technische universität
dortmund

Masterarbeit

Untersuchung der Anfälligkeit perturbationsbasierter Erklärbarkeitsmethoden für Adversarial Attacks

Rahel Luise Wilking
September 2021

Gutachter:
Prof. Dr. Katharina Morik
M.Sc. Matthias Jakobs

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
http://www-ai.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die besten Ergebnisse für Aufgabenstellungen des Machine Learning erzielen meist komplexe Methoden wie Deep Neural Networks oder Ensemblemethoden [30]. Durch ihre hohe Komplexität können ihre Entscheidungen von Menschen kaum nachvollzogen werden, sie stellen eine Blackbox dar [37]. In Bereichen, die Auswirkungen auf das Leben eines Menschen haben, oder in denen Entscheidungen über große Mengen an Geld getroffen werden, wie der Medizin oder dem Finanzwesen, können falsche Entscheidungen schwerwiegende Auswirkungen haben. Für diese Felder ist es notwendig, die Entscheidungen eines Modells nachvollziehen und erklären zu können [11]. Die Ansätze im Bereich eXplainable Artificial Intelligence (XAI) werden verwendet, um das Vertrauen in Modelle zu steigern, Fehler zu verhindern, Modelle zu verbessern, aber auch um Fairness sicherzustellen [11].

Gerade wegen der kritischen Natur dieser Entscheidungen ist es wichtig, dass diese Erklärungen für das Modell wahrheitsgemäß erfolgen. Für einige der Ansätze ist es für einen Angreifer allerdings möglich die Erklärungen zu manipulieren [45]. Dadurch kann diesen Methoden nicht mehr vertraut werden und ihre Verwendung um das Vertrauen in Modelle zu steigern oder andere Ziele zu erreichen ist hinfällig. Diese Arbeit untersucht weitere Methoden auf ihre Anfälligkeit für einen solchen Angriff.

1.1 Motivation und Hintergrund

Das Feld der Explainable Artificial Intelligence ist nicht ganz neu, hat aber in den letzten Jahren viel an Aufmerksamkeit gewonnen [18, 3]. Eine beliebte Gruppe von Ansätzen sind dabei lokale modell-agnostische Ansätze [45]. Diese konstruieren Erklärungen für die Vorhersage einer Instanz und nicht das ganze Modell. Außerdem sind sie modell-agnostisch, können also für jedes beliebige Machine Learning Modell verwendet werden. Dabei werden für diese Methoden häufig Perturbationen der Daten verwendet [40].

Die Perturbationen sind dabei die Schwachstelle, die Slack et al. [45] ausnutzen, um einen Angriff auf diese Klasse an Erklärbarkeitsmethoden zu entwickeln. Durch diese Per-

turbationen entstehen Punkte, die nicht in der Verteilung der echten Daten liegen. Dies nutzt der Angriff aus [45]. Eine genaue Beschreibung des Angriffs findet sich in Abschnitt 2.3.

Slack et al. stellen in ihrer Arbeit fest, dass zwei populäre Erklärbarkeitsmethoden, LIME und SHAP (siehe Abschnitte 3.2.1 und 3.2.2) anfällig für diesen Angriff sind [45].

In dieser Arbeit werden drei weitere Methoden, Anchors [40], LORE [17] und EXPLAN [37] (siehe Abschnitte 3.2.3, 3.2.4 und 3.2.5), auf ihre Anfälligkeit für diesen Angriff untersucht. Alle drei Methoden sind lokale modell-agnostische Erklärbarkeitsmethoden, die mit Perturbationen arbeiten. Neben diesen notwendigen Kriterien für den potentiellen Erfolg des Angriffs, wurden diese Methoden auch aus dem Grund ausgewählt, dass für sie Implementierungen frei verfügbar sind. Alle Methoden haben außerdem etwas unterschiedliche Arten Perturbationen zu erzeugen, welche möglicherweise mehr oder weniger Daten außerhalb der echten Verteilung generieren, was ihre Anfälligkeit für den Angriff beeinflusst. LORE und insbesondere EXPLAN legen dabei sogar einen besonderen Fokus auf die Art und Weise, wie die Nachbarschaft eines Punktes erzeugt wird. Dies ist der Teil der Methoden, welcher die Perturbationen verursacht. Die ausgewählten Methoden geben außerdem Regeln als Ausgabe zurück, welche als leichter verständlich gelten als die linearen Modelle von LIME [40].

Neben den hier untersuchten Methoden gibt es noch weitere, welche möglicherweise auch anfällig für den Angriff von Slack et al. sind, beispielsweise der Ansatz von Baehrens et al. [4]. Aber es gibt auch Ansätze, die das Problem von Perturbationen umgehen, dafür allerdings abhängig von einer Datenmenge sind, beispielsweise LACE [33].

Der von Slack et al. vorgestellte Angriff wird auf eine Erklärbarkeitsmethode ausgerichtet [45]. Neben der Untersuchung, wie anfällig die Methoden für einen speziell auf sie ausgerichteten Angriff sind, wird außerdem getestet, wie erfolgreich der Angriff ist, falls eine Erklärbarkeitsmethode für das manipulierte Modell verwendet wird, auf die der Angriff nicht ausgerichtet ist. Dies wird für alle Kombinationen der zwei bereits von Slack et al. untersuchten und der drei hier neu untersuchten Methoden durchgeführt. Diese Daten können verwendet werden, um in einer Situation, in der ein Angriff dieser Art nicht ausgeschlossen werden kann, eine möglichst gute Wahl einer Erklärbarkeitsmethode treffen zu können. Um dies umzusetzen, wurden für Anchors, LORE und EXPLAN zugeschnittene Angriffe entwickelt, die in Abschnitt 4.2 vorgestellt werden.

