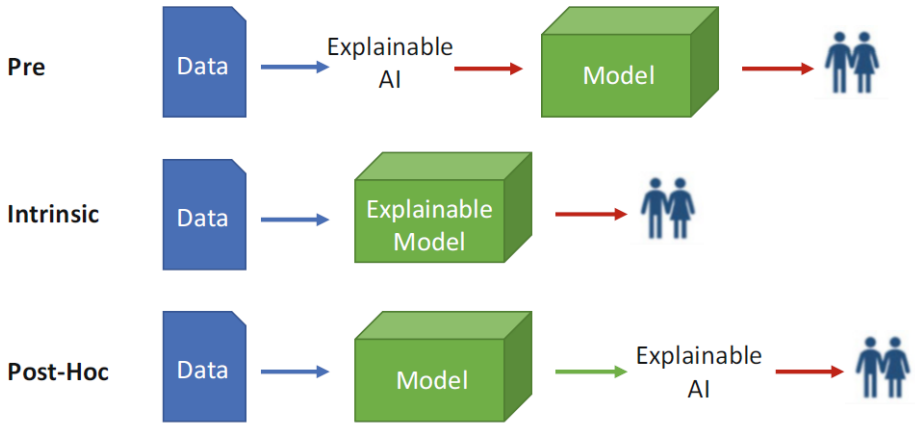


Abbildung 2.12: XAI Methoden nach Stadium, entnommen aus [KL21]

Methoden der XAI lassen sich überdies dahingehend unterscheiden, in welchem Stadium diese eingesetzt werden, was in Abbildung 2.12 illustriert ist. Hierbei wird zwischen *Pre-Model*, *Intrinsic* oder *Post-Hoc* differenziert. Ansätze, welche sich den Pre-Model-Methoden zuordnen lassen, finden vor der eigentlichen Modellauswahl statt. Daher liegt hier der Fokus nicht auf einem konkreten Modell, sondern auf den zugrundeliegenden Daten. Aktivitäten, welche im Rahmen von Pre-Model-Erklärungsansätzen durchgeführt werden können, sind z.B. das Verstehen, die Auswahl oder die Reduzierung der Daten. Bei der Anwendung intrinsischer Erklärmethoden, machen sich Entwickler die selbsterklärende Struktur einiger ML-Algorithmen zunutze, wie sie z.B. Entscheidungsbäume besitzen. Angemerkt sei an dieser Stelle, dass diese Art der Methode etwas eingeschränkt ist, da nicht jeder Algorithmus eine solche Struktur für Erklärungen mit sich bringt. Post-Hoc-Methoden als dritte Art der Unterscheidung füllen diese Lücke, da sie sich auf Blackbox-Systeme anwenden lassen können. Indem Beziehungen zwischen den Eingabewerten und Vorhersagen verdeutlicht werden, lässt sich auch das Verhalten von Modellen erklären [KL21]. Diese Methoden fokussieren sich eher auf den Nutzer und können z.B. mittels Visualisierungen oder Beispielen aus den Trainingsdaten die Entscheidungen des Algorithmus untermauern [Xu+19]. Ein besonderer Vorteil der Post-Hoc-Methoden besteht darin, dass es nicht zwingend notwendig ist, die interne Struktur des ML-Systems in Gänze zu kennen oder zu verstehen [KL21].

Schlussendlich lassen sich Erklärungsansätze entweder als „Modelagnostic“ oder als „Model-specific“ bezeichnen. Während Erkläransätze existieren, welche sich auf jedes Modell anwenden lassen (Modelagnostic), gibt es auch spezielle Methoden, die nur für bestimmte Typen von ML-Algortihmen geeignet sind (Modelspecific) [KL21].

2.1.5.2 Erklärungen

Mithilfe der im vorangegangen Kapitel 2.1.5.1 erläuterten Ansätzen, lassen sich verschiedene Erklärungen für das Verhalten eines ML-Algortihmus erstellen. [KL21] nennen *globale*, *lokale*, *kontrastive*, *Was-wäre-wenn*, *kontrafaktische* und *beispielbasierte Erklärungen* auf welche im folgenden näher eingegangen wird.

Globale Erklärungen sind ganzheitliche top-down-Ansätze und fokussieren sich auf die Funktionsweise des Modells, welche mit Visualisierungen, mathematischen Formeln oder Graphen

abgebildet werden kann. Während globale Erklärungen, darlegen, wie ein Modell funktioniert, bilden lokale Erklärungen als bottom-up-Ansätze eher ab, wieso ein Modell eine bestimmte Entscheidung getroffen hat. Ähnlich dazu können im XAI-Umfeld kontrastive Erklärungen abgegeben werden, die Aussagen der Form „Warum X und warum nicht Y“ treffen. Diese eignen sich besonders dazu, Auswirkungen von minimalen Änderungen einzelner Modellparameter zu analysieren. „Was-wäre-wenn“-Erklärungen legen wiederum da, wie sich ein Modell bei Änderung bestimmter Eingaben oder Parameter verändern würden und schaffen somit Verständnis dafür, wie Vorhersagen und Parameter zueinander in Beziehungen stehen. Ähnlich dazu wirken kontrafaktische Erklärungen, die angeben, welche Änderungen am Modell zu welchen anderen Ausgaben führen würden. Somit bekommen Entwickler eine Idee davon, wie sich ein gewünschtes Ergebnis mit minimalen Änderungen erreichen ließe. Eine simple Art der Erklärung besteht darin Beispiele mitzugeben, welche aus den Trainingsdaten stammen [KL21].

Von diesen bis zu dieser Stelle eher abstrakt umrissenen Erklärung(-sansätzen) existieren konkrete Instanzen, welche ML erklärbar machen. Ein bekanntes Beispiel ist „Local Interpretable Model-Agnostic Explanations“ (LIME), welches sich für mehrere Arten von Klassifikationsverfahren einsetzen lässt [RSG16]. Aufgrund ihrer Nicht-Interpetierbarkeit stehen DL-Netzwerke jedoch besonders im Vordergrund des XAI-Forschungsfeldes [Hol18]. Für diese ML-Typen existieren Verfahren, die z.B. anzeigen, welche Neuronen besonders stark aktivierte wurden. Hiervon wird sich erhofft, Rückschlüsse darauf zu ziehen, welches Merkmal besonders wichtig ist [Xu+19]. Eine weitere konkrete Methode heißt „Layer-Wise Relevance Propagation“, welche Werte für die Relevanz eines Neurons für das Zustandekommen einer bestimmten Entscheidung ermitteln kann. Hier wird der Lernprozess rückwärts durchlaufen und so kann ermittelt werden, welches Neuronen welchen Beitrag für die Entscheidung beigetragen hatte [Mon+19]. Auf weitere konkrete Methoden wird in Kapitel im Rahmen der Handreichung eingegangen. tbd

2.2 Transparenz

Dieses Kapitel definiert den Term "Transparenz", welcher im Mittelpunkt dieser Arbeit steht. Darauf folgend wird dargestellt, warum Transparenz speziell bei ML-System von großer Bedeutung ist.

2.2.1 Definition

Der Begriff Transparenz kommt in den unterschiedlichsten Domänen vor, meint grundsätzlich jedoch, dass etwas durchschaubar oder nachvollziehbar ist [Dud]. Im ML-Umfeld hilft Transparenz dem in Kapitel 2.1.5 beschriebenen Blackbox-Charakter entgegen zu wirken und lässt sich im Datenschutz-Umfeld „Klarheit, Erkennbarkeit und Nachverfolgbarkeit“ verstehen [BA10; Rud19]. Konkret sollten „Systeme [...] durchschaubar und ihre Funktions- und Arbeitsweise nachvollziehbar und verständlich“ sein und die verarbeiteten Daten sowie die beteiligten Personen und Handlungen offenlegen [BA10]. Im XAI-Umfeld finden sich noch weitere Definitionen und Einordnungen des Begriffes der Transparenz. [KL21] stellen heraus, dass ein transparentes ML-Modell sowohl eine verständliche Struktur als auch einen verständlichen Algorotihmus besitzen muss. Transparenz ist eine Voraussetzung dafür, Modelle und deren Funktionsweisen zu verstehen. So sei an dieser Stelle erwähnt, dass Transparenz mit Erklärbarkeit (vgl. die

Definition im vorangegangenen Kapitel 2.1.5) eng verwoben ist. Die Beziehung zwischen diesen Konzepten ist derart, dass sich Erklärbarkeit positiv auf Transparenz auswirkt [KL21].

2.2.2 Relevanz von Transparenz bei Algorithmen des maschinellen Lernens

Transparenz ist insbesondere bei Algorithmen des MLs wichtig, da diese über einen stark ausgeprägten Black-Box-Charakter verfügen [Rud19]. Es kann jedoch gewünscht sein, nachzuvollziehen, wie und warum ein Algorithmus, eine Entscheidung getroffen hat [Fin20]. Für diesen Wunsch nach Transparenz kommen verschiedene Motive unterschiedlichster Akteure in Frage. So ist es denkbar, dass es einem Entwickler wichtig ist, das System zu verstehen mit dem schlussendlichen Ziel dessen Funktionsweise zu verbessern. Daneben ist es für die Nutzer einer Software bedeutsam das System zu verstehen, um Vertrauen in die Technologie fassen zu können. Auch während der Nutzung eines Systems ist es von Vorteil, wenn das Verhalten des Systems nachvollziehbar bleibt, denn dies führt auch dazu, dass der Nutzer nicht zu einem Konkurrenzprodukt greift, was wiederum dem Softwarehersteller zugute kommt. Aber nicht nur für unmittelbar Systembetroffene ist Transparenz von Bedeutung, auch für die Gesellschaft ist es bedeutsam Stärken und Grenzen von Technologien einordnen zu können [Wel19]. Des weiteren rückt Transparenz in den Vordergrund, wenn Menschen fürchten, die Kontrolle über das eingesetzte System zu verlieren, was besonders im medizinischen Kontext fatale Folgen haben könnte [Fin20; Ant+21]. Aber auch im strafrechtlichen Umfeld, in dem Personen auf Grundlage von Entscheidungen eines Algorithmus Überwachung zu befürchten haben, wird Transparenz als ethisch und rechtlich notwendiges Gut betrachtet.Ferner ist auch ein nichtdiskriminierender Charakter gewünscht, welcher die Einhaltung der Grundrechte sichert [EU21]. Auch lassen sich Systemverantwortliche oder das gesamte System mittels Transparenz leichter überwachen und in Bezug auf Sicherheitsstandards testen [Wel19; SCN18]. Überwachung des Systems ist z.B. bei selbstfahrenden Autos denkbar, in dem im Falle eines Unfalls Zurechenbarkeit und rechtliche Haftung ermöglicht werden soll [Wel19]. Auch in einem Vorschlag der Europäischen Kommission [EU21] findet sich die Forderung nach Transparenz besonders bei bestimmten Systemeinsatzkontexten (z.B. Deep-Fake-Systeme) wieder, um Manipulationsrisiken erkennen zu können.

Die Forderung nach Transparenz ist jedoch nicht nur ein Motiv einiger Akteure, sondern bereits in bestehende Gesetze etabliert. Eine Forderung nach Transparenz bei der Datenveranverarbeitung findet sich in der Datenschutz-Grundverordnung (DSGVO) wieder. In Art. 15 DSGVO werden Privatpersonen beispielsweise die Rechte eingeräumt, Auskunft über die „Verarbeitungszwecke“ sowie „die Kategorien personenbezogener Daten, die verarbeitet werden“ einfordern zu dürfen. Die DSGVO fordert insbesondere in Artikel 12, dass „alle Informationen und [...] Mitteilungen [...], die sich auf die Verarbeitung beziehen, in präziser, transparenter, verständlicher und leicht zugänglicher Form in einer klaren und einfachen Sprache [übermittelt werden sollen]“.

Wenngleich viele, berechtigte Gründe für Transparenz innerhalb von ML-Systemen existieren, sei an dieser Stelle angemerkt, dass dies auch negative Folgen nach sich ziehen kann. So ist es möglich, dass Transparenz zu Lasten von Effizienz eingeführt wird, aber es ist auch denkbar, dass Transparenz eines Systems ausgenutzt wird, um ebendieses zu manipulieren oder die daraus gewonnenen Informationen unangemessen zu nutzen [Wel19].

2.3 Dokumentation von IT-Systemen

Nach [KT21] sind Dokumentation ein gutes Werkzeug zur Kommunikation und lassen sich gleichzeitig auch dazu einsetzen, ein System transparent, fair sowie zurechenbar zu machen. Sollen IT-Systeme Menschen näher gebracht werden, ist zunächst festzustellen, wer angesprochen werden soll. Im Umfeld der SW-Architektur-Dokumentationen stellen eine Art von Dokumentationen solche dar, welche für zukünftige Teammitglieder geschrieben sind, die aus dieser Motivation heraus das System verstehen wollen. Eine solche Dokumentation bietet darüber hinaus das Potenzial das System zu analysieren und zu verbessern. Neben der technischen Seite, sollten weiter verschiedene Stakeholder wie Nutzer, Auftraggeber oder Projektmanager beachtet werden, mit denen das System kommuniziert werden soll. Neben dem *Wer* gibt es bei der Kommunikation von Systemen die Frage nach dem *Was*. Grundsätzlich können einzelne Module, welche Teile von Software abbilden, Gegenstand von Dokumentationen sein. Auch kann die Performance z.B. als Fluss-Diagramm abgebildet werden. Weiterführend gibt es Unterschiede in der Art der Kommunikation. Diese ist informal durch die Unterstützung von Grafiken möglich, aber auch semiformal z.B. in Form eines UML-Diagramms [BCK03]. Die Standards im Bereich der Softwaredokumentation lassen sich nur bedingt auf selbstlernende Algorithmen übertragen [Rod99]. Im Rahmen der Handreichung in Kapitel tbd wird auf Besonderheiten bei Verfahren des maschinellen Lernens eingegangen.

Vlt. hier noch auf Besonderheiten bei ML eingehen?

3 Transparenzschaffende Maßnahmen aus der Literatur

Im folgenden Kapitel ist der aktuelle Stand der Forschung dargestellt, welcher sich mit transparenzsschaffenden Maßnahmen im Umfeld des maschinellen Lernens beschäftigt. Die Erkenntnisse der Literaturrecherche bilden maßgeblich den Grundstein, für die zu erstellende Handreichung, die den Kern dieser Masterarbeit darstellt. Zunächst (in Kapitel 3.1) werden die Rahmenbedingungen dargelegt, an denen sich das Vorgehen orientierte. Darauf folgend werden in Kapitel 3.2 mit Blick auf die in Kapitel 1.2 formulierten Forschungsfragen die Ergebnisse der Literaturrecherche vorgestellt.

3.1 Vorgehen der Literaturrecherche

Unter einer Literaturrecherche versteht man das strukturierte Zusammentragen, Beschreiben und Erklären von wissenschaftlicher Literatur [Coo88]. Hierbei können Methoden, Modelle oder Theorien im Vordergrund stehen, wogegen es eine Literaturrecherche gleichzeitig auch ermöglicht, den aktuellen Stand der Forschung zusammenzufassen. Somit lässt sich auch aufdecken, an welcher Stelle in einem Forschungsfeld noch Wissenslücken bestehen [Coo88; WW02].

Für eine Literaturrecherche stehen unterschiedlichste Vorgehensmodelle zur Verfügung. Diese Masterarbeit orientiert sich jedoch grob an dem Vorgehen von [Bro+09], da dieses von mehreren wirtschaftsinformatiknahen Forschern im deutschsprachigen Raum entwickelt wurde. Die grobe Struktur des Vorgehensmodells ist in Abbildung 3.1 zu sehen. Die Literaturrecherche nach [Bro+09] umfasst die *Definition des Umfangs der Literaturrecherche*, *Konzeptualisierung des Themas*, *Literaturrecherche*, *Literaturanalyse und -synthese* und *Forschungsplanung*. Bei einzelnen Phasen der Literaturrecherche wird weiter noch auf Erkenntnisse von [Coo88], [Bri13] und [WW02] zurückgegriffen, was in den nachfolgenden Unterkapiteln erläutert wird.

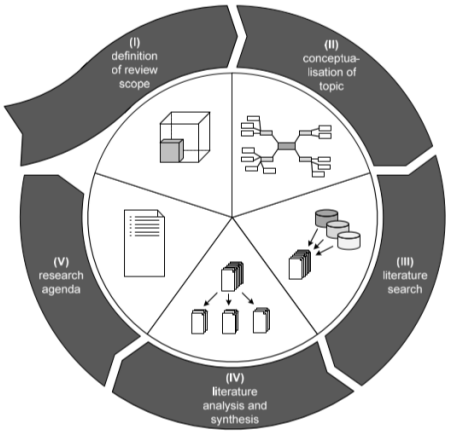


Abbildung 3.1: Vorgehen einer strukturierten Literaturrecherche, entnommen aus [Bro+09]

3.1.1 Definition des Umfangs der Literaturrecherche

Um eine Abgrenzung des Umfangs einer Literaturrecherche vorzunehmen, empfehlen [Bro+09] die Taxonomie von [Coo88], welche in Abbildung 3.1 dargestellt ist. Je nach Charakteristika kann sich während einer Literaturrecherche auf unterschiedliche Kategorien fokussiert werden. Die Entscheidungen, wo jeweils der Fokus liegen sollte, ergab sich meist aus der Aufgabenstellung. In der Abbildung 3.1 ist durch Fettschrift hervorgehoben, welches Merkmal im Fokus lag, was folgend kurz begründet wird.

Charakteristika	Kategorien
Fokus	• **Forschungsergebnisse** • Forschungsmethoden • Theorien • **Praktiken oder Anwendungen**
Ziel	• **Integration** • Kritik • Identifikation zentraler Herausforderungen
Perspektive	• **Neutral** • Bewertend
Umfang	• Vollständig • Vollständig mit Selektion der Zitierungen • **Repräsentativ** • Zentral
Organisation	• Historisch • **Konzeptuell** • Methodologisch
Zielgruppe	• Spezialisierte Wissenschaftler • Generalisierte Wissenschaftler • **Praktiker** • Gesellschaft

Tabelle 3.1: Umfang der Literaturrecherche, angelehnt an [Coo88]

(1) **Fokus:** Da eine Handreichung, welche sich im praktischen Kontext einsetzen lassen soll, primärer Gegenstand der Masterarbeit ist, legen eher Praktiken und Anwendungen im Fokus. Nichtsdestotrotz sollen auch Ergebnisse wissenschaftlicher Forschungen in die Gestaltung der Handreichung einfließen, da diese Handreichung im Rahmen einer wissenschaftlichen (Master-)arbeit entsteht.

(2) **Ziel:** Ziel der Literaturrecherche ist die Integration und Zusammenfassung verschiedener Methoden und Ansätze, welche in der Literatur existieren, um ein ML-System transparenter zu gestalten.

(3) **Perspektive:** Bei dem Zusammentragen verschiedener Maßnahmen wird eher eine neutrale Position eingenommen. Eine Wertung erfolgt jedoch (teilweise) in der Evaluation dieser Arbeit zu einem späteren Zeitpunkt.

(4) **Umfang:** Da in der Masterarbeit mehrere Forschungsfragen beantwortet werden müssen, in denen teilweise auf transparenzschaffende Maßnahmen mehrere ML-Verfahren eingegangen werden muss, wurde sich gegen eine vollständige Erfassung jeder zur Verfügung

stehenden Literatur entschieden, um den zeitlichen Rahmen nicht zu sprengen. Stattdessen wurde eine repräsentative Recherche angestrebt, welche auf selektierten wissenschaftlichen Datenbanken aufbaut.

(5) **Organisation:** Die Rechercheergebnisse werden konzeptuell dargestellt.

(6) **Zielgruppe:** Die Ergebnisse der Literaturrecherche dienen vor allem der Erstellung der Handreichung, welche sich per Aufgabenstellung eher an Praktiker, wie z.B. die Entwickler von ML-Systemen richten soll.

3.1.2 Konzeptualisierung des Themas

Als zweiten Schritt sehen [Bro+09] vor, sich einen groben Überblick über das Thema zu verschaffen, um potenziell relevante Bereiche zu identifizieren. Hierfür wurde eine unstrukturierte Literaturrecherche durchgeführt, um Kernkonzepte zu finden, welche später den Grundstein für die strukturierte Literaturrecherche bilden, da durch diese relevante Suchbegriffe identifiziert werden konnten. Einige Kernaspekte standen nichtsdestotrotz durch die Aufgabenstellung bereits fest, aber um sich zusätzlich einen Überblick zu verschaffen, wurde der Suchbegriff „Transparency in Machine Learning“ in die Datenbank Google Scholar eingegeben.

In der folgenden Tabelle 3.2 sind die die Ergebnisse der Konzeptualisierungsphase, also die Suchbegriffe der strukturierten Literaturrecherche abgebildet. Nach [Bri13] sind hierbei auch noch Synonyme, Ober- und Unterbegriffe sowie verwandte Begriffe miteinzubeziehen.

Gruppierung	Kernbegriffe
Maschinelles Lernen	Machine Learning/ML, Künstliche Intelligenz/KI, Artificial Intelligence/AI, Erklärbares Maschinelles Lernen/Explainable Artificial Intelligence/XAI, Black-Box
Transparenz	Transparency, Verständlichkeit/Erklärbarkeit/Explainability, Interpretierbarkeit/Interpretability, DSGVO/GDPR, Zurechenbarkeit/Accountability
Methoden	Entscheidungsbäume/Decision Trees/Random Forest, k-means Clustering, Support Vector Machines, Logistische Regression/Logistic Regression, k-nächste Nachbarn/k-Nearest Neighbours, Bayessche Netze/Bayesian Networks, (Künstliche) Neuronale Netze/Artificial Neural Network/-Deep Learning
Prozess	Process, Entwicklungslebenszyklus/Lebenszyklus/Entwicklung/Development Lifecycle/Lifecycle, Daten/Data, Datensatz/Dataset, Pipeline, Fluss/Flow

Tabelle 3.2: Suchbegriffe

Um der Literaturrecherche eine auf die Forschungsfragen orientierte Struktur zu geben, wurden spezielle Suchterme definiert, welche teilweise individuell auf die jeweiligen Datenbanken zugeschnitten wurden und im Anhang zu finden sind. Während der Erstellung wurden die jeweiligen Begriffe zunächst probeweise in die Datenbank IEEE [1] eingegeben, um ungefähr abzuschätzen, welche und wie viele Ergebnisse erzielt werden.

1. https://ieeexplore.ieee.org/Xplore/home.jsp

Grundsätzliche transparenzschaffende Maßnahmen (RQ1)

In Bezug auf die erste Forschungsfrage (*RQ1: Wie können die unterschiedlichen Ebenen einer ML-Pipeline transparent gemacht werden?*) wurde zunächst versucht alle Begriffe der Konzeptualisierung in den Suchterm miteinzubauen. Jedoch brachte allein der Suchterm „Transparency AND ML OR DATA“ bereits 1.338.968 Ergebnisse, was als unüberschaubar erschien. Aufgrund dieser sehr vielen Suchergebnisse wurde sich dafür entschieden, den Fokus auf den thematischen Kern dieser Masterarbeit zu legen, sodass für die erste Forschungsfrage der Suchterm „Transparency AND Machine Learning“ gewählt wurde, da dieser auch schon zu genügend Ergebnissen führte.

Transparenzschaffende Maßnahmen konkreter ML-Verfahren (RQ2)

Da der Fokus bei der zweiten Forschungsfrage (*RQ2: Welche Ansätze existieren, um verschiedene ML-Algorithmen, wie z.B. Entscheidungsbäume oder auch neuronale Netzwerke zu erklären?*) eher auf den konkreten Methoden, welche innerhalb des ML-Lebenszyklusses eingesetzt werden liegt, wurde sich dazu entschieden, nach Möglichkeiten zu suchen, diese transparent, aber auch erklärbar zu machen. Der Aspekt des erklärbaren maschinellen Lernens wurde hier deswegen miteinbezogen, da in Kapitel 2.2.1 bereits herausgestellt wurde, dass Erklärbarkeit und Transparenz zusammenspielen. Somit wurde für den zweiten Forschungsaspekt nach dem jeweiligen Verfahren sowie nach Transparenz und Methoden der XAI gesucht. Für Entscheidungsbäume lautete der Suchterm beispielsweise: „decision tree AND Transparency AND XAI“.

Transparenzschaffende Maßnahmen je Zielgruppe

In Bezug auf die dritte Forschungsfrage (*RQ3: Inwieweit und auf welche Weise sollten ML-Algorithmen transparent gemacht werden, wenn diese Experten oder fachfremden Personen veranschaulicht werden?*) konnte während der Konzeptualisierung wenig konkretes gefunden werden, was als Suchterm sinnvoll erschien. In Absprache mit dem Betreuer dieser Masterarbeit wurde sich dazu entschieden, auf diese Forschungsfrage etwas weniger Gewicht zu legen als auf die anderen beiden. Inhaltlich wurden jedoch die Ergebnisse, welche sich mit dem anderen Suchthermen finden ließen, auch auf diese dritte Forschungsfrage hin geprüft.

3.1.3 Literaturrecherche

Nach [Bro+09] steht im dritten Schritt des Vorgehens (vgl. 3.1 das Finden der Literatur an, indem Journal- und Wissensdatenbanken mittels Stichworten durchsucht werden, worauf folgend optional eine Vorwärts- und Rückwärtssuche durchgeführt werden kann. Gesucht wurde im Juni und Juli 2022 auf Englisch in folgenden Datenbanken: IEEE, ACM [2], Computer Science Bibliography [3] sowie Google Scholar [4]. zusätzlich wurde noch die reguläre Suchmaschine *Google* genutzt, da sich die Handreichung schließlich an Prakitker richten soll und somit auch Ansätze aus der Praxis, z.B. von Beratungshäusern oder großen Software-Firmen nicht ausgeschlossen werden sollten .

Das muss ich noch machen

2. https://dl.acm.org/
3. http://dblp.uni-trier.de/
4. https://scholar.google.com/

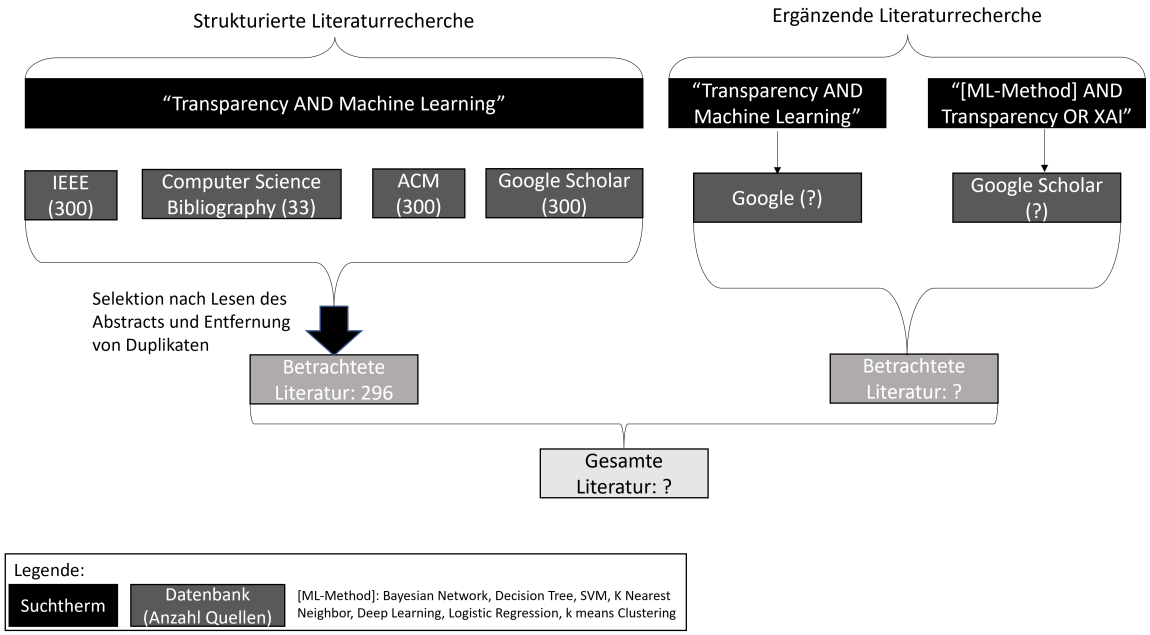


Abbildung 3.2: Vorgehen bei der Literatursuche, eigene Darstellung

Um die Anzahl der zu betrachtenden Publikationen und Beiträge überschaubar zu halten, wurden in wissenschaftlichen Datenbanken für RQ 1 die ersten 200 betrachtet. Da für die erste Forschungsfrage bereits viele wissenschaftlichen Quellen (296) gefunden wurde, die bereits auf Tranpsarenz bei konkreten ML-Methoden eingingen, wurde sich entschieden in Bezug auf RQ2 lediglich eine ergänzende Literaturrecherche durchzuführen, um zum einen keine relevanten transparenzschaffenden Methoden zu übersehen, jedoch zum anderen auf die Dokumentation aller gefundenen Literatur zu verzichten, um den zeitlichen und inhaltlichen Rahmen nicht zu sprengen. - Google für transparenz in Machine Learning um keine in der Praxis relevanten Sachen zu verpassen (hierfür ersten 50 Ergebnisse von Websites) - für RQ2: lesen des Titels und Abstract und dann Aufnahme in den Literaturpool (hierfür wurden die ersten 50 Ergebnisse von Google Scholar je Suchtherm betrachtet) - für die ergänzende Literaturrecherche wurden Duplikate nicht mit in die Dokumentation aufgenommen, sondern nur das was inhaltlich neu war

Eine Überblick über die verschiedenen Datenbanken, Suchterme und -ergebnisse findet sich im Anhang.

3.1.4 Literaturanalyse und -synthese

Teile der gefundenen Literatur wurde zunächst aussortiert. Hierfür wurden im ersten Schritt Duplikate entfernt, worauf folgend im zweiten Schritt die Literatur auf harte Kriterien geprüft wurde. Kriterien, welche hier Anwendung fanden, waren:

- „Peer Reviewed Literature“ (Veröffentlichungen in Jorunals oder Proceedings)
- Monographie/Buch
- Whitepaper
- Dissertationen

- Sprache: Englisch

Im dritten Schritt wurde der Titel und das Abstract der gefunden Beiträge näher betrachtet, um dessen Relevanz für Transparenz im ML einschätzen zu können. So wurden einige Paper aussortiert, die nicht wirklich dem Thema dieser Masterarbeit zuzuordnen sind und beispielsweise nur deswegen in der Suche erschienen, weil sie den Einsatz von ML-Technologien untersuchten, um eine Domäne wie z.B. die Finanzbranche transparenter zu machen. Zu betrachtende Quellen nach dieser ersten Reduktion waren 296 Quellen für RQ1.

das wurde mit dem zweiten Suchtherm nicht mehr gemacht, da hier ja von Anfang an nur die relevanten mit neuen Informationen dokumentiert wurden

Um diese viele Suchergebnisse noch weiter zu strukturieren, wurden die Quellen zunächst noch in grobe Kategorien eingeteilt. Beiträge, welche Transparenz und ML im Titel oder im Abstract behandelten, wurden als relevant klassifiziert, wenn diese auch konkrete Methoden betrachteten. Mit mittlerer Relevanz wurden solche Themen eingeordenet, welche eher den Fokus auf andere Themen legten (z.B. ethische oder rechtliche Behandlungen) legten, aber transparenzschaffende Maßnahmen betrachteten. Mit niedriger Relevanz wurden solche Quellen gekennzeichnet, welche Transparenz im Zusammenhang ML nur als Beispiel betrachteten. Sehr niedrig, wenn sie nur kurz erwähnten, dass etwas transparent ist. Zur inhaltlichen Abgrenzung kamen noch weitere Kategorien hinzu, welche sich zum Teil bereits im Vorhinein aus der Konzeptualisierung in Kapitel 3.1.2 ergaben, jedoch entstanden auch einige Kategorien neu nach Überfliegen der Literatur. So wurden beispielsweise Quellen identifiziert, welche nur bestimmte XAI-Methoden behandelten oder solche, die einen ganzheitlichen Ansatz zur Transparenz im ML darlegten oder sich nur auf Daten fokussierten.

Bei direkten Zitaten Seitenzahl? Joshua fragen

3.1.5 Forschungsagenda

3.2 Ergebnisse der Literaturrecherche

3.2.1 Konzepte

Mit Hilfe der Literaturrecherche konnten weitere Konzepte gefunden werden, welche im Forschungsgebiet des ML im Kontext der Transparenz definiert wurden.

3.2.1.1 Transparentes ML

[ZC18] nennen das Konzept des transparenten maschinellen Lernens. Dies wird definiert als ML, welches es Nutzern ermöglicht die Entscheidungen des Algorithmus zu verstehen. Mit *Nutzer* beziehen sich die Autoren insbesondere auf solche Personen, welche nicht über Kenntnisse im Bereich des Trainieren von ML oder Mathematik verfügen. Transparentes ML bedingt mehrere positive Aspekte. Zum einen ermöglicht es Nutzern bessere Entscheidungen zu treffen und es wird sichtbar, ob ein System so funktioniert wie gewünscht. Im besten Falle ließen sich somit sogar Fehler und deren Ursachen aufdecken, was zu besseren Algorithmen führt. *Besser* meint hier nicht im mathematischen Sinne optimiert, sondern den Einsatz von ML in der realen Welt. Des Weiteren stärkt tansparentes ML das Vertrauen.

3.2.1.2 Mensch und Maschine

Viele Autoren nennen die Zusammenarbeit von Mensch und Maschine als potenzielle Möglichkeit mehr Transparenz zu schaffen. Grundsätzlich lässt sich durch stärkere Intergation des Menschens in die gesamte ML-Pipeline mehr Transaprenz schaffen [ZC18]. Für die Zusammenarbeit von Mensch und Maschine existieren unterschiedliche Konzepte, das am meisten genannte heißt Human-in-the-Loop.

Hier kommt noch eine Human-in-the-Loop Deinfition

Neben den Vorteilen für Nutzer und Entwickler ergeben sich durch Human-in-the-Loop weitere positive Aspekte. Nach [TKC16] haben menschliche Inspektionen und Eingriffe in einen Klassifikationsalgorithmus im Vergleich zu einem traditionellen Algorithmus zu einer höheren Genauigkeit geführt.

[Wen+21] erwähnt *Human-Machine-Teams* als weitere Möglichkeit Transparenz zu schaffen. Hier arbeiten Menschen und Maschinen auf eine Art und Weise zusammen, in der die Maschine neue Muster erlernt und Menschen aufgrund dessen in der Lage sind neue Forschungsfragen und -hyptothesen zu bilden. Hierbei ist eine visuelle Schnittstelle essenziell für die Kommunikation vgl. Kapitel

hier Referenz zu Vialsierungskapitel

. Auch [Wes+20] weisen auf die Chance für Transparenz durch Human-Machine-Teams hin. Als weiterführende Literatur nennen sie hier z.B. das Modell für Human-Machine-Teams nach [Lyo13], welches sich auf gemeinsame Absichten von Mensch und Maschine konzentriert. Weiter nennen sie noch [Che+14], welche das SAT(*situation awareness-based agent transparency*)-Modell entwickelt haben, bei dem das Bewusstsein für konkrete Situationen weiter in den Fokus gerückt wird.

3.2.1.3 Mentale Modelle

Im Rahmen der Literaturrecherche konnte herausgestellt werden, dass mentale Modelle, welche sich Nutzer über den Algorithmus bilden, wichtig für das Verständnis sind. Ein mentales Modell besitzt nach [Kul+13] unterschiedliche Ausprägungsformen, so kann zwischen dem strukturellen und dem funktionalen mentalen Modell unterschieden werden. Bildet sich eine Person ein strukturelles mentales Modell, gewinnt sie Erkenntnisse über die Funktionsweise eines Systems. Ein funktionales mentales Modell, hilft Menschen dabei ein System zu nutzen, während ein Verständnis über das System nicht notwendig ist.

3.2.1.4 Facetten der Transparenz

[VW20] stellen unterschiedliche Bausteine heraus, aus denen sich Transparenz zusammensetzt. So sei Traceability (dt.: Rückverfolgbarkeit) ein Teil von Transparenz, welcher die Dokumentation der gesteckten Ziele, die angewendeten Definitionen sowie das Systemdesign und die während der Entwicklung getroffenen Annahmen mit einschließt. Des Weiteren spielt die Kommunikation über den Zweck aber auch die Grenzen der AI eine große Rolle. Als dritten Baustein nennen sie Intelligibility (dt.: Verständlichkeit), welche Verständnis und Überwachung für Beteiligte

ermöglichen soll. Um Verständlichkeit zu erreichen, spielen XAI-Methoden eine wesentliche Rolle.

[HT21] fächern Transparenz anders auf und gliedern Transparenz in zwei Unteraspekte. Zum einen gibt es die funktionale Transparenz, welche beschreibt, wie ein Algorithmus funktioniert, und des Weiteren die Transparenz in Bezug auf die genutzten Daten.

Eine weitere Facette von Transparenz wurde von [SZI20] herausgestellt. So beantworte Transparenz eher die Fragen, ob Daten legitimiert genutzt wurden und gibt weniger Auskunft darüber, ob der Algorithmus gut geeignet ist oder im mathematischen Sinne optimiert wurde.

[SW18]Transparenz und Erklärbarkeit wirken sich positiv auf Vertrauen aus.

Dieser Satz ist noch etwas einsam, vlt. solche Aspekte rauswerfen

3.2.2 Was transparent machen?

3.2.2.1 Inhaltlicher Kontext

[Sen+20] stellten in ihrer Studie im medizinischen Umfeld heraus, solche Informationen mitzugeben, welche für das Publikum relevante Aspekte aufzeigen. So sollten sich Erklärungen auf den Kontext beziehen und sich auf Problemlösungen fokussieren [Sen+20].

3.2.2.2 Rahmenbedingungen und sozio-technischer Kontext

[Cai+19] untersuchten im medizinischen Kontext, welche Informationen von Mitarbeitenden vor der Einführung eines ML-Systems gewünscht werden. Oberkategorien dieser gewünschten Informationen sind Möglichkeiten und Grenzen, Funktionalitäten, Subjektivität, Design-Ziele, Gedanken vor dem Einsatz.

- In Bezug auf Möglichkeiten und Grenzen des Systems sollte kommuniziert werden, wie die grundsätzliche Performance des Algorithmus ist und worin dieser gute oder eben weniger gute Leistungen erbringt. Dies kann mithilfe von Leistungsmetriken angeben werden, in denen verdeutlicht wird, wie viele Datensätze richtig oder falsch klassifiziert wurden. Weiter sei für Nutzer interessant, wie sich der einzusetzende Algorithmus bei Fällen verhält, welche häufiger selbst von Menschen falsch eingeordnet werden. Auch sind die theoretischen Grenzen von KI grundsätzlich wichtig. Darüber hinaus sind die Trainingsdaten von Interesse. Hierbei ist die Diversität und Menge der Trainingsdaten wichtig, während sich auch das Datenformat (numerisch, binär, Bilder) von Bedeutung ist.
- In Bezug auf die Funktionalität wünschen sich Nutzer eine konkrete Beschreibung, über das, was der Algorithmus leistet. Hierbei steht zum einen der fachliche Kontext im Vordergrund, so nennen [Cai+19], dass im medizinischen Kontext z.B. darauf hingewiesen werden könnte, ob ein Algorithmus „nur“ dazu in der Lage ist Krebs zu erkennen oder noch weitere Krankheiten entdecken kann. Daneben ist auch der technische Kontext nicht uninteressant, so sei anzumerken, wie sich die Funktionsweise des Algorithmus im Vergleich zu der des Menschens unterscheidet. Für Bildklassifikation sei hier beispielsweise die Frage danach genannt, ob ein Algorithmus das Bild im Ganzen betrachtet oder ein Bild zunächst in einzelne Komponenten aufteilt und diese dann untersucht. Weiter ist

von Interesse aus welcher Datenbasis heraus der Algorithmus die Entscheidung trifft. Des Weiteren ist bedeutsam für die medizinischen Praktiker, ob auch andere Daten des gleichen oder anderer Patienten in die Entscheidungsfindung miteinflossen.

- Da in der Praxis die Subjektivität einzelner Mediziner eine große Rolle spielt, ist die Subjektivität des Algorithmus wichtig für spätere Nutzer des ML-Systems. So ist es von Bedeutung Grenzfälle aufzuzeigen, welche auch von Menschen unterschiedlich bewertet werden. Hier interessiert die Praktiker der fachliche Hintergrund des Systems. Während der Einführung sollte außerdem die Möglichkeit bestehen, dass menschliche Praktiker ihre eigenen Diagnosen mit der der KI abgleichen können, um diese somit besser einschätzen zu können.
- Neben stark auf den Algorithmus selbst bezogenen Aspekte sprechen Nutzer auch den Design-Zielen eine hohe Bedeutung zu. So sei interessant, was der Nutzen des Systems in Bezug auf Kosten und Effizienz sei. Daneben ist wesentlich, ob die KI mit Menschen arbeiten soll oder unabhängig handelt.
- Weiterführend sind Gedanken vor dem Einsatz transparent zu machen. Diesbezüglich sollte kommuniziert werden, ob die rechtlichen und regulatorischen Rahmenbedingungen abgeklärt wurden oder ob eine behördliche Prüfung des Systems noch aussteht. Daneben wünschen sich Nutzer zu wissen, ob das Tool bereits schon von anderen Organisationen eingesetzt wird oder ob es im wissenschaftlichen Kontext auf seine Funktionweise hin überprüft wurde. Daneben stellt sich die Frage, ob es sich auf die täglichen Arbeitsabläufe der Praktiker auswirkt.

Ähnlich wie [Cai+19] wurde eine Studie von [Vor18] durchgeführt, die untersuchte, was Menschen über das System wissen wollen, damit sie es als transparent erachten. Hierbei erarbeiteten [Vor18] Faktoren und Beispielfragen in Bezug auf diese Faktoren. Zusätzlich erhoben die Autoren, wie wichtig einzelne Faktoren für Transparenz aus Nutzersicht wirken. In folgender Aufzählung[5] sind die Aspekte zusammengefasst.

- Technische Faktoren:
  - Daten (Wichtig): Wie aktuell sind die Daten, auf denen eine Entscheidung beruht?
  - Unsicherheit (mittel): Trifft das System Annahmen aufgrund von unsauberen Daten?
  - Logik (Wichtig): Was spricht für oder gegen die Entscheidung eines Systems? Woher kommt eine Entscheidung des Systems?
  - Zuverlässigkeit (mittel): Unter welchen Bedingungen lag das System in der Vergangenheit falsch?
- Persönliche Faktoren:
  - Personalisierung (mittel): Wurde diese Empfehlung speziell für einen bestimmten Nutzer ausgesprochen?
  - Vertrauen (Wichtig): Wie wird Vertrauen dieses System gemessen?
  - Nachvollziehbarkeit (weniger wichtig): Was geschieht, wenn ein Nutzer die Entscheidung des Systems ablehnt?

5. entnommen aus [Cai+19], übersetzt mit kleinen Ergänzungen aus dem Fließtext des Papers

- Soziale Faktoren:
  - Soziales Filtern (weniger wichtig): Wie ähnlich ist eine konkrete Person anderen Personen, welche auch diese Empfehlung erhalten haben?

Ergänzen nennen [Vor18] noch weitere Aspekte, die da wären:

- Externe Abhängigkeiten: Mit welchen Daten arbeitet das System noch?
- Kritikalität: Die Bedeutung von Transparenz nimmt zu, wenn die Entscheidungen des Systems einen selbst oder andere beeinflussen.
- Rolle des Nutzers: Welche Informationen von Interesse sind hängt stark von der Rolle des jeweiligen Betrachters ab. So haben Entwickler, fachkundige Endnutzer sowie sekundäre Nutzer einen unterschiedlichen Fokus.

Diese zwei umfassenden Studien werden von anderen Wissenschaftlern weiter ergänzt und untermauert. So merken [ZC18] an, dass das Kommunizieren von Unsicherheiten des Systems einen großen Teil zur Transparenz beitragen. Dies gilt jedoch nicht nur für Endnutzer des Systems, sondern nach [Gom+21] sei auch für Modellentwickler interessant, was die Stärken und Schwächen des Modells sind, was auch durch das Aufzeigen von Fehlerfällen ausgedrückt werden kann. [Iri22] sprechen die Empfehlung aus nicht nur den Algorithmus, sondern auch soziale Aspekte und das gesamte System in die Kommunikation miteinzubeziehen. [Gol19] ergänzt, dass Transparenz auch Klarheit bei der Beschaffung und Umsetzung bedeute. Weiter sind Einfluss von Entscheidungen während der Nutzung von ML-Systemen von Bedeutung.

3.2.3 Daten

3.2.3.1 Bias

Bias/oder auch den ersten Teil des folgenden Abschnitts in mein Grundlagenkapitel mitaufnehmen?

Bias in Daten ist ein großes Problem und kann sich negativ auf Fairness, aber auch auf die Funktionweise des ML-Algorithmus auswirken. Es existieren unterschiedliche Arten von Bias, welche an dieser Stelle kurz zusammengefasst sind [SG21]:

- Historischer Bias: Diese Art von Bias kann selbst dann entstehen, wenn Trainingsdaten optimal erhoben wurden. Schaden durch historischen Bias entsteht auch bei einer exakten Darstellung der realen Welt, jedoch existieren in dieser Stereotypen, welche im späteren Verlauf durch die Anwendung des Algorithmus weiter verstärkt werden könnten.
- Representation Bias ist eine Art des Bias, welche einen Teil der Bevölkerung unterrepräsentiert.
- Measurement Bias: Dieser Typ zielt auf Probleme bei der Auswahl, der Anhäufung und der Berechnung von Merkmalen und Zielvariablen ab. Dies kann durch begrenzte Möglichkeiten in der realen Welt bedingt sein, Daten exakt messen zu können.
- Aggregation Bias: Entitäten, welche in der Realität eigentlich unterschiedlich behandelt werden sollten, werden jedoch mit dem gleichen Modell behandelt.

- Learning Bias zielt auf falsche Modellierungsentscheidungen ab. So wird beispielsweise nicht die richtige Zielfunktion ausgewählt.
- Evaluation Bias: Die für die Evaluation genutzten Daten erfüllen keine oder andere Qualitätskriterien und -maßstäbe als die für das Training verwendeten Daten.
- Beim Deployment Bias steht die Problemlösung des Modells im Vordergrund. Das Modell erarbeitet seine Vorhersage auf eine andere Weise, als dies normalerweise getan wird, weil das Modell bspw. den sozialen Kontext unberücksichtigt lässt.

Um Bias zu erkennen, haben [SG21] ein Framework entwickelt, welches es mithilfe von Formularisierung ermöglichen soll, Bias zu erkennen. [SG21] verstehen den Datenverarbeitungsprozess als Aneinanderreihung von Mapping-Funktionen. Leider geben die Autoren keine Checkliste oder ähnliches an die Hand, um Bias erkennen zu können, weisen jedoch auf Aspekte hin, welche dabei helfen können. In folgender Aufzählung ist dies kurz zusammengefasst:

- Historischer Bias: Wie sind Merkmale und Zielvariablen über die komplette Bevölkerung verteilt?
- Measurement Bias: Wie werden die Merkmale und Zielvariablen erstellt?
- Representation Bias: Kommt es zu Änderungen des Modells bei anderer Zielfunktion oder anderen Merkmalen?
- Aggregation Bias: Kommt es zu Änderungen durch das Wählen einer komplexen Zielfunktion oder anderen Trainingsdaten?
- Learning Bias: Kommt es zu Änderungen durch ein anderes Modell?
- Evaluation Bias: Kommt es zu Änderungen, wenn mehr als ein Datensatz oder mehrere Darebsätze zur Überprüfung genutzt wird?
- Deployment Bias: Dieser Typ ist schwer zu erkennen und aufzuheben, da dieser nur durch Experten aufgedeckt werden kann.

3.2.3.2 Dokumentation der Daten

Für die Dokumentation der Daten werden häufig sogenannte Datasheets empfohlen. Datasheets verfolgen den Zweck getroffene - eventuell unbemerkt getroffene - Annahmen aufzudecken. Diese Datasheets können darüber hinaus dabei helfen Fairness zu gewährleisten und die Leistung des ML-Systems zu optimieren [VW20]. Beispiel für diese Datasheets sind z.B. nach [VW20] Google Facets und Datasheets for Datasets. Google Facets ist ein Visualisierungstool, was es ermöglicht Daten auf unterschiedlichen Granularitätsebenen zu untersuchen [Goo22]. Datasheets for Datasets, entwickelt von [Geb+21], sind Datenblätter, welche für jeden Datensatz erstellt werden sollten und für die Dokumentation von Zusammensetzung, Sammlung, Motivation, der geplanten Verwendung, Verteilung und Pflege eingesetzt werden soll. [Gra+20] haben Datasheets for Datasets etwas abgewandelt für den Bereich der Biologie.

Hier gehe ich noch genauer auf die Datasheets-Arten ein und zeige vlt. Beispiele

Merkmale, welche sich bereits in diesen Datasheets wiederfinden lassen, werden auch von anderen wissenschaftlichen Arbeiten herausgestellt. So betont [De 18], dass für die Transparenz eines Datensatzes angegeben werden sollte, welche Merkmale vorhanden sind und wie diese z.B.

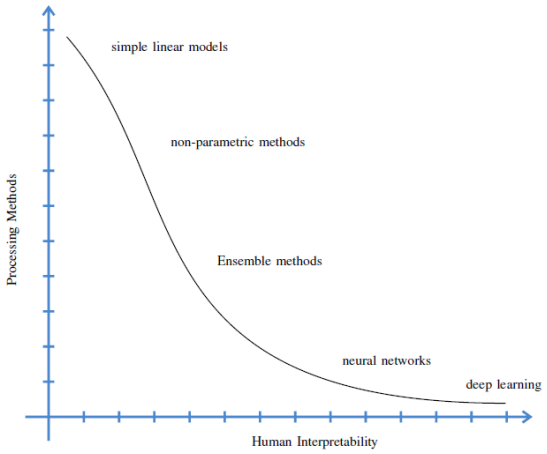


Abbildung 3.3: Interpretierbarkeit je nach Modell, entnommen aus: [Vor18]

über Frauen und Männer verteilt sind. Klassenungleichheiten sollten bei Datensätzen transparent gemacht werden.

Fairness kann auch mithilfe von Datendokumentation aufgedeckt werden, so könnte demonstriert werden wie und warum die Datensätze entstanden sind und warum ein konkreter Datensatz überhaupt ausgewählt wurde [Den+20]. In Bezug auf Fairness betonen jedoch [De 18], dass es schwierig ist diskriminierende Merkmale überhaupt zu erkennen.

3.2.3.3 Dokumentation der Datenverarbeitung

Bevor ein Datensatz überhaupt genutzt werden kann, muss dieser häufig zunächst bereinigt werden. [GM22] empfehlen hier auch den Prozess der Bereinigung zu prüfen und auch festzuhalten, wie oder ob auf Bias in den Daten geprüft wurde. Beispielsweise kann angegeben werden, ob Zielvariablen korrigiert wurden.

3.2.3.4 Transparenz über gleiche Datenformate

[HT21] empfehlen für den leichteren Überblick das Nutzen eines einheitlichen Datenformats. Hier nennen sie JSON oder XML als Optionen, wenngleich sie XML den Vorzug geben, da diese vollständige Schemata sowie Standardtransformationen ermöglicht.

3.2.4 Transparenz durch das gewählte Modell

Einige ML-Algorithmen gelten als in sich transparent. So liegt es auf der Hand, dass diese für transparentes ML eingesetzt werden können. Abbildung 3.3 gibt einen guten Überblick, welche dabei hilft einzuordnen, ob ein Algorithmus leicht von Menschen verstanden werden kann.

3.2.4.1 Transparente Modelle

Folgend wird aufgelistet, welche Methoden als transparent gelten mit Bezug auf die jeweiligen Quellverweise.

- Lineare Regression [HZW21; Che+21; Pal+22; RSG16]
- Logistische Regression [HZW21; Sch22; Kra+17; Pal+22; Hil+18]
- Entscheidungsbäume [HZW21; Che+21; Kra+17; De 18; Pal+22]
- Random Forest [Che+21; RSG16]
- Entscheidungsregeln (Wenn-Dann-Aussagen) [HZW21; Che+21]
- Generalized Linear Models (GLM) [HZW21]
- Generalized Linear Rules [HZW21]
- Generalized Additive Models (GAMs) [HZW21]
- Bayesche Regeln [De 18]
- k-nächste Nachbarn [Pal+22]

In Bezug auf transparente Modelle sei angemerkt, dass einige dieser Methoden teilweise transparenter sind als andere. So geben [Hil+18] an, dass logistische Regression leichter nachzuvollziehen sei als Random Forest, wenngleich Random Forests bessere Ergebnisse liefern. Ein weiterer wichtiger Aspekt ist, dass transparente Modelle (konkret von den Autoren genannt: logistische Regression und Entscheidungsbäume) zwar transparent seien, aber schwer zu generalisieren sind.

Daneben existieren noch weitere Ansätze von Wissenschaftlern, welche ihre eigenen transparenten Modelle entwickeln. Diese basieren jedoch meistens auf den Mechanismen, welche sich in den Methoden der obigen Listen wiederfinden lassen. Ein Beispiel für ein solches Modell sind fast-and-frugal-trees [Kel+20], dessen Funktionsweise auf Entscheidungsbäumen basiert. Als besonderer Vorteil wird hier von den Autoren genannt, dass sich diese Art von Bäumen leicht visualisieren lässt und kein Hinzufügen nachträglicher Erklärungen (wie es bei Post-Hoc-Methoden der Fall ist) nötig ist. Somit ist eine Entwicklung mit einer solchen Methode zusätzlich weniger aufwendig. Auch [ALW09] entwickelten ein eigenes transparentes Klassifikationsmodell, indem sie mehrere transparente Methoden kombinieren, z.B. neuronale Netze und Wenn-Dann-Regeln. Diese erlauben eine transparente Abbildung des neuronalen Netzes. Dieses ist streng genommen keine neue Methode, sondern eine Anwendung von Global Surrogate

Todo: Referenz auf Abschnitt im Kapitel der XAI-Methoden

3.2.5 Modellauswahl

Wie vorangegangen beschrieben existieren auch im ML-Bereich transparente Methoden. Um ein ML-Modell somit transparent zu machen, läge es also nahe einfach ein transparentes zu wählen, jedoch gibt es für diesen Ansatz Pro- und Contraargumente. Wird ein interpretierbares Modell gewählt, stößt man eventuell auf Grenzen, welche sich besonders bei nichtlinearen, komplexen Zusammenhängen oder einem Problemraum mit besonders vielen Merkmalen in den Trainingsdatensätzen zeigen [HZW21]. Andere Autoren sehen jedoch in der Praxis eine andere Herangehenweise als zielführend an. Einfache Modelle funktionieren nicht unbedingt schlechter in der Realität. Vor allem in Branchen wie der Medizin oder Strafverfolgung, in denen Menschen das System gut verstehen sollten, kann es wünschenswert sein eine höhere Interpretierbarkeit als eine besonders gute Genauigkeit zu implementieren [VW20; Kel+20].

3.2.6 Erklärungen

XAI- oder Visualisierungsmethoden können unterschiedliche Arten von Erklärungen produzieren.

[HAP21] haben in einer Studie in der Luftfahrtbranche untersucht, welche Art von Erklärungen sich am besten eignen, um Vertrauen zu schaffen. Folgende Auflistung stellt die Ergebnisse da, wobei solche Erklärungen, welche am ehesten Vertrauen in die Technologie schaffen, in absteigender Reihenfolge genannt werden:

- Wichtigste Faktoren, welche die Vorhersage beeinflussen
- Konkrete Beispiele für die Vorhersage
- Visuelle Darstellung der Funktionsweise
- Kausale Erklärungen (Welche Änderungen an den Eingabedaten oder am Algorihmus selbst würde eine andere Vorhersage ergeben?)
- Erläuterung des Lebenszyklus und des Entwurfs
- Kontrafaktische Erklärung (Warum ist die Vorhersage „A“ und nicht „B“?)

Wenngleich [HAP21] herausgefunden haben, dass sich zumindest für das Vertrauen kontrafaktische Erklärungen weniger eignen, stellen [TR21] heraus, dass diese Art der Erklärungen Erkenntnisse liefern können, welche bei der praktischen Umsetzung während der Programmierung hilfreich sein können. Inbesondere liefern kontrafaktische Erklärungen ein gutes Verständnis über die Ursache-Wirkung-Beziehung.

3.2.7 Einordnung von XAI-Methoden

Für das Einordnen von XAI-Methoden existieren viele Ansätze, welche sich teilweise unterscheiden, aber auch Dinge gemein haben.

[LPK20] unterscheiden zwischen vier Oberkategorien mit mehreren Unterkategorien.

- Zweck der Interpretierbarkeit (Anwendung eines White-Box-Modells, Post-hoc-Erklärungen eines Black-Box Modells, Sicherstellung von Fairness, Prüfen der Sensitivität von Vorhersagen)
- Modellspezifisch vs. Modellagnostisch
- Datentypen (Tabellen, Text, Bilder, Graphen)
- Lokal vs. Global

Hier schreibe ich dann, wie ich die einordne, wenn ich weiß, wie ich das machen will

3.2.8 XAI-Methoden für transparente Methoden

Wenngleich transparente Modelle per Definition keiner Erklärung bedürften, wurden in der Literatur jedoch einige Methoden gefunden, welche zur Interpretation dieser eingesetzt werden können. Im folgenden Kapitel werden diese je Algorithmus präsentiert.

3.2.8.1 Lineare Regression

[Wes+20] empfehlen für das Erklären des Outputs linearer Modelle die Hervorhebung der wichtigsten Eingabeparameter und deren relativen Bedeutung zueinander.

3.2.8.2 Entscheidungsbäume

[Gui+18b] empfehlen für die Erklärung von Entscheidungsbäumen eine Visualisierung dieser oder alternativ eine textuelle Erklärung in Form von Wenn-Dann-Regeln. Dies kann sowohl für lokale als auch globale Erklärungen angewendet werden.

3.2.8.3 Random Forest

Für das Erklären von Random Forests können Entscheidungsbäume und Wenn-Dann-Regeln eingesetzt werden. Hierbei kann ein Entscheidungsbaum globales Verständnis über den Random Forest bieten, während die Regeln der ungefähren Abbildung der Entscheidungen der einzelnen Bäume dienen sollen [Che+21].

3.2.8.4 Sonstiges

Die vorliegende Masterarbeit orientiert sich aus den in Kapitel 2.1.4 genannten Gründen nur an einer Auswahl an ML-Methoden, für die XAI-Ansätze gesucht werden. In der Literaturrecherche wurden jedoch für andere Verfahren auch XAI-Methoden gefunden. Der Vollständigkeit halber werden diese hier aufgelistet. Auf weitere Erläuterungen wird jedoch verzichtet.

- GAM
  - Gamut:interaktive Visualisierung mit der Zielgruppe: Data Scientists [Hoh+19]

3.2.9 Modellagnostische Methoden - lokal

Was noch fehlt: Für welche Art von Problemen funktionieren diese Algorithmen? LIME nur für Klassifikation, aber die anderen? Vor- und Nachteile ausbauen...

Folgendes Unterkapitel fasst eine Auswahl an Methoden zusammen, welche als modellagnostisch und lokale Ansätze einzuordnen sind (für eine Erklärung der Begriffe siehe Kapitel 2.1.5.1). Alle diese Ansätze, da sie modellagnostisch sind, haben gemeinsam, dass keine Zugriff auf die Modellstruktur notwendig ist. Es muss lediglich möglich sein Vorhersagen für Dateninstanzen abrufen zu können.

3.2.9.1 LIME

Die Methode *LIME*, entwickelt von Ribeiro et al. [RSG16], wurde in der betrachteten Literatur am häufigsten genannt (sieht z.B.: [Wei+19b], [Gol19], [TR21], [TG20], [HZW21], [Kra+17], [Gom+21], [Wan+20], [Str19], [Wes+20], ...) LIME steht für *Local Interptetable Model Agnostic Eplanations* und erklärt einzelne Vorhersagen eines jeden Klassifikationsalgorithmus, indem ein interpretierbares Modell lokal um eine zu erklärende Instanz herum gelernt wird [RSG16]. Diese Erklärtechnik wird als lokaler Stellvertreter (eng.: *Local Surrogate*) bezeichnet [Mol22].

Konzept

Bei LIME kommt es zu einer lokalen Annäherung an ein Black-Box-Modell (f) durch ein zu interpretierbares Modell (g). Nach Ribeiro et al. [RSG16] setzt LIME daher eine Abwägung zwischen Genauigkeit und Interpretierbarkeit voraus. Beispiel für interpretierbare Modelle sind in Kapitel 3.2.4.1) zu sehen. In Abbildung 3.4 sind die zwei Modelle f und g sowie verschiedene Instanzen zu sehen. Die rosa und die blauen Flächen demonstrieren die Funktionsweise des Black-Box-Modells, welche komplex und nicht-linear ist. Ziel von LIME ist es nun, zu erklären, warum eine bestimmte Instanz (das große rote Kreuz in der Mitte der Abbildung) in die Klasse „rosa“ einsortiert wurde. Die LIME-Methode konzentriert sich nun als lokaler Ansatz auf den Bereich, um diese zu erklärenden Instanz. Um die zu erklärende Instanz herum werden nun zufällig weitere Datenpunkte erzeugt (Pertubationen). Dies ist durch die blauen Punkte und roten Kreuze dargestellt. Für diese Pertubationen, wird nun die Zielvariable mithilfe des Black Je näher sich diese Pertubationen an der zur erklärenden Instanz befinden, desto stärker fallen sie ins Gewicht. Nun wird sich mithilfe eines interpretierbaren Modell g an dieser Instanz dem Black-Box-Modell f angenähert. g kann nun von Menschen interpretiert werden [RSG16]. In der Abbildung würde ein Mensch also sehen, dass alles Werte links der gestrichelten Linie der Klasse „rosa“ zuzuordnen sind. Alle Werte rechts der Linie, würden als „blau“ klassifiziert werden. Hier wird noch einmal deutlich, dass LIME lediglich lokale Erklärungen erzeugt, denn Instanzen, welche sich auf dem Bild ganz links, weit entfernt der Entscheidungslinie befinden, gehören der blauen Klasse an. So ist die lokale Erklärung hier richtig, aber lässt global falsche Schlüsse zu.

Formale Betrachtung Ziel von LIME ist das Finden des lokalen Stellvertreters, der lokal die eine bestimmte Instanz x erklärt. Dieses Modell sollte so einfach wie möglich sein. Jedoch sind nicht alle Modelle gleich verständlich. So gibt es unterschiedliche Instanzen g aus der Klasse G, die alle Instanzen von z.B. linearen Funktionen enthält. G kann jedoch theoretisch auch die Menge aller Entscheidungsbäume sein [RSG16]. So führen Ribeiro et al. [RSG16] den Faktor $\Omega(g)$ ein. Dieser misst die Komplexität eines interpretierbaren Modells, dies kann bei

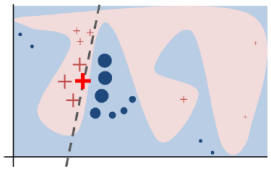


Abbildung 3.4: Funktionsweise von LIME, entnommen aus: [RSG16]

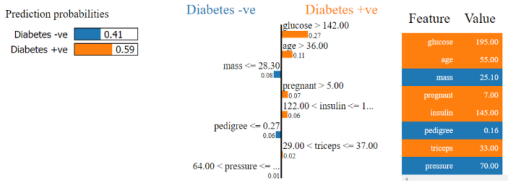


Abbildung 3.5: Erklärung der Klassifikation einer bestimmten Instanz, entnommen aus [Rao20]

Entscheidungsbäumen beispielsweise die Tiefe eines Baums sein oder bei linearen Modellen die Anzahl an Gewichten. Nach Ribeiro et al. [RSG16] kann nun eine Erklärung für den Datenpunkt x durch folgendes Optimierungsproblem gefunden werden:

$$\xi(x) = \underset{g \in G}{argmin} \mathscr{L}(f, g, \pi_x) + \Omega(g) \tag{3.1}$$

$\mathscr{L}(f, g, \pi_x)$ gibt an, dass eine gute Abschätzung an das Black-Box-Modell f gefunden werden soll, welches sich in der Nachbarschaft π_x befindet. Des Weiteren soll das gefundene Modell g so einfach wie möglich sein, was durch $\Omega(g)$ angegeben ist [RSG16]. Z.B. kann $\Omega(g)$ ausdrücken wie viele Merkmale von der interpretierbaren Funktion g erfasst werden soll [Mol22]. Um nun ein gutes Modell g zu finden, muss der erste Therm der Formel optimiert werden. Hierfür wird sich die Nachbarschaft angesehen und dort lokal ein Modell gefunden. Ziel ist es das Modell mit der höchsten Genauigkeit zu finden. Dies geschicht, indem für alle Datenpunkte in der Nachbarschaft aus Abbildung 3.4 sowohl das interpretierbare Modell g als auch das Black-Box Modell f aufgerufen wird. g wird nun daraufhin optimiert, so wenig Fehler wie möglich zu haben, was mit den Vorhersagen des Black-Box-Modells überprüft werden kann. Bei diesem Optimierungsproblem fallen Punkte, die näher an x liegen stärker ins Gewicht.

Anwendung von LIME

Ein Beispiel für eine mit LIME erzuegte Erklärung ist in Abbildung 3.5 dargestellt. Hier wurde anhand verschiedener Werte untersucht, ob eine Person Diabetes hat. Hier wurde eine Person mit einer Wahrscheinlichkeit von ungefähr 60 (links dargestellt) als Diabetes-Patient eingeordnet. Diese Entscheidung wird vor allem durch die orange dargestellten Merkmale unterstützt, so z.B. das Alter. Blau markiert sind solche Merkmale, welche eher für einen gesunden Patienten sprechen würden. Rechts sind zur Übersicht die Merkmalswerte aufgezeigt. Zur praktischen Umsetzung von LIME stellen die Erfinder eine Bibliothek[6] zur Verfügung. LIME kann für jeden Klassifikationsalgorithmus auf tabellarischen Daten, Text oder auch Bilder angewendet werden. Weiter kann durch den Entwickler ausgewählt werden, wie viele Merkmale dargestellt werden sollen [Mol22]. Je nach dem, welche Daten dem Klassifikationsalgorithmus zu Grunde liegen, kommt es zu leichten Unterschieden in der Anwendung. So muss zwischen Merkmalen, mit

6. https://github.com/marcotcr/lime

denen ein Algorithmus arbeitet und Merkmalen, welche für Menschen interpretierbar sind, unterschieden werden [RSG16]. Ribeiro et al. [RSG16] demonstrieren dies mit einem Beispiel aus der Texklassifikation. Ein binärer Vektor, der beschreibt, ob sich ein bestimmtes Wort in einem Text befindet und somit zur Klassenvorhersage führte, ist für den Menschen interpretierbarer als die Datenstruktur, welche durch den Algorithmus in Wahrheit (z.B. Wordembeddings) zur Bearbeitung der Klassifikationsaufgabe genutzt wird [RSG16]. Bei Textklassifikation werden nun Permutationen dadurch erzeugt, dass zufällig Wörter aus einem Text etfernt werden. Die Klasse wird danndamit erklärt, ob oder wie oft ein bestimmtes Wort in einem Text vorkam.

Vor- und Nachteile

Die Formel von LIME kann mit unterschiedlichen ML-Methoden von f und g genutzt werden [RSG16]. Dies bedeutet, das zugrundeliegende Black-Box Modell kann ausgetauscht werden, während der lokale Stellvertreter g nicht geändert werden muss. Auch kann ein lokaler Stellvertreter genutzt werden, welchen die Erklärungsbetrachter gut verstehen, weil schon Vorwissen vorhanden ist (z.B. Entscheidungsbäume) [Mol22]. Ribeiro et al. [RSG16] stellen in ihrer Veröffentlichung besonders heraus, dass LIME Vertrauen in ML-Algorithmen erhöhen kann und zur Erkennung von unerwünschten Verhalten beiträgt. So weisen sie auf einen Klassifikationsalgorithmus hin, welcher Fotografien von Wölfen und Huskys unterscheiden sollte. LIME ermöglichte es, aufzudecken, dass dieser Algorithmus solche Bilder mit Schnee im Hintergrund, der Klasse Husky zuordnete und die Tiere selbst gar nicht beachtete [RSG16].

Nachteil ist die Schwierigkeit bei tabellarischen Daten, die Nachbarschaft zu bestimmen [Mol22]. Weiter entdeckten Slack et al. [Sla+20] eine Möglichkeit, dass sich Erklärungen manipulieren lassen um Bias in den Daten zu verstecken, was zu Vertauensverlust führen kann. Daneben werden die Daten immer zufällig generiert, sodass gleiche Instanzen unterschiedliche Erklärungen erhalten können. Dieses Problem löst eine Abwandlung von LIME. Zafar und Khan [ZK19] entwickelten DLIME (*Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems*). Hier werden die Pertubationen anders generiert (agglomeratives hierarchisches Clutering) und für die Auswahl der sich der zur erklärenden Instanz am nächsten befindlichen Datenpunkte den in Kapitel 2.1.4.5 beschriebenen KNN-Algorithmus nutzen.

3.2.9.2 SHAP

Auch die modellagnostische Methode *SHAP* wurde von vielen Autoren genannt ([TR21], [VW20], [Sch22], [HZW21], [Gom+21], [Wan+20], [Pal+22], ...). Diese Methode wurde zuerst von Lundberg und Lee [LL17] vorgestellt und basiert auf den sogenannten SHAPLEY-Werten aus der Spieltheorie. SHAPLEY-Werte beantworten die Frage, was ein einzelner Spieler innerhalb einer Gruppe beitrug. Angewendet auf das Forschungsfeld des MLs können einzelne Spieler als Merkmalswerte betrachtet werden[Wan+20].

Konzept

Die Shapley-Werte wurden von Shapley [Sha51] eingeführt. Gegeben ist eine Gruppe von Spieler, welche gemeinsam einen Wert v erzielen. In dem Beispiel von Molnar [Mol22] ist dies ein Preis für eine Wohnung. Verschiedene Spieler oder im ML-Kontext Merkmale wirken sich unterschiedliche auf einen Wohnungspreis aus. Beispielhafte Merkmale können hier das Stockwerk, die Lage oder das Verbot von Haustieren sein. Um nun den Beitrag eines Merkmals zu berechnen, kann nun getestet werden, welchen Wohnungspreis ohne ein bestimmtes Merkmal

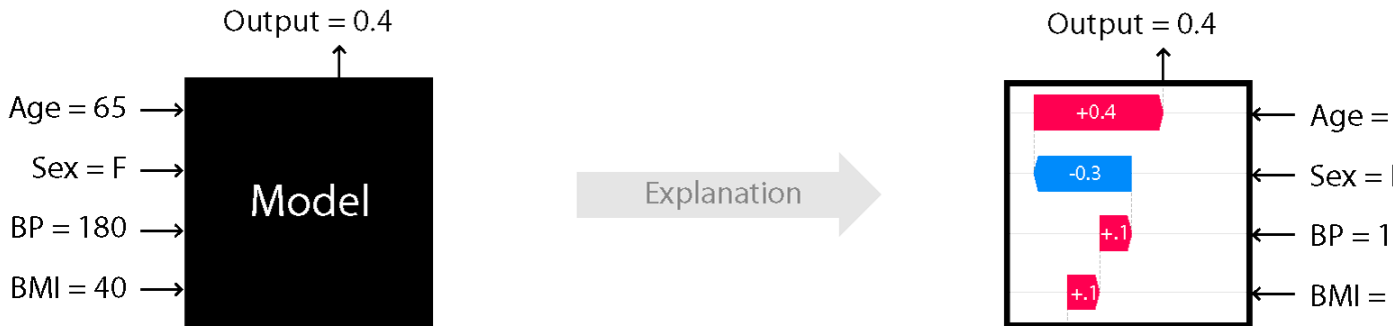


Abbildung 3.6: Beispiele Erklärungen mit SHAP, entnommen aus der SHAP Bibliothek

erzielt werden würde. Nun wird dieses Merkmal, z.B. das Haustierverbot, ausgeklammert und der Wohnungspreis berechnet. Dies muss für alle Kombinationen, wo das Haustierverbot fehlt, geschehen, um Abhängigkeiten zwischen Merkmalen zu erkennen. Nun wird die Differnez zwischen dem ursprünglichen Wert V und dem neuen Ergebnis berechnet. Über alle Ergebnisse wird der Durchschnitt gebildet. Dies geschieht für alle anderen Merkmale analog. Somit erhält man die Shapley-Werte für alle Merkmale [Mol22].

Formale Betrachtung textcitelundberg2017unified wenden dieses Konzept in ihrer Methode (SHAP (*SHAPLEY ADDITIVE EXPLANATIONS*)) an, um einzelne Vorhersagen zu erklären, indem Beiträge eines jeden Merkmals berechnet werden. Sei x die zu erklärende Instanz und $f(x)$ das Black-Box-Modell. Zu x existieren noch vereinfachte lokale Eingabe x'. Dieses muss eingeführt werden, um die Werte der Merkmale zu vereinfachen, so werden sie z.B. als binäre Vektoren dargestellt. Weiter existiert für das Black-Box-Modell ein Modell g, für das gilt $g(x') \approx f(x)$. g sieht weiter wie folgt aus und gibt an, was ein Merkmal zu einer Vorhersage einer Instanz x' beiträgt (vereinfacht von [Gia21] nach [LL17]):

$$g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{3.2}$$

Mit ϕ_0 ist der sogenannte *Null-Output* und meint den durchschnittlichen Output des Modells, den von den einzelnen Merkmalen unabhängig ist. i ist ein Merkmal und M ist die Anzahl aller Merkmale. ϕ_i ist der Erklärungseffekt eines Merkmals i und gibt an, wie viel dieses Merkmal zu einer Veränderung der Vorhersage beiträgt [LL17].

Anwendung Lundberg und Lee [LL17] stellen eine Bibliothek[7] für die Nutzung von SHAP zur Verfügung. Dort ist auch aufgelistet, worfür SHAP verwendet werden kann, z.B. für Klassifikation oder Regression. Ein Beispiel für eine mit SHAP generierte Erklärung ist in Abbildung 3.6 dargestellt. Hier wird die Ausgabe des Wertes 0,4, welche durch ein Black-Box-Modell erstellt wurde, erklärt. Merkmale, welche die Vorhersage in einen höheren Wert ändern, wie z.B. das Alter, sind rot eingezeichnet. Blau eingezeichnete Merkmale, wie z.B. das Geschlecht würden den Wert der Vorhersage verringern. Ein praktisches Problem mit SHAP besteht in der Berechnung der SHAPLEY-Werte. Würden 4 Features vorliegen, müssen 16 verschiedene Möglichkeiten für die Exkludierungen der Features berechnet werden. Bei 32 Features, wären dies schon 17,1 Milliarde Berechnungen. Hierfür haben Lundberg und Lee [LL17] den Shapley Kernel entwickelt, welcher sich SHAPLEY-Werte annähert und weniger Datensätze erstellen muss. Für Kernel SHAP definieren die Autoren auch Sonderformen, wie z.B. Deep SHAP für DL, welche sich die Funktionsweise der zugrunde liegenden ML-Modelle zunutze machen.

7. https://shap.readthedocs.io/en/latest/index.html

SHAP kann weiterhin auch für globale Erklärungen verwendet werden, wenn diese für jede Instanz angewendet wird [Mol22].

Vor- und Nachteile Nach Lundberg und Lee [LL17] grenzen sich SHAP-Erklärungen von anderen Erkläransätzen ab, da sie drei wichtige Eigenschaften erfüllen: lokale Genauigkeit, Fehlen, Konsistenz. Als erste wünschenswerte Eigenschaften wird lokale Genauigkeit definiert, so muss das Erklärungsmodell g das Black-Box-Modell f approximieren, sodass auch die Ausgaben von f und g ungefähr übereinstimmen. Des Weiteren sollten auch die Werte der Erklärungen 0 für nicht vom Modell betrachtete Features sein (Eigenschaft: Fehlen). Die dritte Eigenschaft ist Konsistenz [LL17]. So sollte eine Zunahme (oder Gleichbleiben) eines Merkmalswertes bedingen, dass auch der SHAPLEY-Wert zunimmt (oder gleich bleibt). Weiter basiert SHAP auf einer soliden wissenschaftlichen Grunglage und beziehen bei ihrer Erklärung alle Merkmale mit ein. Laut Molnar [Mol22] macht sie dies zu einer geeigneten Methode, das Recht auf Erklärungen der DSGVO umzusetzen. Daneben können sowohl lokale als auch globale Erklärungen generiert werden [Mol22]. Ein Nachteil dieser Methode besteht jedoch auch in den Shapley-Werten an sich. Da sie immer jedes Merkmal beachten, sind Shapley Werte, die falsche Wahl, wenn dem Nutzer nur einige wenige Merkmale präsentiert werden sollen. Hier sei LIME besser geeignet, da hier die Reduktion auf einige wenige Merkmale möglich ist. Ein weiterer Nachteil, der jedoch auch für LIME gilt, sit die von Slack et al. [Sla+20] entdeckte Manipulationstechnik, welche angewendet werden kann, um Bias in den Daten zu verstecken. Weiter ist auch die Berechnung mit dem SHAP-Kernel recht langsam und aufwendig [Mol22].

3.2.9.3 Anchors

Die folgende Methode wurde von den selben Wissenschaftlern entwickelt, welche auch LIME vorgestellt hatten und produziert Wenn-Dann-Regeln, die „Anchors“ genannt werden [RSG18].

Konzept Der folgende Ansatz setzt auf einer Schwäche der LIME-Methode auf. Wie in 3.2.9.1 beschrieben, werden Erklärungen mittels LIME durch lokale Annäherungen an bestimmte Instanzen produziert. In der Abbildung 3.4 wurde gezeigt, dass alles lokal links der Entscheidungslinie der rosa Klasse zuzuordnen war, wenngleich sich links, weiter entfernt der Entscheidungslinie Instanzen befanden, welche als blau zu klassifizieren sind. LIME beantwortet also nicht die Frage, für welche Instanzen ihre Erklärung übertragbar ist. Dieses Problem wird von Ribeiro et al. [RSG18] allerdings mit Hilfe der Anchors adressiert. In Abbildung 3.7 sind die Funktionsweise von LIME und Anchors zusammen dargestellt. Die jeweiligen Kästchen stellen die Wenn-Dann-Regeln dar. Diese bringen ihre Grenzen zum Ausdruck und geben genau an, für welche Umgebung sie gültig sind.

Formale Betrachtung Sei nach Ribeiro et al. [RSG18] f das Black-Box-Modell und x die zu erklärende Instanz. Weiter kommt es zu einer Pertubation um x herum, dessen Verteilung als D definiert ist. Um einen Anchor zu definieren sei nun A eine Regel, die aus einer Menge von Aussagen besteht. $A(x)$ ist dann 1, wenn alle Merkmalsaussagen wahr sind, für eine Instanz x. In einem Textklassifikationbeispiel, was z.B. den Satz „The Movie is not bad“ als positive Aussage klassifiziert ist, X der Satz und $f(x)$ positiv. Weiter wäre hier die Regel $A = \{not, bad\}$. In Worten könnte die Regel wie folgt formuliert werden: *Wenn die Wörter not und bad vorkommen, ist die Klasse positiv.* Sei nun $D(\cdot|A)$ die Verteilung von Instanzen, auf die A auch zutrifft (z. B. ähnliche Texte, in denen „not“ und „bad“ vorkommt) [RSG18]. $\mathscr{T}$ auf der rechten Seite der Formel gibt eine Präzisionsschwelle an. So sollen nur solche Wenn-Dann-Regeln als gültig

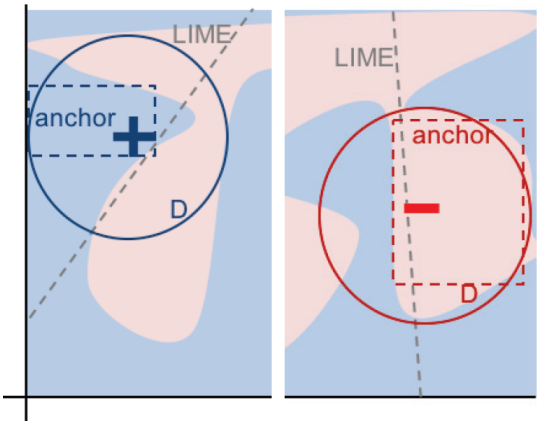


Abbildung 3.7: Funktionsweise der Anchors, entnommen aus [RSG18]

erachtet werden, wenn sie eine lokale Treue von mindestens $\mathscr{T}$ erreichen. Formal sei also ein Anchor definiert als [RSG18]:

$$\mathbb{E}_{D_{(z|A)}}[\mathbb{1}_{f(x)=f(z)}] \geq \mathscr{T}, A(x) = 1 \tag{3.3}$$

Molnar [Mol22] gibt diese Formel in eigenen Worten wieder: Wenn eine Instanz x erklärt werden soll, kann dies mit einem Anchor geschehen, welcher für x gilt. Hierbei sollte, die selbe Klasse für die Nachbarn von x mit einer Wahrscheinlichkeit von $\mathscr{T}$ vorausgesagt werden, damit dieser Anchor anwendbar bleibt. Die Genauigkeit einer Regel ergibt sich durch das Aufrufen des Black-Box-Modells für die Nachbarn, hier notiert mit $[\mathbb{1}_{f(x)=f(z)}]$.

Anwendung Ribeiro et al. [RSG18] zeigen in ihrer Veröffentlichung die Anwendung von Anchors für Klassifikation, strukturierte Vorhersagen und Textgenerierung auf Grundlage verschiedener Datentypen (tabellarisch, textuell, visuell). Auch für das Erstellen von Anchors gibt es eine Bibliothek[8], welche Entwickler unterstützt. Für das Finden von Anchors existieren mehrere Ansätze. Ein einfacher „Bottom-Up-Ansatz“ beruht darauf, mit einer leeren Regel zu starten und in Iterationen dieser Regel einen neuen Merkmalswert hinzuzufügen. Getestet würde diese Regel dann mit einer Prüfung der Genauigkeit $\mathbb{1}_{f(x)=f(z)}$ für jeden Nachbarn von x [RSG18]. Dies wäre jedoch in bei der Arbeit mit großen Datensätzen oder mit sehr vielen Merkmalsvariablen zu rechenaufwendig [Mol22]. Daher schlagen die Autoren vor lediglich Stichproben zu ziehen und auf statistisches Vertrauen in die Genauigkeit zu setzen. Technisch geschieht die Suche, welche auch in der Anchors-Bibliothek implementiert ist mit einem Reinforcement-Learning-Ansatz (Multi-Armed-Bandit), welcher für die Modellaufrufe zuständig ist, die für das Evaluieren von Regeln sorgen. Um die besten Kandidaten für eine mögliche Wenn-Dann-Regel A in die nächste Runde zu übertragen wird wird eine Balkensuche (modified Beam Search) angewendet [RSG18]. Wie viele Kandidaten durch diese Suche begutachtet werden, bestimmt auch die geforderte Rechenleistung und die zu erreichende Genauigkeit [Mol22]

Vor- und Nachteile Anchors sind einfach zu verstehen und werden dann angewendet, wenn alle Bedingungen der Regel erfüllt sind und weisen hierdurch eine hohe Genauigkeit auf [RSG18]. Daneben können auch genaue Erklärungen entstehen, selbst wenn die lokale Nachbarschaft komplex ist. Anchors können sehr generell, aber gleichzeitig auch sehr spezifisch werden, was für Nutzer verwirrend sein könnte. Weiter müssen viele Modellaufrufe stattfinden, was zur Lasten der Performance geht. Weiter existiert bei der Bildklassifikation ein konzeptuelles Problem, da nicht klar ist, wie Abdeckung hier genau zu definieren ist [Mol22].

8. https://github.com/marcotcr/anchor

3.2.9.4 Kontrafaktische Erklärungen

Kontrafaktische Erklärungen (engl. Counterfactual Explanations) beschreiben kausale Zusammenhänge in der Form „Wäre Merkmal A ein anderes gewesen, dann wäre auch die Vorhersage eine andere gewesen“ [Mol22]. Sie bringen also Gegenargumente (engl. Counter Facts) für eine Vorhersage.

3.2.9.5 Konzept

Bisher beschriebene XAI-Methoden stellen heraus, inwiefern sich Merkmale auf eine Vorhersage auswirken. Kontrafaktische Erklärungen hingegen, beantworten die Frage, welche Änderungen an einer Eingabe vorgenommen werden müssen, um eine andere Vorhersage zu erhalten [NP22]. Theoretisch wäre es zur Identifikation von kontrafaktischen Erklärungen möglich, zufällig Änderungen an der Eingabe vorzunehmen und zu beobachten, wie sich dies auf die Vorhersage auswirken würde. Zum einen ist dieses Vorgehen sehr rechenaufwendig, zum anderen hätte dies nach Molnar [Mol22] keinen Mehrwert für Nutzer. Kontrafaktische Erklärungen lassen sich eher als eine Erklärung definieren, welche die kleinste Veränderung an der Eingabe wiedergeben, sodass sich die Vorhersage ändert. Es kann eher wünschenswert sein, dass eine kontrafaktischen Erklärung eine solche Vorhersagenerklärung ist, welche die kleinste Veränderung in den Merkmalswerten wiedergibt, was zu einer Änderungen in der Vorhersage führt. Eine Herausforderung, die diese Erklärungen meistern müssen, basiert auf dem sogenannten Rashomon-Effekt. Rashomon ist ein Film, in dem Akteure eine Geschichte aus unterschiedlichen Perspektiven erzählen. Jede einzelne Geschichte ist in sich schlüssig, jedoch widersprechen sie sich untereinander. Dieses Phänomen kann auch bei kontrafaktischen Erklärungen auftreten. So könnte eine Erlkärung verlangen den Wert x_1 zu ändern, während eine andere vorschlägt Merkmal x_2 zu modifizieren, es können also mehrere gültige kontrafaktische Erklärungen existieren. Es Um Anforderungen an die kontrafaktischen Erklärungen zu definieren ist es wichtig, dass Nutzer definieren, was genau eine relevante Veränderung in der Vorhersage einer Instanz ist (z.B. Genauigkeit oder andere Klasse). Daneben ist es in der Realität sinnvoll, solche Erklärungen zu geneieren, die ungefähr dem Original entsprechen. Bei einen Algorithmus, welcher bspw. die Preise für eine Wohnung vorhersagt, sollte die Quadratmeter-Zahl der Wohnung nicht negativ werden [Mol22].

Formale Grundlagen Für eine zu erklärende Instanz x muss ein Kontrafakt x' gefunden werden [KL21]. Für dieses x' sollte das zugrundeliegende Modell f die eine andere Vorhersage als für x ausgeben, hier mit y' notiert. Wie vorangegangen beschrieben, besteht weiter das Ziel, dass sich das Kontrafakt nicht besonders von der zu betrachtenden Instanz unterscheidet (hier durch d angegeben). Dies lässt sich formal in folgendem Optimierungsproblem L beschreiben[KL21]:

$$\min_{x'} L(x'|x) = (f(x') - y')^2 + \lambda d(x,x') \tag{3.4}$$

λ bestimmt das Verhältnis von $(f(x') - y')^2$ und der Distanzfunktion d. Für einen großes λ wird ein Kontrafakt erstellt, welches sich in den Merkmalswerten von x wenig unterscheidet. Für einen kleinen Wert werden solche Kontrafakte erstellt, die sich sehr nahe an der gewünschten Ausgabe y' befinden. Das Kontrafakt x' und dessen Merkmalswerte können nun genutzt werden, um mit Bezug auf die ursprüngliche Instanz eine Wenn-Dann-Regel aufzustellen, die beschreibt, welche Werte für eine andere Vorhersage geändert werden müsste.

Anwendung Für kontrafaktische Ekärungen bestehen sowohl modell-agnostische ([Dhu+19; Gui+18a]) als auch modellspezifische Ansätze (Wachter et al. [WMR17] und Mothilal et al. [MST20]), welche sich z.B. die Gewichte von neuronalen Netzen zu Nutze machen. Die DICE-Bibliothek von Microsoft bietet Möglichkeiten zur Generierung von kontrafaktischen Erklärungen an [9]. Beim Benutzen dieser Biblothek kann angegeben werden, wie viele kontrafaktische Erklärungen produziert werden soll.

Vor- und Nachteile Ein Vorteil, welchen kontrafaktische Erklärungen gegenüber anderen Methoden wie z.B. LIME besitzen, ist ihre Genauigkeit. Sie stellen relativ klar heraus, welche Merkmale geändert werden müssen. LIME z.B. kann keine Aussagen über den genauen Wert geben, da diese nur lokale Genauigkeit besitzen, wie in Abbilung 3.4 gezeigt. Daneben sind sie für Menschen gut verständlich [Mol22]. Ein Nachteil dieser XAI-Methode besteht in dem grundsätzlichen Rashomon-Effekt, der es erschwert, die richtige Erklärung auszuwählen [KL21]. Im Zweifel sollten den Nutzer daher eine größere Zahl vieler verschiedener Erklärungen präsentiert werden [Mol22]. Daneben haben einige Methoden, z.B. die von [WMR17] zum Finden der Kontrafakte, dass Problem, nicht nur solche Kontrafakte zu finden, welche wenig Merkmale abändern. Des Weiteren werden auch unrealistische Merkmale miteinbezogen [Mol22]

3.2.10 Modellagnostische Methoden - global

3.2.10.1 Partial Dependence Plot

Ein Partial Dependence Plot kann zur Visualisierung eingesetzt werden, nachdem ein Modell trainiert wurde [Gia21]. Sie verfolgt das Ziel aufzudecken, inwiefern Merkmale die Black-Box-Modell-Ausgaben beeinflussen. Diese Art der Darstellung misst, wie sich ein Merkmal auswirkt, wenn man den Wert diesen immer etwas abändert, aber die Werte der anderen Merkmale nicht modifiziert. [KL21]. Vereinfacht gesagt würde man bei zwei Variablen, die jeweils z.B. einen möglichen Wert von 1-5 annehmen können, für das erste Merkmal das Modell mit dem Wert 0 aufrufen, während man das zweite Merkmal nicht verändert. Im nächsten Schritt würde man den Wert des ersten Merkmals auf 1 setzen usw. Ist dies für alle möglichen Werte des ersten Merkmals geschehen, wird der Mittelwert der Vorhersagen berechnet. Dies wird daraufhin für das zweite Merkmal wiederholt. Formal ist die PD definiert mit d als Anzahl der Merkmalen ([KL21] nach [Fri01]):

$$PD(x_1) = \mathbb{E}_{x_2}[f(x_1, x_2)] = \int p(x_2) f(x_1, x_2) d_{x_2} \tag{3.5}$$

Abbildung 3.8 zeigt ein Beispiel für einen PDP.

Ein Vorteil der PDP-Methode ist es weniger zu zeigen, welches Merkmal den größten Einfluss auf eine Vorhersage hat (wie z.B. die Methode der Permutation Importance), sondern bestimmte Schwellenwerte aufzudecken. So können PDP dabei helfen, zu definieren, wie ein Wert eines Merkmals besetzt sein sollte, um ein bestimmtes Ergebnis zu bekommen [Gia21]. Ein Nachteil dieser Methode beruht auf der Annahme, dass die Merkmale unabhängig voneinander sind. So wird, wie oben erklärt, der Wert des ersten Features nach Belieben geändert, während der Wert des zweiten Features gleich bleibt. Jedoch führt dies zu Kombinationen, welche in der Realität nicht vorkommen können [Mol22].

dazu muss ich noch ein Kapitel schreiben

9. https://www.microsoft.com/en-us/research/project/dice/

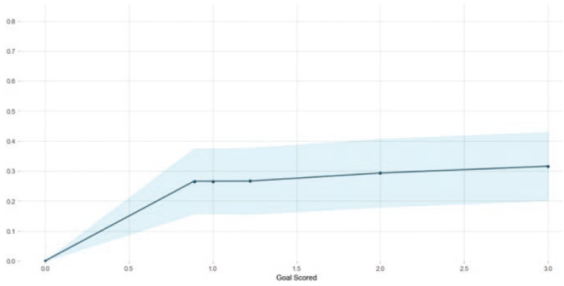


Abbildung 3.8: Beispiel für einen PDP, entnommen aus: [Gia21]

Eine weitere ähnliche Methode ist der *Accumulated Local Effects (ALE) Plot* nach Apley und Zhu [AZ20], welches das Problem der Abhängigkeit löst.

3.2.10.2 Individual Conditional Expectation (ICE)

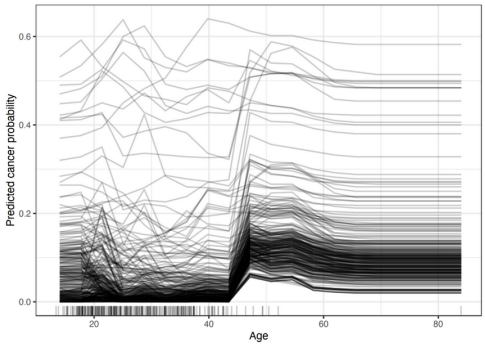


Abbildung 3.9: Beispiel für ICE, entnommen aus: [Kim21]

Die Funktionsweise der ICE-Methode ist die selbe der PDP (vgl. Unterkapitel 3.2.10.1), wir hier jedoch lokal anstatt global angewendet, indem die Werte für jede Instanz berechnet werden [Mol22]. Somit lassen sich der Einfluss von Änderungen einzelner Features bestimmen [HZW21]. Nach Molnar [Mol22] ist ein PDP ein Mittelwert der ICE. ICE kann jedoch eine gute Alternative bieten, tiefere Einblicke in die Feature-Interaktionen zu erhalten [Mol22]. In Abbildung 3.9 ist ein Beispiel für eine ICE zu sehen:

3.2.10.3 Global Surrogate

Um jedes beliebige Black-Box-Modell zu erklären, kann ein interpretierbares Modell eingesetzt werden, welches sich dem ursprünglichen Modell annähert. Diese Methode heißt Global Surroagte (globaler Stellvertreter) [Mol22]. In Kapitel 3.2.9.1 ist dieses Konzept lokal beschrieben. Ziel dieser Methode ist eine ungefähre Abschätzung des ursprünglichen Modells, während Interpretierbarkeit gewährleistet wird. Entscheidend bei dieser Methode ist die Wahl des Modells, denn dieses muss interpretierbar sein. Eine Auswahl von transparenten Modellen ist in Kapitel 3.2.4.1 zu finden, so sind bspw. lineare Modelle oder Entscheidungsbäume geeignet [Mol22].

Die Implementation eines solchen globalen Stellvertretermodells ist relativ simpel, so wird kein Zugriff und auch kein Wissen über die Modellstruktur benötigt. Entwickler, welche einen

globalen Stellvertreter entwickeln wollen, müssen sich zunächst für einen Datensatz entscheiden und für diesen die Vorhersagen abrufen. Im nächsten Schritt wird sich für ein interpretierbares Modell entschieden, welches dann auf die ursprünglichen Daten und Vorhersagen trainiert wird. Daraufhin findet eine Evaluation statt [Mol22].

Eine Möglichkeit, zu überprüfen, wie exakt der Stellvertreter das Black-Box-Modell annähert, ist das R^2-Maß, welches wie folgt definiert ist:

$$1 - \frac{\sum_{i=1}^{n}(\hat{y}_*^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{n}(\hat{y}^{(i)} - \bar{\hat{y}})^2} \tag{3.6}$$

Die Vorhersage des interpretierbaren Modells ist für die i-te Instanz $\hat{y}_*^{(i)}$, während die Vorhersage des Urpsrungmodells mit $\hat{y}^{(i)}$ notiert ist. $\bar{\hat{y}}$ gibt den Mittelwert der Vorhersagen des Black-Box-Modells an. Wenn sich das Ergebnis in der Nähe von 1 befindet, ist dies so zu verstehen, dass der Stellvertreter sich dem Modell gut annähert. Nähert sich der Stellvertreter sehr gut an, kann sogar in Betracht gezogen werden, dieses anstatt des ursprünglichen Modells zu verwenden. Liegt der Wert nahe bei 0, bedeutet dies analog, dass sich das Stellvertretermodell nicht eignet [Mol22].

Ein Vorteil dieser Methode ist dessen einfache Anwendung. Daneben kann flexibel jedes mögliche Modell als Stellvertreter gewählt werden. So kann sowohl auf die Kenntnisse der Entwickler als auch auf die der Nutzer eingegangen werden. Wenn alle Stakeholder sich z.B. gut mit Entscheidungsbäumen auskennen, kann dieses Modell als Stellvertreter gewählt werden. Daneben existieren natürlich auch Nachteile dieser Methode. Jedes interpretierbare Modell bringt selbst seine eigenen Schwächen mit und es werden Annahmen über das Black-Box-Modell getroffen, welche eventuell nicht der Wahrheit entsprechen. Dazu kann es eine schwierige Entscheidung sein, welcher R^2-Wert als ausreichend betrachtet wird.

3.2.10.4 Protoypen und Kritiken

XAI-Modelle können auch mit konkreten Dateninstanzen erklärt werden. Als Protoypen werden solche Instanzen bezeichnet, die einen Datensatz repräsentieren, während Kritiker solche Datenpunkte sind, welche schlecht die gesamten Datensätzen wiedergeben. Mit diesen Konzepten lassen sich sowohl Datensätze als auch ML-Modelle erklären, was in Abbildung 3.10 dargestellt ist für Instanzen, welche die Merkmale $x1$ und $x2$ in unterschiedlicher Ausprägung besitzen [Mol22]. Zum Finden dieser Datenpunkte existieren verschiedene Methode. Die einfachste Methode stammt von Rdusseeun und Kaufman [RK87], lässt sich jedoch nur für Protoypen anwenden. Diese Methode heißt *k-medoids* und funktioniert ähnlich zu der in Kapitel 2.1.4.2 beschriebenen Methode k-means Clustering. Eine weitere Methode, die sowohl das Identifzieren von Protoypen als auch Kritiker ermöglicht, heißt *MMD-critic* und wurde von Kim et al. [KKK16] vorgestellt.

Hier kommt noch eine Beschreibung der MMD-Critic-Methode

3.2.11 Modellspezifisch: Neuronale Netze

Folgendes Unterkapitel beschäftigt sich mit solchen Methoden, welche sich auf neuronale Netze anwenden lassen.

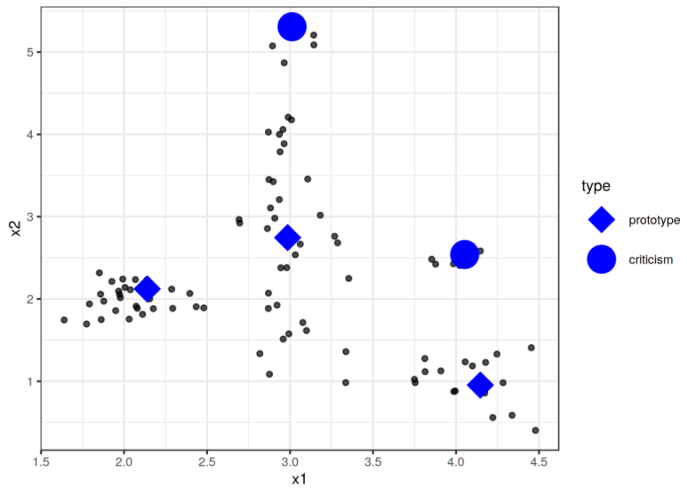


Abbildung 3.10: Protoypen und Kritker, entnommen aus: [Mol22]

3.2.11.1 Layer-wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) wurde erstmals von [Bac+15] vorgestellt und zeigt bei neuronalen Netzen die Relevanz eines Eingabewertes für die Ausgabe. Bach et al. [Bac+15] illustrieren dies in ihrem Paper sehr gut anhand eines Klassifikationsalgorithmus (CNN), welcher Bildern eine Klasse zuweist. Ein bestimmter Pixel ist hier die Eingabe und ist auf eine gewisse Weise für die Vorhersage verantwortlich. Um herauszufinden welche Pixel eines Bildes sich besonders auf die Vorhersage auswirken, wir im Rahmen der LRP-Methode ein Relevanzwert für jedes Neuron in jedem Layer berechnet, indem rückwärts durch das neuronale Netz gegangen wird. Somit wird sich in verkehrter Reihenfolge dem Input angenähert und die relevanten Pixel werden sichtbar. Dies nennen die Autoren „pixel-wise decomposition“ [Bac+15]. Bach et al. [Bac+15] stellen folgende Möglichkeit zur Berechnung der Relevanzwerte vor:

$$R_i^{(l)} = \sum_i \frac{z_{ij}}{\sum_{i'} z_{i'j}} R_j^{l+1} \text{wobei gilt: } z_{ij} = x_i^{(l)} w_{ij}^{(l,l+1)} \tag{3.7}$$

$l+1$ sei hier der letzte Layer, mit dem gestartet wird und somit wird von hinten für jedes Neuron j im Layer $l+1$ die Aktivierung berechnet. Diese Aktivierung ergibt sich dadurch, inwiefern sich ein Neuron i im Vergleich zu den anderen Neuronen in dem Layer l auswirkt. So werden die Gewichte mit dem Input x mutlipliziert, welche sich jeweils in den nächsten Layer übertragen. Die Aktivierung gibt Aufschluss darüber, wie relevant ein Neuron jeweils für den nachfolgenden Layer (und somit auf den endgültigen Output) war. Ist dies für alle Layer geschehen, wird der ursprüngliche Input, also die Pixel des Bildes erreicht. Diese lassen sich dann visualisieren.

3.2.11.2 Saliency Maps

Saliency Maps basieren auf einem ähnlichen Prinzip wie die LRP-Methode und lassen sich auch auf Bildklassifikation anwenden. Hier wird auch gezeigt, welche Teile eines Bildes releavant für die Vorhersage waren und es wird auf die innere Struktur des Netzwerkes zurückgegriffen [Gia21]. Saliency Maps wurden zuerst vorgestellt von [SVZ13] und basieren auf der Berechnung, der Gradienten, welche bei der Optimierung der Verlustfunktion von neuronalen Netzen zum Einsatz kommen. Konkret helfen Gradienten dabei die Verlustfunktionsparameter zu optimieren, in dem vorgegeben wird, wie weit sich auf einer Funktion bewegt werden muss [Mol22].

Nach Simonyan et al. [SVZ13] gibt es hier ein Bild I_0 und eine Klasse c. Daneben existiert eine Klass-Score-Funktion $S_c(I)$. Ziel ist es, die Pixel anhand ihres Einflusses auf den Score $S_c(I)$ zu ordnen. Dieser Score ist zumeist in CNNs nicht linerar berechenbar, lässt sich jedoch annähern (mithilfe einer Taylor Series Expansion):

$$S_c(I) \approx w^T I + b \tag{3.8}$$

Analog zum Grundlagenkapitel sei hier b der Bias und W sind die Gewichte. Die Saliency Map lässt sich für ein Merkmal j wie folgt bestimmen [KL21]:

$$M_j = |w_j| \tag{3.9}$$

Ein Beispiel für eine Saliency Map ist in Abbildung 3.11 dargestellt. Ein Vorteil dieser XAI-Methode besteht in ihrer leichten Berechnung und dass keine weiteren Annotationen zur Erklärung eingeführt werden müssen [KL21]. Gianfagna [Gia21] schreiben jedoch, dass diese Art der Darstellung für Menschen nicht besonders hilfreich ist, da wenig erkannt werden kann. Daneben besteht ein Nachteil, welcher sich durch die Score-Funktion ergibt. Ist diese nicht ableitbar, lässt sich kein geeigneter Score ermitteln, was z.B. für die ReLu-Funktion gilt [SGK17]. DeconvNet von [ZF14] ist ein ähnlicher Ansatz, welcher das Problem mit der ReLU-Function löst. Grundsätzlich können jedoch auch LIME oder SHAP für das Erstellen von Saliency-Maps genutzt werden [Mol22].

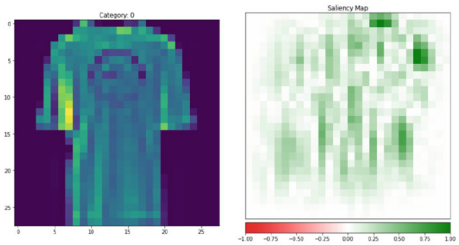


Abbildung 3.11: Beispiel für Saliency Map, entnommen aus: [KL21]

3.2.11.3 TCAV

TCAV steht für „Testing with Concept Activation Vectors“ und wurde von Kim et al. [Kim+18] vorgestellt. Diese Methode konzentriert sich darauf, mithilfe von Konzepten die Funktionsweise von neuronalen Netzen Menschen näher zu bringen. Nach Kim et al. [Kim+18] bringen viele andere XAI-Methoden den Nachteil mit sich, dass die Fokussierung auf Merkmale schlecht für Nutzer zu interpretieren ist. Merkmale liegen häufig in niedriger Abstraktionsebene vor, aber Menschen denken nicht auf niedriger, sondern auf hoher Abstraktionsebene - in Konzepten.

Die TCVA-Methode kann genutzt werden, um auf hoher Abstraktionsebene zu testen, ob ein bestimmtes Konzept in die Vorhersage des Algorithmus miteinfloss. Dies ist sogar dann möglich, wenn dieses Konzept nicht Teil des Trainingsdatensatzes war. Illustriert wird die Fuktionweise in Abbildung 3.12 [Kim+18].

Als Beispiel ist hier ein neuronales Netz, welches Bilder klassifziert und ausgibt, ob sich auf dem Bild ein Zebra befand. Was nun von Interesse sein kann, ist die Frage, ob das Konzept *Streifen* mit in die Bestimmung der Vorhersage einfloss. Konzepte lassen sich als Vektoren

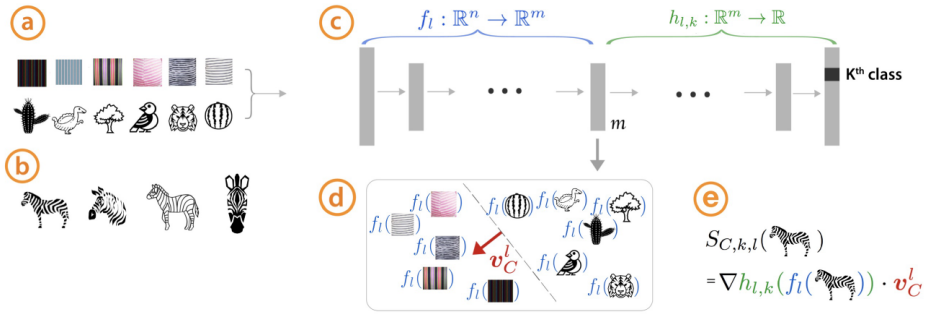


Abbildung 3.12: Funktionsweise von TCAV, entnommen aus: [Kim+18]

(CVA: Contept Activation Vector) definieren. Um einen solchen Vektor zu bilden, sind zunächst Beispiele von Nöten, z.B. Bilder von Streifen. Daneben werden noch andere, zufällige Bilder benötigt. Eine weitere Voraussetzung für das Durchführen der TCAV-Methode ist der Zugang zum Netzwerk und seinen Aktivierungen. Zu Beginn werden alle Beispiele (solche mit Streifen und solche ohne) durch das Netzwerk geleitet und die Aktivierungen gesammelt. Daraufhin wird ein linearer Klassifikationsalgorithmus trainiert, welcher die beiden Klassen voneinander abgrenzt. Der CVA steht nun orthogonal auf der durch den linearen Klassifzierungsalgorithmus erstellten Entscheidungsgrenze und zeigt auf die Bilder, welche Streifen enthielten. Nun kann mittels des Vektors das Konzept getestet werden.

Ich würde noch die zwei unvollständigen Kapitel fertig schreiben und dann noch die Methode DeepLift, Neural Shrubs und etwas zu Feature Interaktionen schreiben

Hier fehlt noch, wie das Testen an sich funktioniert

3.2.12 Weitere Methoden

Wenn das in Ordnung ist, würde ich hier noch einige Methoden auf hoher Abstraktionsebene beschreiben. Es gibt halt sehr viele, und ich glaube nicht dass ich jede (und schon gar nicht zu genau) beschreibene sollt. In der folgenden Auflistung siehst du ein paar von den Methoden, welche mir theoretisch fehlen würde, auch wenn dies nicht alle sind, die ich insgesamt gefunden habe.

- CAM with global average pooling (Sonderformen: Grad-CAM, Guided Grad-CAM mit Feature Occlusion, Repsond CAM, Multi Layer CAM)
- DeepLIFT
- Prediction Difference Analysis
- Slot Activation Vectors
- PRM (Peak Response Mapping)
- direct output labels
- Image Corruption and Testing Region of interest
- Attention map with autofocus convolutional layer
- DecnovNet

- SVD
- CCA
- SVCCA
- DWT
- Explanation Vector
- LSTMVis

Die Methoden könnten dann so oder so ähnlich beschrieben werden

PCA

Principal Component Analysis wurde von [Pea01] eingeführt. und kann zur Feature Extraktion genutzt werden [TG20]. Hier wird weniger ein Modell erklärt, als dass die Dimensionen der jeweiligen Merkmale reduziert werden und somit für den Nutzer einfacher verständlich werden.

3.2.13 Präsentation von Erklärungen

Neben dem Erstellen der Erklärung, existieren auch einige Hinweise in der wissenschaftlichen Literatur, wie diese an den Erklärungsempfänger zu übermitteln sind.

3.2.13.1 Integration von Domänenwissen und Kontext

Nach [ZC18] ist die Integration von Domänenwissen besonders wichtig, um Erklärungen verständlicher und praxisorientierter zu gestalten. [SD05] integrierten bspw. Domänenwissen in bayesche Netzwerke.

3.2.13.2 Form der Präsentation

[Wei+19b] überprüfen in ihrer Veröffentlichung, ob ein virtueller Agent die von dem XAI-Algorithmus erstellten Erklärungen an Endnutzer kommunizieren könnte. Ergebnis dieser Untersuchung, dass menschliche Züge eines virtuellen Agenten die Interaktion erleichtern, was gleichzeitig zu einem besseren Verständnis und auch zu mehr Vertrauen auf Endnutzerseite führt. Hier sei jedoch angemerkt, dass Vertrauen schwer zu messen ist [Wei+19b]. Auch [VW20] stellen heraus, dass für Menschen verbale sowie nonverbale Kommunikation zum Verständnis hilfreich ist. Um für Menschen verständlichere Erklärungen generieren zu können, haben [KHA20] eine Methode entwickelt, welche aus Charts (z.B. Balkendiagrammen) automatisch einen beschreibenden Text generieren kann.

3.2.13.3 Interaktivität

Bei der Präsentation können Visualisierungstools helfen. Interaktivität ist ein besonders wichtiges Element dieser Tools, da ein Vor- und Zurückgehen durch die Daten und Erklärungen hilfreich sein können, um die Daten und die Funktionsweise des Modells besser zu verstehen. Des Weiteren kann der Nutzer sich selbst aussuchen, was für sein Verständnis wichtig ist und wie er Informationen am besten aufnimmt [VW20]. Es existieren auch Untersuchungen im wissenschaftlichen Kontext, welche die Vorteile von Interaktivität untersuchten. So fanden [Vac+18] heraus, dass Menschen mit dem Output eines System zufriedener sind, wenn sie diesen mithilfe von einem interaktiven Schieberegler abfragten. Außerdem hilft Interaktivität und die Möglichkeiten des Nutzers den Output der XAI-Methoden mitzubestimmen, dabei explorative Experimente durchzuführen [WB19]. Dies erlauben beispielweise Drill-Down-Methoden hin zu konkreten Erklärungen und die Möglichkeit Erklärungen miteinander zu vergleichen oder die Wahlmöglichkeit bei den Darstellungen [WB19].

Tools können dabei helfen solche interaktiven Visualisierungsformen zu erstellen [ZC18]. Ein solches Tool ist z.B. der Model Tracker, welcher Analysen bzgl. der Performance und Debugging ermöglicht [Ame+15]. Ein weiteres Visualisierungstool wurde von [Che+16] entwickelt, welches zehn verschiedene Techniken (shaded confusion matrix, ManiMatrix, learning curve, learning curve of multiple models, McNemar Test matrix, EnsembleMatrix, Customized SmartStripes, Customized ModelTracker, confusion matrix with sub-categories, force-directed graph) zur Anwendung bereitstellt. Zur Visualisierung von Partial Dependence wurden Tools von [KPN16] bereitgestellt, welche sich vor allem an Data Scientists richten.

Eventuell erkläre ich diese noch oder lasse es ganz weg

Ein Beispiel für ein solches Tool wird an dieser Stelle kurz illustriert, um dessen Vorteile sichtbar zu machen. Der von Krause et al. [Kra+17] entwickelte Workflow richtet sich an Domänen-Experten und Data Scientists. In ihrer Veröffentlichung wird demonstriert, wie sich durch Visualisierung ein Klassifikationsalgorithmus erklären lässt. Hierbei wird sich darauf konzentriert einzelne Instanzen zu erklären. Elemente ihrer Visualisierung sind:

- Aggregierte Statistiken verdeutlichen z.B. den Anteil korrekt und fehlerhaft klassifizierter Instanzen (vgl. Abbildung 3.13)
- Erklärungen, welche die Relevanz einzelner Merkmale verdeutlichen
- Präsentation von Rohdaten, damit Nutzer diese als ergänzende Erklärungen heranziehen können

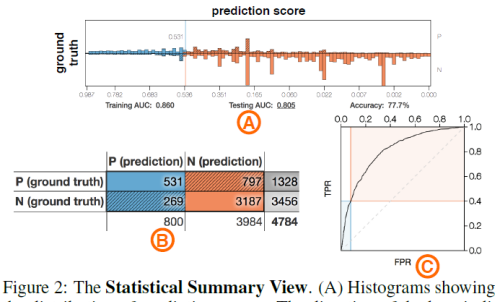


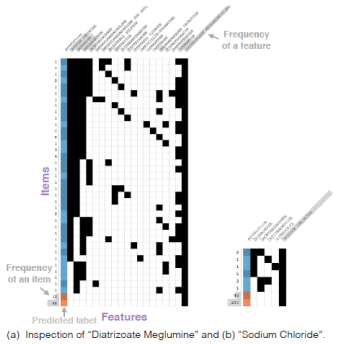
Figure 2: The **Statistical Summary View**. (A) Histograms showing the distribution of prediction scores. The direction of the bars indicates the ground truth and their position relative to the threshold line (at 0.531) indicates the predicted label. (B) The confusion matrix shows the number of correct and incorrect predictions. (C) The ROC curve shows the prediction quality.

Abbildung 3.13: Statistische Zusammenfassung, entnommen aus [Kra+17]

Das Prinzip des Workflows beruht auf Interaktivität, da dies die Bildung eines mentalen Modells auf Nutzerseite unterstützt und hiermit Algorithmusentscheidungen leichter nachzuvollziehen sind. Weiter abstrahieren sie von der Funktionsweise des ML-Algorithmus. Dies begründen sie darin, dass Nutzer, welche sich hiermit nicht auskennen, nur mit zu viel Wissen überfordert werden würden. Workflow: - vorher: Erklärungen und Visualisierung (interaktiv Erklärungen

Figure 3: In the **Explanation Explorer** each row represents a group of data items explained by a set of features (E). An indicator is shown for explanations longer than 3 features. Column (F) shows the distribution of true / false positive / negative data items within the group. Colors show the predicted label ("blue" for positive and "orange" for negative) and a hatching pattern indicates incorrect predictions. Column (G) shows the number of items captured by the explanation. The bars are relative to the size of the largest explanation. Column (H) shows the odds ratio of the group on a logarithmic scale. Whiskers show the confidence interval. The arrows on the right (I) navigate to the Item Level Inspector focusing on the given explanation. The controls of the Explanation Explorer are shown on the left. The first entry of the list of filtered data items (B) represents the full dataset and following entries show sizes after filter steps are applied. The "+" creates a new filter according to the current selection of explanations. Explanations can be selected satisfying a condition (C) or by searching for features in the search box (A). The sort order of explanations is defined by the list at the bottom (D).

Abbildung 3.14: explanation explorer, entommen aus: [Kra+17]



(a) Inspection of "Diatrizoate Meglumine" and (b) "Sodium Chloride".

Figure 4: The **Item Level Inspector** showing a matrix of data items as rows and features as columns for the explanations *Diatrizoate Meglumine* and *Sodium Chloride* in the initial data set of the case study (Section 6). Rows group identical instances together and show the count on the left side. Features are sorted by "relative feature importance" showing from left to right how labels can be separated.

Abbildung 3.15: item level explorer, entommen aus: [Kra+17]

für Daten anzeigen) - Outcome-Level: grundsätzliche Accuracy z.B. mit Confusion matrix - Feature-Level: Alle Erklärungen mit gleichen Merkmalen gebündelt anzeigen - Instance-Level: Erklärung für eine Instanz der Daten Interaktive Auswahl an Erklärungen, Nutzer hat kein Zugriff auf die interen Struktur Angewendete XAI-Methoden: Prospector [17], LIME, Martens und Provost [24 (Erklärungen für binären Input) Grenzen des Ansatzes: nur binary-Data und kein Multihttps://de.overleaf.com/project/625d7985bac938c0561d8eda-Class So sieht der Bums aus: Statistische Zusammenfassung:

Explanation Explorer Itemlevel Explorer

Andere ähnliche Methoden wie die von [Kra+17]: -ModelTracker [3]: Schnittstelle für Fehlererkennung und Fehlersuche bei binären Klassifizierern, die itemweise Verteilungen von Vorhersageergebnissen anzeigt. - Bilal et al.: confusion wheel visualization [1], um die Wahrscheinlichkeiten der Zugehörigkeit von Elementen zu verschiedenen Klassen für Mehrklassenklassifikatoren anzuzeigen. - Squares [26] bietet eine einzige, einheitliche Visualisierung von Leistungsmetriken und einen einfachen Zugang zu den Daten für das Debugging von Mehrklassenklassifikatoren. - Cortez und Embrechts [9]: Ansatz der Sensitivitätsanalyse, der es den Benutzern ermöglicht, die Auswirkungen der Variation von Eingabewerten auf die Modellergebnisse zu verstehen

3.2.14 XAI-Plattformen

Ich überlege noch wie/ob ich einen Unterschied zwischen Visualisierungstools, Plattformen, Toolkits mache

Zur Unterstützung bei der Anwendung von XAI-Methoden können Plattformen genutzt werden.

Für das Entwickeln von Plattformen haben [RJ+20] einige Charakteristiken identifiziert, welche optimalerweise vorhanden sein müssen. Zunächst wird auf die *Mensch-Maschinen-Interaktion* eingegangen. In einer Benutzeroberfläche sollten kontextbezogene Informationen neben Gründen für eine Vorhersage vorhanden sein. Daneben sollte auch eine Darstellung über den Entwicklungsprozess vorhanden sein. *Entscheidungsfindung* sollte durch eine Plattform auch unterstützt werden. Bei Entscheidungsunterstützungen sollte auch in der Plattform beachtet werden, dass diese den Plänen der Organisation folgen sollten. Daneben sollte der Prozess Entscheidungsfindung auch das Fachwissen der Nutzer unterstützen. Das Tool sollte außerdem Risiken aufzeigen, welcher mit einer Entscheidung, die auf Grundlage des Outputs getroffen werden einhergehen. Die *Architektur* eines Systems ist im Idealfall in einer Plattform zu zeigen. Inhalte, die in Bezug auf die Architektur erfasst werden sollten sind z.B. die Datenerfassung, die Datenanalyse, oder das Bewerten von Prognosen. Auch Informationen anderer Systeme sollten integriert werden. Kommt es zu Ausfällen bei Prognosen, hat die Plattform auf diese aufmerksam zu machen. [RJ+20] haben selbst eine experimentelle Plattform vorgestellt, welche als Automated Reliability Decision Aid System (ARDAS) bezeichnet wird.

Zum Anwenden verschiedener XAI-Methoden existieren Software-Toolkits. Ein Beispiel ist das AI Explainability 360 Toolkit von IBM [10] [Ary+21]. Dieses Toolkit unterstützt die Entwicklung von ML-Algorithmen in der Programmiersprache Python mithilfe von acht verschiedenen Erklärungsmethoden:

- *Boolean Decision Rules via Column Generation* werden mithilfe des BRCGExplainer erklärt, welcher binäre Klassifikation unterstützt und direkt interpretierbar ist [DGW18]
- Die direkt interpretierbaren *Generalized Linear Rule Models* werden mithilfe des GLRMExplainer erklärt [Wei+19a]
- Der *ProtodashExplainer* ist für das Erläutern von Datensätzen zuständig. Hier werden repräsentative Teile der Daten ausgewählt oder Testinstanzen erklärt [Gur+19].
- Der *ProfWeightExplainer* liefert globale Erklärungen durch das Darstellen von Gewichten neuronaler Netze [Dhu+18b].
- Personenspezifische Erklärungen werden mithilfe des *TEDExplainers* umgesetzt [Hin+19].
- Die Methode von [Dhu+18a] (*CEMExplainer*) wird genutzt, um lokale Erklärungen zu generieren. Dies sind konstrative Erklärungen. Z.B. wird die Frage beantwortet, was minimal ausreichend ist, um die ursprüngliche Klassenvorhersage beizubehalten.
- Für Erklärungen von Bildern wird der *CEMMAFImageExplainer* nach [Lus+19] eingesetzt.
- Abhängigkeiten von Merkmalen werden mithilfe von dem *IPVAEExplainer* nach [KSB17] verdeutlicht.

10. https://aix360.mybluemix.net/

Daneben bietet das Explainability 360 Toolkit auch Bewertungsmetriken an und eine erweiterbare Softwarearchitektur. [Ary+21] empfehlen für das Arbeiten mit ihrem Toolkit des Weiteren einen drei-schrittigen Ablauf, welcher zunächst das Filtern auf verschiedene Instanzen vorsieht. Daraufhin sollten Merkmale einzelner Instanzen präsentiert werden und diese sollten mit anderen Instanzen verglichen werden. Abschließend sollten Ekrlärungen generiert werden, z.B. kontrafaktische Erklärungen. Auch [Ary+21] weisen darauf hin, dass es schwierig ist zu entscheiden, welche Erklärung genau zu wählen ist. Zielgruppe dieses Tookits sind Entwickler und Data Scientists. Zur Unterstützung werden außerdem Tutorials angeboten, welche auch mit fachlichem Kontext (Medizin) vorhanden sind.

Toolkit	Data Explanations	Directly Interpretable	Local Post-Hoc	Global Post-Hoc	Persona-Specific Explanations	Metrics
AIX360 [9]	✓	✓	✓	✓	✓	✓
Alibi [17]			✓			
Skater [4]		✓	✓	✓		
H2O [3]		✓	✓	✓		
InterpretML [22]		✓	✓	✓		
EthicalML-XAI [2]				✓		
DALEX [10]			✓	✓		
tf-explain [5]			✓	✓		
iNNvestigate [7]			✓			
modelStudio	✓	✓	✓	✓		
ELI5 [1]		✓	✓	✓		
Iml [21]		✓	✓	✓		
Captum [18]			✓			
WIT [24]	✓		✓	✓		

Abbildung 3.16: Toolkits im Vergleich, entommen aus: [Ary+21]

Neben dem Explainability 360 Toolkit weisen [Ary+21] auch auf andere Toolkits hin und vergleichen diese in Abbildung 3.16 in Bezug auf ihre Funktionalitäten.

3.2.15 Vorgehen bei Entwicklung

3.2.15.1 Vorgehensmodell für transparentes ML

Grundsätzlich ist es bei einem praktischen Problem der realen Welt empfohlen, dass bei der Entwicklung von Software mit Endnutzern und Domänenexperten zusammengearbeitet wird und der gemeinsame Prozess so gestaltet wird, dass Endnutzer und Domänenexperten Feedback geben können [ZC18]. Hierfür haben [ZC18] ein Modell entwickelt, welches *twodimensional (2D) transparency space* heißt und in Abbildung 3.17 zu sehen ist. Dieses Modell beschreibt den Workflow des transparenten ML (vgl. Kapitel 3.2.1.1) und integriert sowohl ML-Experten als auch Domänennutzer in den Entwicklungsprozess.

Aus Nutzerperspektive ergeben sich nach diesem Vorgehen drei Hauptaspekte. Zunächst wird das ML-Forschungsprogramm eingerichtet, was der Problemdefinition dient [ZC18]. Gleichzeitig sollten Nutzer sogenannte *transparente Fragen* stellen. Hier werden Informationsbedarfe für das ML-System aufgedeckt und beantwortet. Während der Vorbereitung ist beispielsweise wichtig, wie das Problem in ML überführt wird. In der Entwicklungsphase hingegen kann interessant sein, wie auf Daten zugegriffen wird. Wie die Leistung oder die Interpretation der Ergebnisse aussieht, sind transparente Fragen, welche während des Deployments stattfinden. Der dritte Baustein aus Nutzerperspektive ist der *transparente Nutzen*. Während der Vorbereitung sollte verständlich gemacht werden, in welcher Form sich Inputs auf Outputs auswirken. Gleichzeitig sollen Unklarheiten über die Geschäftsanforderungen an das ML-System aufgelöst werden.

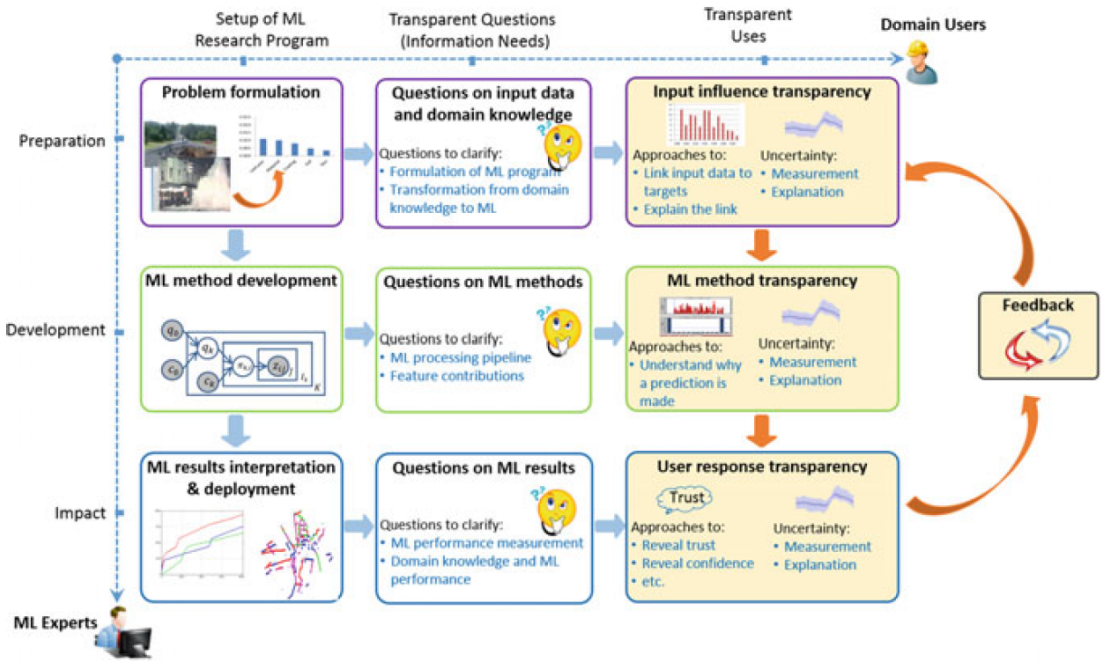


Abbildung 3.17: 2D transparency space, entnommen aus [ZC18]

Weiter sollte die ML-Methode selbst transparent gemacht werden, indem dessen Funktionsweise dargelegt wird. Eine Frage, welche hier beantwortet werden sollte ist z.B. „Warum wird diese mithilfe von Wissen über die Domäne Entscheidung getroffen?“. Während des Deployments sollten Reaktionen der Benutzer untersucht werden, um deren Vertrauen und Zuversicht zum ML-System zu stärken [ZC18]. Ein wichtiger Bestandteil sind die Feedback-Schleifen. Sollte zum Ende des Projektes herausgefunden werden, dass Nutzer kein Vertrauen in das System haben, sollte die Problemdefinition und die Art der Überführung des Domänenproblems in ein ML-System erneut überdacht werden [ZC18].

Auch [VW20] weist darauf hin, wie wichtig es ist verschiedene Stakeholder einzubeziehen, um dessen Bedürfnisse zu kennen. Dies sei besonders wichtig, um den Einsatz des ML-Systems im realen Kontext einschätzen zu können. Mögliche Stakeholder sind: Entwickler, Designer, Data Scientists, Manager, Regulierungsbehörden, Nutzer, Personen, welche in sonstiger Weise vom System betroffen sind.

Stakeholder früh einzubinden kann weiter das Problem des kognitiven Bias’ bei Nutzern lösen, welcher bewirkt dass sie Erklärungen falsch oder anders verstehen. Erklärungen sollten Nutzern früh präsentiert werden, auch wenn dies bedeutet, dass noch Fehler in den Erklärungen vorhanden sein könnten[Nou+21]. Für das Präsentieren der Erklärungen empfehlen [Nou+21] zu Beginn high-level-Erklärungen, um ein grundsätzliches Bild darüber zu schaffen, wie das System funktioniert. Danach sollten einzelne Instanzen gezeigt werden. Die Erklärmethode sollte hier bewusst gewählt werden.

3.2.15.2 Transparenz während der Implementation

Auch während der Implementation gibt es Maßnahmen, welche das ML-System transparenter machen. So empfiehlt z.B. [HT21] das Anwenden einfacher Methoden. Unterstützt werden kann

```
FROM DATA 'credit_data.csv'
TRAIN A 'decision tree'
PREDICTING 'default'
WRITE MODEL TO 'credit_score.model'
PROTECTED CLASSES 'race', 'gender', 'age'
REQUIRED FAIRNESS 'disparate impact' < 1.1
EXPLANATION 'decision_reason'
```

Abbildung 3.18: Beispiel Arbiter, entnommen aus [Zd20]

transparente Entwicklung durch eine Toolbox namens Bob [11]. Diese Toolbox bietet (Python-/C++)-Entwicklern Hilfestellungen bei der Siganlverarbeitung und dem ML. Unterstützt wird Reproduzierbarkeit durch Protokolle, aber auch durchschaubarer Code, Dokumentationen und Unit-Tests.

Den Ansatz der Vereinfachung verfolgen auch [Zd20], die eine eigene domänenspezifischen Sprache entwickelt haben, welche im Rahmen des ethischen ML neben Fairness, Zurechenbarkeit und Reproduzierbarkeit auch Transparenz gewährleistet. Unter einer domänenspezifischen Sprache, werden Programmiersprachen verstanden, welche nur in einem bestimmten Kontext eingesetzt werden. Hierdurch sind sie jedoch auch etwas begrenzter in ihrer Funktionalität. Diese domänenspezifische Sprache heißt *Arbiter* und ist außerdem deklarativ. Dies bedeutet, dass Nutzer angeben, welches Ziel zu erreichen ist und das System selbst bestimmen kann, wie dies zu erreichen ist [VH04]. Transparenz während der Entwicklung wird mithilfe von *Arbiter* vor allem dadurch geschaffen, dass Python Code kürzer dargestellt wird. Außerdem wird klarer herausgestellt, was eine bestimmte Zeile Code tut. Ein Beispiel für die Nutzung von *Arbiter* ist in Abbildung 3.18 zu sehen.

3.2.15.3 AutoML

Bei der Entwicklung eines ML-Systems kann es vorkommen, dass unterschiedliche Algorithmen ausprobiert werden. Zur automatisierten Unterstützung kann hierfür AutoML Anwendung finden. Dies sind Methoden, welche die Algorithmenauswahl und die Abstimmung der Parameter übernehmen, da es für Menschen zu aufwendig sein kann, dies händisch für eine Vielzahl an Möglichkeiten durchzuprobieren [Wan+19]. Um diesen automatischen Prozess transparent zu machen haben [Wan+19] das Tool *ATMSeer* entwickelt. Dies ist ein Visualisierungstool, welches es Entwicklern ermöglicht einen Überblick über den AutoML-Prozess zu bekommen oder Statistiken auf unterschiedlichen Granularitätsebenen anzeigen zu lassen.

3.2.15.4 Verteiltes Lernen

Eine weitere Besonderheit bei dem Trainieren von ML-Modellen während der Entwicklung ist das verteilte Lernen. Dies sollte ursprünglich Datenschutzprobleme lösen, jedoch arbeiten die meisten Ansätze in der Praxis mit zentralisierten Koordination, welche anfällig für Angriffe von außen sind. Weiter besteht ein Problem beim verteilten Lernen darin, dass nicht klar ist, woher welche Daten stammen [Ban+22]. [Ban+22] haben eine transparenzschaffende Maßnahme mithilfe der Blockchain-Technologie implementiert, welche Vertrauen und Transaprenz über die im verteilten Lernen erstellten Modelle schaffen soll. Ihre Lösung heißt Bassa-ML, welche ModelCards in einer Blockchain speichert und die Historie der verschiedenen Modelle

11. https://www.idiap.ch/software/bob/

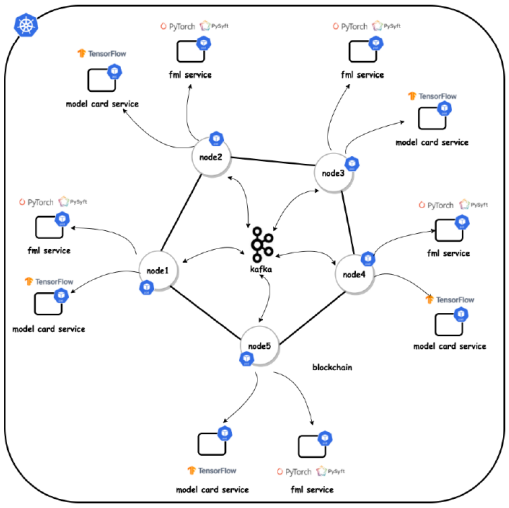


Abbildung 3.19: Architektur von Bassa-ML, entnommen aus: [Ban+22]

transparent macht. Jeder Teilnehmer des Netzwerkes erstellt lokale Modelle und die zugehörigen ModelCards, welche zu lokalen Modellen aggregiert werden können, wenn Blöcke für das Block-Chain-Netzwerk erstellt werden. Die grobe Systemarchitektur ist schematisch in Abbildung 3.19 dargestellt.

Auch [Zer+20] machen sich die Vorteile von Block-Chain-Ansätzen zu nutze. Sie entwickelten *Chained Distributed Machine learning (C-DistriM)*. Vorteile von solchen Ansätzen bestehen vor allem in der Transparenz über die Herkunft der Daten und in der Überwachung des Lernprozesses [Zer+20].

3.2.16 Dokumentation

3.2.16.1 Dokumentation des ML-Algorithmus

Für die Dokumentation des Algorithmus kann sich an Model Cards for Model Reporting orientiert werden.

Hier erkläre ich noch Model Cards for Model Reporting

Es existieren auch Beispiele für die Anwendung von Model Cards for Model Reporting in der medizinischen Branche [Sen+20] sowie der Biologie [Gra+20].

Daneben können zur Dokumentation des Algorithmus auch Grafiken genutzt werden, welche häufig schon während der Entwicklung angewendet werden. So könnte eine Confusion-Matrix [Has+22] genau so wie die Darstellung von Precision, Recall, Specificity und F1-Score [HT21] mit in die Dokumentation einfließen.

Hier werde ich noch kurz erklären, was die Metriken aussagen und vlt. Bilder einbauen

3.2.16.2 Dokumentation der Modellauswahl

Auch die Modellauswahl sollte dokumentiert werden. So kann von Interesse sein, welche Verfahren ausprobiert wurden. Weiter könnte dokumentiert werden, ob oder wann z.B. um Überanpassung zu vermeiden mit dem Training gestoppt wurde [De 18].

3.2.16.3 Ganzheitliche Dokumentation

Ein bekanntes Beispiel für die ganzheitliche Dokumentation eines ML-Systems ist ABOUT ML. Dies ist eine Sammlung von Best Pratices zur Dokumentation von Daten, Modellen und betrachtet hierbei den gesamten Lebenszyklus [VW20; RY19]. Daneben werden auch Artefakte oder Prozesse dokumentiert und es richtet sich an unterschiedliche Arten von Stakeholdern [RY19].

Ein Framework, an dem sich bei der Dokumentation orientiert werden kann, wurde von [HAP21] im Bereich der Luftfahrt entwickelt. Dieses sollte zwar ursprünglich setzt zwar den initialen Fokus darauf Vertrauen zu schaffen, aber kann auch zu mehr Transparenz beitragen. Aspekte, welche in die Dokumentation mit einfließen könnten, sind:

- Zweck des Systems: Was macht der Algorithmus? Welcher Algorithmus wurde aus welchen Gründen ausgewählt (mögliche Gründe: Performance oder Genauigkeit)?
- Technische Robustheit: Datenquellen (Qualität, Zugang, Integrität, Schutz und Sicherheit), Lebenszyklus, Zuverlässigkeit, Reproduzierbarkeit, Interpretierbarkeit/Erklärbarkeit
- Maßnahmen im Bereich der IT-Sicherheit
- Darstellung von nutzerfreundlichen Erklärungen sowie Reporting und Auditing

Ein weiteres Framework wurde von [GM22] entwickelt. Dieses legt zwar auch den Fokus eher auf Audits von ML-Systemen, kann jedoch auch dabei helfen ein System transparenter zu machen. In folgender Aufzählung sind kurz die wichtigsten Aspekte dargestellt, gekürzt um die Punkte, welche sich konkret auf das Durchführen von Audits beziehen:

Die beiden Checklisten gieße ich eventuell noch in den Text - auf jeden Fall fließen die Inhalte irgendwie in meine Handreichung ein, da besonders die zweite Abbildung meine dritte Forschungsfrage beantwortet

- Scoping: Dokumentation der Systemanforderungen und der angewendeten ML-Prinzipen, Use-Cases für ein ethisches Review und einer social impact analysis
- Mapping: Mapping von Stakeholdern auf Anforderungen, Dokumentation der Stakeholder-Interviews
- Artefakte: Checklisten, Model Cards, Datasheets for Datesets
- Testen: gegenseitige Prüfung der Dokumentation
- Reflektion: Zeitplan

Table 1: Stages of Machine Learning Lifecycle Addressed by Checklists

Checklist	ML Lifecycle Stage					
	Problem Definition	Dataset Procurement	Model Training	Product Test	Product Release	Product Monitoring
ABOUT ML	X	X	X	X	X	X
AI-TREE	X	X	X	X	X	X
Radiology's AI guide		X	X	X		
CLAIM	X	X	X	X		
CONSORT-AI	X			X		
Datasheets for datasets	X	X	X			
ECLAIR					X	X
FactSheets	X	X	X	X		X
Model cards	X		X	X		
Model Facts				X	X	X
PROBAST-AI		X	X	X		
Quality control questions				X	X	X
SPIRIT-AI	X			X		
STARD-AI	X	X	X	X		
The Dataset Nutrition Label		X	X			
The ML Test Score			X	X	X	X
TRIPOD-AI	X	X	X	X		

Note.—ABOUT ML = Annotation and Benchmarking on Understanding and Transparency of ML lifecycles, AI = artificial intelligence, AI-TREE = transparent, replicable, ethical and effective research in AI, CLAIM = Checklist for AI in Medical Imaging, CONSORT-AI = Consolidated Standards of Reporting Trials–AI, ECLAIR = evaluating commercial AI solutions in radiology, ML = machine learning, PROBAST-AI = prediction model risk of bias assessment tool–AI, SPIRIT-AI = Standard Protocol Items: Recommendations for Interventional Trials–AI, STARD-AI = Standards for Reporting of Diagnostic Accuracy Studies–AI, TRIPOD-AI = Transparent Reporting of a multivariable prediction model for Individual Prognosis or Diagnosis–AI.

Abbildung 3.20: Checkliste je ML-Stadium, entommen aus: [GM22]

Table 2: Roles Supported by Checklists

Checklist	Role						
	Data Scientist	ML Engineer	Product Manager	Clinician*	Clinician†	Scientific Author or Reviewer	Technician
ABOUT ML	X	X	X				
AI-TREE	X	X	X	X		X	
Radiology's AI guide	X	X	X	X		X	
CLAIM	X	X	X	X		X	
CONSORT-AI	X		X	X	X	X	
Datasheets for datasets	X	X	X			X	
ECLAIR			X	X	X		X
FactSheets			X	X	X	X	X
Model cards						X	X
Model Facts			X	X	X	X	X
PROBAST-AI			X	X		X	
Quality control questions		X	X	X	X		X
SPIRIT-AI	X		X	X	X	X	
STARD-AI	X	X	X	X	X	X	
The Dataset Nutrition Label	X	X					
The ML Test Score		X	X				X
TRIPOD-AI	X	X	X	X	X	X	

Note.— ABOUT ML = Annotation and Benchmarking on Understanding and Transparency of ML Lifecycles, AI = artificial intelligence, AI-TREE = transparent, replicable, ethical and effective research in AI, CLAIM = Checklist for AI in Medical Imaging, CONSORT-AI = Consolidated Standards of Reporting Trials–AI, ECLAIR = evaluating commercial AI solutions in radiology, ML = machine learning, PROBAST-AI = prediction model risk of bias assessment tool–AI, SPIRIT-AI = Standard Protocol Items: Recommendations for Interventional Trials–AI, STARD-AI = Standards for Reporting of Diagnostic Accuracy Studies–AI, TRIPOD-AI = Transparent Reporting of a multivariable prediction model for Individual Prognosis or Diagnosis–AI.
* This refers to clinicians evaluating a product.
† This refers to clinicians using a product.

Abbildung 3.21: Checkliste je Publikum, entommen aus: [GM22]

Zusätzlich geben [GM22] einen Überblick über mögliche Checklisten und geben an, welche Checkliste in welchem zeitlichen Stadium der ML-Entwicklung 3.20 angewendet wird und an wen diese sich richtet 3.21.

Weiterführend finden sich in der Veröffentlichung von [GM22] Hinweise darüber, wann welche Dokumentationsmethode eingesetzt werden kann. Konkret gehen sie auf Datasheets for Datasets, Modelcards und Modelfact ein (siehe Abbildung 3.22).

3.2.17 Probleme mit Transparenz

Bisher wurden in der Arbeit neben transparenzschaffenden Maßnahmen vor allem die positiven Aspekte von Transparenz im ML herausgestellt. Jedoch ergab die Literaturrecherche auch einige

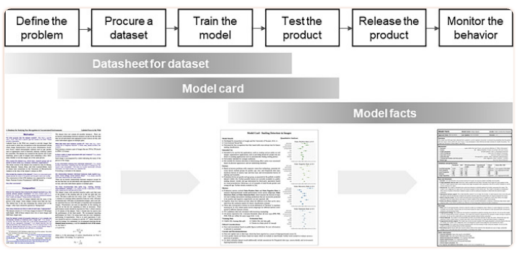


Abbildung 3.22: Übersicht XAI-Methode mit Audience, entommen aus: [GM22]

negative Aspekte von Transparenz oder deckte Bereiche auf, in denen noch Forschungs- oder Klärungsbedarf besteht.

3.2.17.1 Offene Fragen bei Transparenz

[ZC18] beschreiben das Konzept des transparenten ML (vgl. Kapitel 3.2.1) und führen hierbei zusätzlich offene Fragen an, die von ihrem Konzept noch nicht abgedeckt werden. So beschreiben sie, dass grundsätzlich Vertrauen bei transparenten ML zwar wichtig ist, jedoch wenig darüber bekannt ist, wie Vertrauen in Menschen entsteht. [VW20] führen hierzu an, dass es vorkomme, dass Menschen einem ML-System mehr vertrauen, welches viele Features benutzt, obwohl sie es nicht verstehen. Ein Modell, welches sie verstehen, aber beispielsweise nur wenige Features nutze, führt zu weniger Vertrauen in das System. Weiter bewege nach [ZC18] Nutzer von ML-Systemen das Risiko bei der Nutzung eines solchen Systems. So seien diese Systeme nicht perfekt und ideal wäre eine Abwägung der Kosten, wenn auf Grundlage eines Systems in der realen Welt eine falsche Entscheidung getroffen wird. Weiter leiden bisherige XAI-Methoden vor allem unter einer Verallgemeinerung der Erklärungsempfänger. So bringen alle menschlichen Betrachter unterschiedliche Bildung, Kultur oder Geschlecht mit, was die Anforderungen an die Art und Weise der geforderten Kommunikation bedingt. Besonders zu viel Transparenz und die Kommunikation besonders komplizierter Modelle sei nicht geeignet für Fachfremde, da hier das technische Hintergrundwissen nicht vorhanden ist [De 18]. Des Weiteren ist schwer zu entscheiden, welches Maß an Transparenz gut ist. Zu wenig Transparenz kann problematisch werden [Kul+13] und wichtige Aspekte werden einfach ausgelassen [Bla18]. Zu viel Transparenz und zu viele Erklärungen führen jedoch dazu, dass Nutzer sehr viel Zeit und unter Umständen Wissensakkumulation dafür aufbringen müssen, die Erklärungen zu verstehen [Bla18]. [Bla18] sprechen hier von dem Transparenzparadoxon. Auch [BT21] fanden heraus, dass das Erstellen von „guten und nutzerfreundlichen“ Erklärungen nicht trivial ist. Sie führten einen Touringtest durch, in denen die Studienteilnehmer bestimmen sollten, ob eine vorliegende Erklärung (Markierung von 3 Buzz-Words bei einer Textklassifikation) von einem Menschen oder einer angewendeten XAI-Methode getroffen wurde. Ergebnis der Studie war, dass Menschen später nicht in der Lage waren, zu sagen, ob eine Klassifikationsentscheidung automatisiert oder von Hand getroffen wurde.

3.2.17.2 Zeitaufwand

Des Weiteren ist es laut [VW20] vorgekommen, dass Nutzer das Erstellen eines mentalen Modells mithilfe von Erklärungen als Zeitverschwendung betrachteten.

3.2.17.3 Gefahren durch Transparenz

Neben den bereits angeführten Problemen können einige Aspekte von Transparenz jedoch auch schwerwiegende Folgen in der Praxis haben, wenn Systeme z.B. in der Medizin oder im Verkehr eingesetzt werden. Erklärungen lassen sich z.B. manipulieren, sodass Nutzer den Erklärungen nicht mehr trauen können [TG20]. Ein Beispiel hierzu wird von [GAZ19] beschrieben, das zeigt, wie aus einem originalen Bild ein für den Nutzer kaum zu unterscheidbares zweites Bild erstellt werden kann, dass jedoch eine gänzlich andere Erklärung erhält. Neben funktionalen Problemen,

welche kritisch für das Vertrauen und die korrekte Anwendung sind, bestehen noch weitere Probleme in Bezug auf Datenschutz [De 18]. Dies liegt auf der Hand, wenn wie in

Hier verweise ich noch auf die Stelle (die habe ich nur noch nicht geschrieben :D)

beschrieben konkrete Beispiele oder Auszüge aus den Trainingsdaten offen gelegt werden, jedoch existieren auch subtilere Angriffe auf die Privatsphäre bei ML-Systemen. [Sho+17] beschreiben die „Membership Inference Attack“, welche sich bei ML-Algorithmen anwenden lässt und es einem Angreifer ermöglicht herauszufinden, ob ein bestimmter Datenpunkt in dem genutzten Trainingsdatensatz vorhanden war.

Eidesstattliche Versicherung

Bitte verwenden Sie hier in jedem Fall die offizielle von der Prüfungsbehörde vorgegebene Formulierung der Selbständigkeitserklärung.

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ggf. streichen: Ich bin damit einverstanden, dass meine Abschlussarbeit in den Bestand der Fachbereichsbibliothek eingestellt wird.

Hamburg, den 24. August 2022

Lynn Oesterwind

Literatur

[Alp19] Ethem Alpaydin. *Maschinelles Lernen*. De Gruyter Oldenbourg, 2019.

[ALW09] Raja Noor Ainon, Adel Lahsasna und Teh Ying Wah. *A transparent classification model using a hybrid soft computing method*. In: *2009 Third Asia International Conference on Modelling & Simulation*. IEEE. 2009, S. 146–151.

[Ame+15] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard und Jina Suh. *Modeltracker: Redesigning performance analysis tools for machine learning*. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, S. 337–346.

[ANK18] Jafar Alzubi, Anand Nayyar und Akshi Kumar. *Machine learning from theory to algorithms: an overview*. In: *Journal of physics: conference series*. Bd. 1142. 1. IOP Publishing. 2018, S. 012012.

[Ant+21] Anna Markella Antoniadi, Yuhan Du, Yasmine Guendouz, Lan Wei, Claudia Mazo, Brett A Becker und Catherine Mooney. *Current challenges and future opportunities for XAI in machine learning-based clinical decision support systems: a systematic review*. In: *Applied Sciences* 11.11 (2021), S. 5088.

[Ary+21] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović et al. *AI Explainability 360 Toolkit*. In: *8th ACM IKDD CODS and 26th COMAD*. 2021, S. 376–379.

[Ayo10] Taiwo Ayodele. *Types of Machine Learning Algorithms*. In: Feb. 2010.

[AZ20] Daniel W Apley und Jingyu Zhu. *Visualizing the effects of predictor variables in black box supervised learning models*. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82.4 (2020), S. 1059–1086.

[BA10] Mark Bedner und Tobias Ackermann. *Schutzziele der IT-Sicherheit*. In: *Datenschutz und Datensicherheit-DuD* 34.5 (2010), S. 323–328.

[Bac+15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller und Wojciech Samek. *On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation*. In: *PloS one* 10.7 (2015), e0130140.

[Bac16] Klaus Backhaus. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 14. Aufl. 2016. Berlin, Heidelberg: Springer Gabler, 2016.

[Ban+22] Eranga Bandara, Sachin Shetty, Abdul Rahman, Ravi Mukkamala, Juan Zhao und Xueping Liang. *Bassa-ML—A Blockchain and Model Card Integrated Federated Learning Provenance Platform*. In: *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2022, S. 753–759.

[BCK03] Len Bass, Paul Clements und Rick Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.

[Beh15] Joachim Behnke. *Logistische Regressionsanalyse: Eine Einführung*. Wiesbaden: Springer VS, 2015.

[BK98] Christian Borgelt und Rudolf Kruse. *Attributauswahlmaße für die induktion von entscheidungsbäumen: Ein überblick*. In: *Data Mining*. Springer. 1998, S. 77–98.

[Bla18] Christina Blacklaws. *Algorithms: transparency and accountability*. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2128 (2018), S. 20170351.

[BÖ14] Yalin Baştanlar und Mustafa Özuysal. *Introduction to machine learning*. In: *miRNomics: MicroRNA biology and computational analysis* (2014), S. 105–128.

[BPV16] Prafulla Bafna, Dhanya Pramod und Anagha Vaidya. *Document clustering: TF-IDF approach*. In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE. 2016, S. 61–66.

[Bre+17] Leo Breiman, Jerome H. Friedman, Richard A. Olshen und Charles J. Stone. *Classification And Regression Trees*. Routledge, 2017.

[Bri13] Alfred Brink. *Anfertigung wissenschaftlicher Arbeiten: Ein prozessorientierter Leitfaden zur Erstellung von Bachelor-, Master- und Diplomarbeiten*. 4., korr. und aktualisierte Aufl. 2013. Springer eBook Collection. Wiesbaden: Springer Gabler, 2013. ISBN: 978-3-8349-4397-2. DOI: 10.1007/978-3-8349-4397-2.

[Bro+09] Jan vom Brocke, Alexander Simons, Björn Niehaves, Kai Reimer, Ralf Plattfaut und Anne Cleven. *RECONSTRUCTING THE GIANT: ON THE IMPORTANCE OF RIGOUR IN DOCUMENTING THE LITERATURE SEARCH PROCESS*. en. In: *ECIS 2009 Proceedings*. Bd. 161. AIS Electronic Library (AISeL), 2009, S. 14. URL: https://aisel.aisnet.org/ecis2009.

[BT21] Felix Biessmann und Viktor Treu. *A Turing Test for Transparency*. In: *arXiv preprint arXiv:2106.11394* (2021).

[Cad17] Field Cady. *The data science handbook*. Hoboken, NJ: John Wiley & Sons Inc, 2017.

[Cai+19] Carrie J Cai, Samantha Winter, David Steiner, Lauren Wilcox und Michael Terry. *"Hello AI": uncovering the onboarding needs of medical practitioners for human-AI collaborative decision-making*. In: *Proceedings of the ACM on Human-computer Interaction* 3.CSCW (2019), S. 1–24.

[Che+14] Jessie Y Chen, Katelyn Procci, Michael Boyce, Julia Wright, Andre Garcia und Michael Barnes. *Situation awareness-based agent transparency*. Techn. Ber. Army research lab aberdeen proving ground md human research und engineering . . ., 2014.

[Che+16] Dong Chen, Rachel KE Bellamy, Peter K Malkin und Thomas Erickson. *Diagnostic visualization for non-expert machine learning practitioners: A design study*. In: *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2016, S. 87–95.

[Che+21] Qili Chen, Guangyuan Pan, Wenbai Chen und Peiliang Wu. *A novel explainable deep belief network framework and its application for feature importance analysis*. In: *IEEE Sensors Journal* 21.22 (2021), S. 25001–25009.

[Cho+20] Kenny Choo, Eliska Greplova, Mark H. Fischer und Titus Neupert. *Machine Learning kompakt: Ein Einstieg für Studierende der Naturwissenschaften*. Wiesbaden: Springer Spektrum, 2020.

[CL20] Jürgen Cleve und Uwe Lämmel. *Data Mining*. 3. Auflage. Berlin und Boston: De Gruyter, 2020.

[CML18] Zhiyuan Chen, Nianzu Ma und Bing Liu. *Lifelong learning for sentiment classification*. In: *arXiv preprint arXiv:1801.02808* (2018).

[Col+19] Cody Coleman, Daniel Kang, Deepak Narayanan, Luigi Nardi, Tian Zhao, Jian Zhang, Peter Bailis, Kunle Olukotun, Chris Ré und Matei Zaharia. *Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark*. In: *ACM SIGOPS Operating Systems Review* 53.1 (2019), S. 14–25.

[Coo88] Harris M. Cooper. *Organizing knowledge syntheses: A taxonomy of literature reviews*. en. In: *Knowledge in Society* 1.1 (März 1988), S. 104–126. ISSN: 0897-1986. DOI: 10.1007/BF03177550. URL: http://link.springer.com/10.1007/BF03177550 (besucht am 05. 03. 2022).

[De 18] Paul B De Laat. *Algorithmic decision-making based on machine learning from big data: can transparency restore accountability?* In: *Philosophy & technology* 31.4 (2018), S. 525–541.

[Den+20] Emily Denton, Alex Hanna, Razvan Amironesei, Andrew Smart, Hilary Nicole und Morgan Klaus Scheuerman. *Bringing the people back in: Contesting benchmark machine learning datasets*. In: *arXiv preprint arXiv:2007.07399* (2020).

[DF12] Frank Dehen und Jörg Feldhusen. *Bayes-Netzwerke für die Kostenprognose in der frühen Phase der Produktentwicklung*. Shaker, 2012.

[DGW18] Sanjeeb Dash, Oktay Gunluk und Dennis Wei. *Boolean decision rules via column generation*. In: *Advances in neural information processing systems* 31 (2018).

[Dhu+18a] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam und Payel Das. *Explanations based on the missing: Towards contrastive explanations with pertinent negatives*. In: *Advances in neural information processing systems* 31 (2018).

[Dhu+18b] Amit Dhurandhar, Karthikeyan Shanmugam, Ronny Luss und Peder A Olsen. *Improving simple models with confidence profiles*. In: *Advances in Neural Information Processing Systems* 31 (2018).

[Dhu+19] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam und Ruchir Puri. *Model agnostic contrastive explanations for structured data*. In: *arXiv preprint arXiv:1906.00117* (2019).

[Döb+18] Inga Döbel, Miriam Leis, Manuel Molina Vogelsang, Dmitry Neustroev, Henning Petzka, Stefan Rüping, Angelika Voss, Martin Wegele und Juliane Welz. *Maschinelles Lernen–Kompetenzen, Anwendungen und Forschungsbedarf*. In: *Fraunhofer IAIS, Fraunhofer IMW, Fraunhofer Zentrale. Zugriff am* 21 (2018), S. 2020.

[Dör18] Sebastian Dörn. *Programmieren für Ingenieure und Naturwissenschaftler: Intelligente Algorithmen und digitale Technologien*. 1. Aufl. 2018. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.

[DSB17] Derek Doran, Sarah Schulz und Tarek R Besold. *What does explainable AI really mean? A new conceptualization of perspectives*. In: *arXiv preprint arXiv:1710.00794* (2017).

[Dud] Duden. *Deep Learning vs. Machine Learning – What's The Difference?* URL: https://www.duden.de/rechtschreibung/Transparenz (besucht am 10. 06. 2022).

[DZ11] Mita K Dalal und Mukesh A Zaveri. *Automatic text classification: a technical review*. In: *International Journal of Computer Applications* 28.2 (2011), S. 37–40.

[EM15] Issam El Naqa und Martin J. Murphy. *What Is Machine Learning?* In: *Machine Learning in Radiation Oncology*. Hrsg. von Issam El Naqa, Ruijiang Li und Martin J. Murphy. Bd. 3. Cham: Springer International Publishing, 2015, S. 3–11.

[Ert21] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. 5. Auflage. Springer eBook Collection. Wiesbaden: Springer Vieweg, 2021. ISBN: 978-3-658-32075-1.

[EU21] EU. *Vorschlag für eine Verordnung des Europäischen Parlaments und des Rates zur Festlegung harmonisierter Vorschriften für künstliche Intelligenz (Gesetz über künstliche Intelligenz) und zur Änderung bestimmter Rechtsakte der Union*. 2021.

[Fin20] Verena Fink. *Quick Guide KI-Projekte–einfach machen: Künstliche Intelligenz in Service, Marketing und Sales erfolgreich einführen*. Springer-Verlag, 2020.

[Fio20] Luigi Fiori. *K-Means Clustering using Python*. 2020. URL: https://medium.com/@luigi.fiori.lf0303/k-means-clustering-using-python-db57415d26e6/.

[FPS96] Usama Fayyad, Gregory Piatetsky-Shapiro und Padhraic Smyth. *From data mining to knowledge discovery in databases*. In: *AI magazine* 17.3 (1996), S. 37–37.

[Fri01] Jerome H Friedman. *Greedy function approximation: a gradient boosting machine*. In: *Annals of statistics* (2001), S. 1189–1232.

[GAZ19] Amirata Ghorbani, Abubakar Abid und James Zou. *Interpretation of neural networks is fragile*. In: *Proceedings of the AAAI conference on artificial intelligence*. Bd. 33. 01. 2019, S. 3681–3688.

[Geb+21] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii und Kate Crawford. *Datasheets for datasets*. In: *Communications of the ACM* 64.12 (2021), S. 86–92.

[Gia21] Leonida Gianfagna. *Explainable AI with Python*. Cham: Springer International Publishing AG, 2021. ISBN: 978-3-030-68639-0. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6578043.

[GM22] Christian Garbin und Oge Marques. *Assessing Methods and Tools to Improve Reporting, Increase Transparency, and Reduce Failures in Machine Learning Applications in Health Care*. In: *Radiology: Artificial Intelligence* 4.2 (2022).

[Gol19] Jake Goldenfein. *Algorithmic transparency and decision-making accountability: Thoughts for buying machine learning algorithms*. In: *Jake Goldenfein,'Algorithmic Transparency and Decision-Making Accountability: Thoughts for buying machine learning algorithms' in Office of the Victorian Information Commissioner (ed), Closer to the Machine: Technical, Social, and Legal aspects of AI (2019)* (2019).

[Gom+21] Oscar Gomez, Steffen Holter, Jun Yuan und Enrico Bertini. *AdViCE: Aggregated Visual Counterfactual Explanations for Machine Learning Model Validation*. In: *2021 IEEE Visualization Conference (VIS)*. IEEE. 2021, S. 31–35.

[Goo22] Google. *Visualizations for ML datasets*. 2022. URL: https://pair-code.github.io/facets/.

[Gra+20] Isabella Grasso, David Russell, Abigail Matthews, Jeanna Matthews und Nicholas R Record. *Applying algorithmic accountability frameworks with domain-specific codes of ethics: A case study in ecosystem forecasting for shellfish toxicity in the Gulf of Maine*. In: *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*. 2020, S. 83–91.

[Gui+18a] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini und Fosca Giannotti. *Local rule-based explanations of black box decision systems*. In: *arXiv preprint arXiv:1805.10820* (2018).

[Gui+18b] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti und Dino Pedreschi. *A survey of methods for explaining black box models*. In: *ACM computing surveys (CSUR)* 51.5 (2018), S. 1–42.

[Gur+19] Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi und Charu Aggarwal. *Efficient data representation by selecting prototypes with importance weights*. In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2019, S. 260–269.

[HAP21] Carolina Sanchez Hernandez, Samuel Ayo und Dimitrios Panagiotakopoulos. *An Explainable Artificial Intelligence (xAI) Framework for Improving Trust in Automated ATM Tools*. In: *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. IEEE. 2021, S. 1–10.

[Has+22] Khan Md Hasib, Farhana Rahman, Rashik Hasnat und Md Golam Rabiul Alam. *A Machine Learning and Explainable AI Approach for Predicting Secondary School Student Performance*. In: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. 2022, S. 0399–0405.

[HBV01] Maria Halkidi, Yannis Batistakis und Michalis Vazirgiannis. *On clustering validation techniques*. In: *Journal of intelligent information systems* 17.2 (2001), S. 107–145.

[Hil+18] Flynn Hill, David Fulcher, Rory Sie und Maarten De Laat. *Balancing accuracy and transparency in early alert identification of students at risk*. In: *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE. 2018, S. 1125–1128.

[Hin+19] Michael Hind, Dennis Wei, Murray Campbell, Noel CF Codella, Amit Dhurandhar, Aleksandra Mojsilović, Karthikeyan Natesan Ramamurthy und Kush R Varshney. *TED: Teaching AI to explain its decisions*. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 2019, S. 123–129.

[Hoh+19] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine und Steven M Drucker. *Gamut: A design probe to understand how data scientists understand machine learning models*. In: *Proceedings of the 2019 CHI conference on human factors in computing systems*. 2019, S. 1–13.

[Hol18] Andreas Holzinger. *From machine learning to explainable AI*. In: *2018 world symposium on digital intelligence for systems and machines (DISA)*. IEEE. 2018, S. 55–66.

[HT21] Aleks Huč und Denis Trček. *Anomaly detection in IoT networks: From architectures to machine learning transparency*. In: *IEEE Access* 9 (2021), S. 60607–60616.

[HZW21] Ambreen Hanif, Xuyun Zhang und Steven Wood. *A Survey on Explainable Artificial Intelligence Techniques and Challenges*. In: *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE. 2021, S. 81–89.

[IBM] IBM. *CRISP-DM Dokumentation*. URL: https://www.ibm.com/docs/de/spss-modeler/SaaS?topic=dm-crisp-help-overview (besucht am 23. 05. 2022).

[Iri22] K Irion. *Algorithms Off-Limits?* In: (2022).

[JZH21] Christian Janiesch, Patrick Zschech und Kai Heinrich. *Machine learning and deep learning*. In: *Electronic Markets* 31.3 (2021), S. 685–695.

[Kel+20] Niklas Keller, Mirjam A Jenny, Claudia A Spies und Stefan M Herzog. *Augmenting Decision Competence in Healthcare Using AI-based Cognitive Models*. In: *2020 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE. 2020, S. 1–4.

[Ker+20] Sören Kerner, Jens Leveling, Oliver Urbann, Luise Weickhmann, Maximilian Otten und Maurice Vogel. *Anwendungsfelder von künstlicher Intelligenz in Industrie-4.0-Systemen*. In: *Handbuch Industrie 4.0: Band 3: Logistik*. Hrsg. von Michael ten Hompel, Thomas Bauernhansl und Birgit Vogel-Heuser. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 227–250.

[KHA20] Dae Hyun Kim, Enamul Hoque und Maneesh Agrawala. *Answering questions about charts and generating visual explanations*. In: *Proceedings of the 2020 CHI conference on human factors in computing systems*. 2020, S. 1–13.

[Kim+18] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas et al. *Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)*. In: *International conference on machine learning*. PMLR. 2018, S. 2668–2677.

[Kim21] Seungjun Kim. *Explainable AI (XAI) Methods Part 2— Individual Conditional Expectation (ICE) Curves*. 2021. URL: https://towardsdatascience.com/explainable-ai-xai-methods-part-2-individual-conditional-expectation-ice-curves-8fe76919aab7.

[KKH22] Dominik Kreuzberger, Niklas Kühl und Sebastian Hirschl. *Machine Learning Operations (MLOps): Overview, Definition, and Architecture*. In: *arXiv preprint arXiv:2205.02302* (2022).

[KKK16] Been Kim, Rajiv Khanna und Oluwasanmi O Koyejo. *Examples are not enough, learn to criticize! criticism for interpretability*. In: *Advances in neural information processing systems* 29 (2016).

[KL19] Kristian Kersting und Christoph Lampert. *Wie Maschinen lernen: Künstliche Intelligenz verständlich erklärt*. Wiesbaden: Springer, 2019.

[KL21] Uday Kamath und John Liu. *Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning*. Springer, 2021.

[KM21] Markus Kalisch und Lukas Meier. *Logistische Regression: Eine anwendungsorientierte Einführung mit R*. Springer Nature, 2021.

[KPN16] Josua Krause, Adam Perer und Kenney Ng. *Interacting with predictions: Visual inspection of black-box machine learning models*. In: *Proceedings of the 2016 CHI conference on human factors in computing systems*. 2016, S. 5686–5697.

[Kra+17] Josua Krause, Aritra Dasgupta, Jordan Swartz, Yindalon Aphinyanaphongs und Enrico Bertini. *A workflow for visual diagnostics of binary classifiers using instance-level explanations*. In: *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE. 2017, S. 162–172.

[KSB17] Abhishek Kumar, Prasanna Sattigeri und Avinash Balakrishnan. *Variational inference of disentangled latent concepts from unlabeled observations*. In: *arXiv preprint arXiv:1711.00848* (2017).

[KT21] Florian Königstorfer und Stefan Thalmann. *Software documentation is not enough! Requirements for the documentation of AI*. In: *Digital Policy, Regulation and Governance* (2021).

[Kul+13] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan und Weng-Keen Wong. *Too much, too little, or just right? Ways explanations impact end users' mental models*. In: *2013 IEEE Symposium on visual languages and human centric computing*. IEEE. 2013, S. 3–10.

[KZP+07] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas et al. *Supervised machine learning: A review of classification techniques*. In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), S. 3–24.

[LL17] Scott M Lundberg und Su-In Lee. *A unified approach to interpreting model predictions*. In: *Advances in neural information processing systems* 30 (2017).

[Lor20] Uwe Lorenz. *Reinforcement Learning: Aktuelle Ansätze Verstehen - Mit Beispielen in Java und Greenfoot*. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2020.

[LPK20] Pantelis Linardatos, Vasilis Papastefanopoulos und Sotiris Kotsiantis. *Explainable ai: A review of machine learning interpretability methods*. In: *Entropy* 23.1 (2020), S. 18.

[Lus+19] Ronny Luss, Pin-Yu Chen, Amit Dhurandhar, Prasanna Sattigeri, Karthikeyan Shanmugam und Chun-Chen Tu. *Generating contrastive explanations with monotonic attribute functions*. In: (2019).

[Lyo13] Joseph B Lyons. *Being transparent about transparency: A model for human-robot interaction*. In: *2013 AAAI Spring Symposium Series*. 2013.

[Mat21] Stephan Matzka. *Künstliche Intelligenz in den Ingenieurwissenschaften: Maschinelles Lernen verstehen und bewerten*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021.

[MG18] Agnieszka Mikołajczyk und Michał Grochowski. *Data augmentation for improving deep learning in image classification problem*. In: *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE. 2018, S. 117–122.

[Mic22] Microsoft. *How to select algorithms for Azure Machine Learning*. 2022. URL: https://docs.microsoft.com/en-us/azure/machine-learning/how-to-select-algorithms (besucht am 28. 03. 2022).

[Mol22] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2. Aufl. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[Mon+19] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek und Klaus-Robert Müller. *Layer-wise relevance propagation: an overview*. In: *Explainable AI: interpreting, explaining and visualizing deep learning* (2019), S. 193–209.

[MST20] Ramaravind K Mothilal, Amit Sharma und Chenhao Tan. *Explaining machine learning classifiers through diverse counterfactual explanations*. In: *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 2020, S. 607–617.

[Mül19] Müller, Tobias. *Spielarten der Künstlichen Intelligenz: Maschinelles Lernen und Künstliche Neuronale Netze*. 24. Mai 2019. URL: https://blog.iao.fraunhofer.de/spielarten-der-kuenstlichen-intelligenz-maschinelles-lernen-und-kuenstliche-neuronale-netze/ (besucht am 08. 06. 2022).

[Ngu+21] Quang Hung Nguyen, Hai-Bang Ly, Lanh Si Ho, Nadhir Al-Ansari, Hiep Van Le, Van Quan Tran, Indra Prakash und Binh Thai Pham. *Influence of data splitting on performance of machine learning models in prediction of shear strength of soil*. In: *Mathematical Problems in Engineering* 2021 (2021).

[Nob06] William S Noble. *What is a support vector machine?* In: *Nature biotechnology* 24.12 (2006), S. 1565–1567.

[Nou+21] Mahsan Nourani, Chiradeep Roy, Jeremy E Block, Donald R Honeycutt, Tahrima Rahman, Eric Ragan und Vibhav Gogate. *Anchoring Bias Affects Mental Model Formation and User Reliance in Explainable AI Systems*. In: *26th International Conference on Intelligent User Interfaces*. 2021, S. 340–350.

[NP22] Anirban Nandi und Aditya Kumar Pal. *Interpreting Machine Learning Models*. Berkeley, CA: Apress, 2022. ISBN: 978-1-4842-7801-7. DOI: 10.1007/978-1-4842-7802-4.

[NS18] Annalyn Ng und Kenneth Soo. *Data Science – was ist das eigentlich?!* Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.

[Pal+22] Reshika Palaniyappan Velumani, Meng Xia, Jun Han, Chaoli Wang, ALEXIS K LAU und Huamin Qu. *AQX: Explaining Air Quality Forecast for Verifying Domain Knowledge using Feature Importance Visualization*. In: *27th International Conference on Intelligent User Interfaces*. 2022, S. 720–733.

[Par20] Europäisches Parlament. *Was ist künstliche Intelligenz und wie wird sie genutzt?* 2020. URL: https://www.europarl.europa.eu/news/de/headlines/society/20200827STO85804/was-ist-kunstliche-intelligenz-und-wie-wird-sie-genutzt.

[Pea01] Karl Pearson. *LIII. On lines and planes of closest fit to systems of points in space*. In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), S. 559–572.

[Per09] Patrick O Perry. *Cross-validation for unsupervised learning*. Stanford University, 2009.

[Rao20] Rahul Raoniar. *Diabetes Prediction Model Explanation using LIME*. 2020. URL: https://onezero.blog/diabetes-prediction-model-explanation-using-lime/.

[Ras18] Sebastian Raschka. *Model evaluation, model selection, and algorithm selection in machine learning*. In: *arXiv preprint arXiv:1811.12808* (2018).

[RJ+20] Fahimeh Rajabiyazdi, Greg A Jamieson et al. *A Machine Learning-Based Micro-World Platform for Condition-Based Maintenance*. In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2020, S. 288–295.

[RK87] LKPJ Rdusseeun und P Kaufman. *Clustering by means of medoids*. In: *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*. Bd. 31. 1987.

[Rod99] David M Rodvold. *A software development process model for artificial neural networks in critical applications*. In: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. Bd. 5. IEEE. 1999, S. 3317–3322.

[Roj01] Raúl Rojas. *Künstliche neuronale Netze als neues Paradigma der Informationsverarbeitung*. In: *Neurowissenschaften und Philosophie: eine Einführung. Frankfurt am Main: UTB* (2001), S. 269–297.

[RSG16] Marco Tulio Ribeiro, Sameer Singh und Carlos Guestrin. *"Why should i trust you?Ëxplaining the predictions of any classifier*. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, S. 1135–1144.

[RSG18] Marco Tulio Ribeiro, Sameer Singh und Carlos Guestrin. *Anchors: High-precision model-agnostic explanations*. In: *Proceedings of the AAAI conference on artificial intelligence*. Bd. 32. 1. 2018.

[Rud19] Cynthia Rudin. *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*. In: *Nature Machine Intelligence* 1.5 (2019), S. 206–215.

[RY19] Inioluwa Deborah Raji und Jingying Yang. *About ml: Annotation and benchmarking on understanding and transparency of machine learning lifecycles*. In: *arXiv preprint arXiv:1912.06166* (2019).

[Sar21] Iqbal H Sarker. *Machine learning: Algorithms, real-world applications and research directions*. In: *SN Computer Science* 2.3 (2021), S. 1–21.

[Sch22] Jakob Schoeffer. *A Human-Centric Perspective on Fairness and Transparency in Algorithmic Decision-Making*. In: *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 2022, S. 1–6.

[SCN18] Jatinder Singh, Jennifer Cobbe und Chris Norval. *Decision provenance: Harnessing data flow for accountable systems*. In: *IEEE Access* 7 (2018), S. 6562–6574.

[SD05] Qiang Sun und Gerald DeJong. *Explanation-augmented svm: an approach to incorporating domain knowledge into svm learning*. In: *Proceedings of the 22nd international conference on Machine learning*. 2005, S. 864–871.

[Sen+20] Mark Sendak, Madeleine Clare Elish, Michael Gao, Joseph Futoma, William Ratliff, Marshall Nichols, Armando Bedoya, Suresh Balu und Cara O'Brien. *"The human body is a black boxßupporting clinical decision-making with deep learning*. In: *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 2020, S. 99–109.

[SG21] Harini Suresh und John Guttag. *A framework for understanding sources of harm throughout the machine learning life cycle*. In: *Equity and access in algorithms, mechanisms, and optimization*. 2021, S. 1–9.

[SGK17] Avanti Shrikumar, Peyton Greenside und Anshul Kundaje. *Learning important features through propagating activation differences*. In: *International conference on machine learning*. PMLR. 2017, S. 3145–3153.

[Sha51] Lloyd S Shapley. *Notes on the N-person Game–I: Characteristic-point Solutions of the Four-person Game*. Rand Corporation, 1951.

[Sho+17] Reza Shokri, Marco Stronati, Congzheng Song und Vitaly Shmatikov. *Membership inference attacks against machine learning models*. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, S. 3–18.

[Sla+20] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh und Himabindu Lakkaraju. *Fooling lime and shap: Adversarial attacks on post hoc explanation methods*. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, S. 180–186.

[SM18] Raymond Sheh und Isaac Monteath. *Defining explainable ai for requirements analysis*. In: *KI-Künstliche Intelligenz* 32.4 (2018), S. 261–266.

[SS18] Pramila Shinde und Seema Shah. *A Review of Machine Learning and Deep Learning Applications*. In: *2018 Fourth International Conference on Computing Communication Control and Automation*. 2018.

[Str19] Martin Strobel. *Aspects of transparency in machine learning*. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2019, S. 2449–2451.

[SVZ13] Karen Simonyan, Andrea Vedaldi und Andrew Zisserman. *Deep inside convolutional networks: Visualising image classification models and saliency maps*. In: *arXiv preprint arXiv:1312.6034* (2013).

[SW18] Keng Siau und Weiyu Wang. *Building trust in artificial intelligence, machine learning, and robotics*. In: *Cutter business technology journal* 31.2 (2018), S. 47–53.

[SZI20] Donghee Shin, Bouziane Zaid und Mohammed Ibahrine. *Algorithm appreciation: Algorithmic performance, developmental processes, and user interactions*. In: *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. IEEE. 2020, S. 1–5.

[TG20] Erico Tjoa und Cuntai Guan. *A survey on explainable artificial intelligence (xai): Toward medical xai*. In: *IEEE transactions on neural networks and learning systems* 32.11 (2020), S. 4793–4813.

[TKC16] Gary KL Tam, Vivek Kothari und Min Chen. *An analysis of machine-and human-analytics in classification*. In: *IEEE transactions on visualization and computer graphics* 23.1 (2016), S. 71–80.

[TR21] Maria Tsiakmaki und Omiros Ragos. *A Case Study of Interpretable Counterfactual Explanations for the Task of Predicting Student Academic Performance*. In: *2021 25th International Conference on Circuits, Systems, Communications and Computers (CSCC)*. IEEE. 2021, S. 120–125.

[Vac+18] Kristen Vaccaro, Dylan Huang, Motahhare Eslami, Christian Sandvig, Kevin Hamilton und Karrie Karahalios. *The illusion of control: Placebo effects of control settings*. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, S. 1–13.

[Ver20] Vaibhav Verdhan. *Supervised Learning with Python*. Berkeley, CA: Apress, 2020.

[VH04] Peter Van Roy und Seif Haridi. *Concepts, techniques, and models of computer programming*. MIT press, 2004.

[Vor18] Eric S Vorm. *Assessing demand for transparency in intelligent systems using machine learning*. In: *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE. 2018, S. 1–7.

[VW20] Jennifer Wortman Vaughan und Hanna Wallach. *A human-centered agenda for intelligible machine learning*. In: *Machines We Trust: Getting Along with Artificial Intelligence* (2020).

[Wan+19] Qianwen Wang, Yao Ming, Zhihua Jin, Qiaomu Shen, Dongyu Liu, Micah J Smith, Kalyan Veeramachaneni und Huamin Qu. *Atmseer: Increasing transparency and controllability in automated machine learning*. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, S. 1–12.

[Wan+20] Maonan Wang, Kangfeng Zheng, Yanqing Yang und Xiujuan Wang. *An explainable machine learning framework for intrusion detection systems*. In: *IEEE Access* 8 (2020), S. 73127–73141.

[WB19] Daniel S Weld und Gagan Bansal. *The challenge of crafting intelligible intelligence*. In: *Communications of the ACM* 62.6 (2019), S. 70–79.

[Web+19] Christian Weber, Pascal Hirmer, Peter Reimann und Holger Schwarz. *A New Process Model for the Comprehensive Management of Machine Learning Models.* In: *ICEIS (1)*. 2019, S. 415–422.

[Wei+19a] Dennis Wei, Sanjeeb Dash, Tian Gao und Oktay Gunluk. *Generalized linear rule models*. In: *International Conference on Machine Learning*. PMLR. 2019, S. 6687–6696.

[Wei+19b] Katharina Weitz, Dominik Schiller, Ruben Schlagowski, Tobias Huber und Elisabeth André. *"Do you trust me?Ïncreasing user-trust by integrating virtual agents in explainable AI interaction design*. In: *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*. 2019, S. 7–9.

[Wel19] Adrian Weller. *Transparency: motivations and challenges*. In: *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer, 2019, S. 23–40.

[Wen+21] John Wenskovitch, Corey Fallon, Kate Miller und Aritra Dasgupta. *Beyond Visual Analytics: Human-Machine Teaming for AI-Driven Data Sensemaking*. In: *2021 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*. IEEE. 2021, S. 40–44.

[Wes+20] Carl Westin, Brian Hilburn, Clark Borst, Erik-Jan Van Kampen und Magnus Bång. *Building Transparent and Personalized AI Support in Air Traffic Control*. In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE. 2020, S. 1–8.

[Wit02] Frank Wittig. *Maschinelles Lernen Bayes' scher Netze in benutzeradaptiven Systemen*. In: (2002).

[WMR17] Sandra Wachter, Brent Mittelstadt und Chris Russell. *Counterfactual explanations without opening the black box: Automated decisions and the GDPR*. In: *Harv. JL & Tech.* 31 (2017), S. 841.

[Wol22] Arne Wolfewicz. *Deep Learning vs. Machine Learning – What's The Difference?* 21. Apr. 2022. URL: https://levity.ai/blog/difference-machine-learning-deep-learning (besucht am 23. 05. 2022).

[Wut22] Laurenz Wuttke. *Praxisleitfaden für Künstliche Intelligenz in Marketing und Vertrieb: Beispiele, Konzepte und Anwendungsfälle*. 1st ed. 2021. Springer eBook Collection. Wiesbaden: Springer Fachmedien Wiesbaden und Imprint Springer Gabler, 2022.

[WW02] Jane Webster und Richard Thomas Watson. *Analyzing the Past to Prepare for the Future: Writing a Literature Review*. In: *MIS Q.* 26 (2002).

[Xu+19] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao und Jun Zhu. *Explainable AI: A brief survey on history, research areas, approaches and challenges*. In: *CCF international conference on natural language processing and Chinese computing*. Springer. 2019, S. 563–574.

[ZC18] Jianlong Zhou und Fang Chen. *2D transparency space—bring domain users and machine learning experts together*. In: *Human and Machine Learning*. Springer, 2018, S. 3–19.

[Zd20] Julian Zucker und Myraeka d'Leeuwen. *Arbiter: A Domain-Specific Language for Ethical Machine Learning*. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, S. 421–425.

[Zer+20] Fadila Zerka, Visara Urovi, Akshayaa Vaidyanathan, Samir Barakat, Ralph TH Leijenaar, Sean Walsh, Hanif Gabrani-Juma, Benjamin Miraglio, Henry C Woodruff, Michel Dumontier et al. *Blockchain for privacy preserving and trustworthy distributed machine learning in multicentric medical imaging (C-DistriM)*. In: *Ieee Access* 8 (2020), S. 183939–183951.

[ZF14] Matthew D Zeiler und Rob Fergus. *Visualizing and understanding convolutional networks*. In: *European conference on computer vision*. Springer. 2014, S. 818–833.

[ZK19] Muhammad Rehman Zafar und Naimul Mefraz Khan. *DLIME: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems*. In: *arXiv preprint arXiv:1906.10263* (2019).

Todo list

Da der Pipeline-Begriff schon vergeben ist (siehe z.B. z.B. [KKH22] oder https://docs.microsoft.com/de-de/azure/machine-learning/concept-ml-pipelines), hier daran denken, diesen in Datenfluss umzuändern :) iv

Ein Kapitel zu Über- und Unteranpassung und Bias? 4

Später überdenken: eigenes Kapitel für Anwendungsfälle sinnvoll? 5

Diese Auswahl am Ende noch einmal überdenken/anpassen 6

Überanpassung ggf. einführen 14

tbd 16

Vlt. hier noch auf Besonderheiten bei ML eingehen? 18

Das muss ich noch machen 22

Bei direkten Zitaten Seitenzahl? Joshua fragen 24

Hier kommt noch eine Human-in-the-Loop Deinfition 25

hier Referenz zu Vialsierungskapitel 25

Dieser Satz ist noch etwas einsam, vlt. solche Aspekte rauswerfen 26

Bias/oder auch den ersten Teil des folgenden Abschnitts in mein Grundlagenkapitel mitaufnehmen? 28

Hier gehe ich noch genauer auf die Datasheets-Arten ein und zeige vlt. Beispiele . . . 29

Todo: Referenz auf Abschnitt im Kapitel der XAI-Methoden 31

Hier schreibe ich dann, wie ich die einordne, wenn ich weiß, wie ich das machen will 33

Was noch fehlt: Für welche Art von Problemen funktionieren diese Algorithmen? LIME nur für Klassifikation, aber die anderen? Vor- und Nachteile ausbauen... . . . 34

dazu muss ich noch ein Kapitel schreiben 41

Hier kommt noch eine Beschreibung der MMD-Critic-Methode 43

Hier fehlt noch, wie das Testen an sich funktioniert 46

Ich würde noch die zwei unvollständigen Kapitel fertig schreiben und dann noch die Methode DeepLift, Neural Shrubs und etwas zu Feature Interaktionen schreiben . . 46

Wenn das in Ordnung ist, würde ich hier noch einige Methoden auf hoher Abstraktionsebene beschreiben. Es gibt halt sehr viele, und ich glaube nicht dass ich jede (und schon gar nicht zu genau) beschreibene sollt. In der folgenden Auflistung siehst du ein paar von den Methoden, welche mir theoretisch fehlen würde, auch wenn dies nicht alle sind, die ich insgesamt gefunden habe. 46

Die Methoden könnten dann so oder so ähnlich beschrieben werden 47
Eventuell erkläre ich diese noch oder lasse es ganz weg 48
Ich überlege noch wie/ob ich einen Unterschied zwischen Visualisierungstools, Plattformen, Toolkits mache 50
Hier erkläre ich noch Model Cards for Model Reporting 54
Hier werde ich noch kurz erklären, was die Metriken aussagen und vlt. Bilder einbauen 54
Die beiden Checklisten gieße ich eventuell noch in den Text - auf jeden Fall fließen die Inhalte irgendwie in meine Handreichung ein, da besonders die zweite Abbildung meine dritte Forschungsfrage beantwortet 55
Hier verweise ich noch auf die Stelle (die habe ich nur noch nicht geschrieben :D) . . 58
Bitte verwenden Sie hier in jedem Fall die offizielle von der Prüfungsbehörde vorgegebene Formulierung der Selbständigkeitserklärung. 59