1.2 Aufbau der Arbeit

Diese Arbeit besteht aus insgesamt sechs Kapiteln. Im nächsten Kapitel wird das Thema der Arbeit in seinen Kontext eingeordnet. Dabei wird auf die Felder der Explainable Artificial Intelligence und der Adversarial Attacks eingegangen. Außerdem wird in diesem Kapitel der untersuchte Angriff vorgestellt. In dem darauf folgenden Kapitel werden die Er-

klärbarkeitsmethoden vorgestellt, die in den Experimenten untersucht werden, sowie einige Grundlagen für diese. Die Experimente und die dafür konstruierten spezifischen Angriffe werden in Kapitel 4 beschrieben und in Kapitel 5 evaluiert. Abschließend wird ein Fazit gezogen und ein Ausblick gegeben.

Kapitel 2

Einordnung des Themas

In diesem Kapitel wird zunächst das Feld der Explainable Artificial Intelligence vorgestellt und darauf eingegangen, wie die Ansätze dieses Feldes kategorisiert werden können. Danach werden Adversarial Attacks und ihr Bezug zur Explainable Artificial Intelligence erklärt. Das Thema dieser Arbeit ist beiden Feldern zugehörig. Im dritten Abschnitt wird auf den in dieser Arbeit untersuchten Angriff eingegangen. Abschließend werden kurz verwandte Ansätze erwähnt.

2.1 Explainable Artificial Intelligence

Das Feld der eXplainable Artificial Intelligence (XAI), auch Interpretable Artificial Intelligence, hat in den letzten Jahren stark an Interesse gewonnen [3]. Insbesondere für sensitive Bereiche wie der Medizin und dem Finanzwesen besteht dabei eine Notwendigkeit für XAI-Methoden, aber auch in vielen anderen Feldern [11]. Eine Erklärung wird wichtiger je stärker eine Entscheidung das Leben eines Menschen beeinflusst [30]. Dabei werden Erklärbarkeitsmethoden eingesetzt, um, direkt oder indirekt, viele verschiedene Ziele zu erreichen. Ein wichtiges Ziel darunter ist es, Vertrauen in ein Modell zu steigern, aber auch Fairness sicherzustellen oder die Robustheit zu erhöhen [11].

Der Fokus der meisten Methoden liegt dabei bei dem Erklären von Ansätzen des Supervised Machine Learning. Ein Blackbox-Modell $f : X \to Y$, von einem Feature-Raum X mit n Features in einen Zielraum Y bezeichnet dabei ein Modell, dessen interne Funktionsweise entweder gänzlich unbekannt ist, oder zu komplex, um sie interpretieren zu können [18]. Im Supervised Learning wird ein solches Modell durch Training mit einer Datenmenge $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$, welche aus Tupeln von Vektoren aus dem Feature-Raum X und ihren Labeln $y^{(i)} \in Y$ besteht, gelernt [11]. Beispiele für Blackbox-Modelle sind Deep Neural Networks (DNN) und Tree Ensembles [18]. Mit g wird ein interpretierbares Modell bezeichnet, dessen Funktionsweise verstanden werden kann. Beispiele für interpretierbare

Modelle sind Entscheidungsbäume, regelbasierte Modelle und lineare Modelle, wobei diese alle nicht zu komplex werden dürfen [18].

Es besteht keine Einheitlichkeit bei der Verwendung der Begriffe Interpretierbarkeit und Erklärbarkeit, jedoch wird Interpretierbarkeit üblicherweise verwendet für das Verständnis des Modells als Ganzes, während Erklärbarkeit meist benutzt wird, wenn Erklärungen für ein Modell erzeugt werden, welches selbst nicht verständlich ist [11]. Auch bei dem Verständis davon, was eine Erklärung darstellt, gibt es unterschiedliche Auffassungen. Ein wichtiger Aspekt dabei ist, dass eine Erklärung auf die Zielgruppe angepasst sein muss, da unterschiedliches Vorwissen eine Erklärung leichter oder schwieriger verständlich machen kann [3]. Menschen bevorzugen Erklärungen, die kontrastiv sind, also das Ergebnis in einen Kontrast zu einer anderen Vorhersage setzen, sowie kurze und prägnante Erklärungen [11]. Weiterhin sind ungewöhnliche Gründe interessanter für Menschen und sie bevorzugen Erklärungen, die ihrem bisherigen Weltbild entsprechen [30]. Diese Eigenschaften sind schwierig zu evaluieren, häufig wird die Größe bzw. Länge einer Erklärung als Merkmal verwendet, es fehlen im Bereich XAI jedoch klare Metriken wie die Accuracy oder der F1-Score um die Qualität von Erklärungen zu messen [11].

Es gibt eine Vielzahl an Ansätzen, die nach verschiedenen Aspekten kategorisiert werden können. Eine erste Unterscheidung ist die zwischen ante-hoc und post-hoc Ansätzen. Ante-hoc Ansätze sind dabei solche, bei denen die Interpretierbarkeit Teil des Modells ist. Darunter fallen die oben genannten interpretierbaren Modelle, aber auch Ansätze um Modelle leichter verständlich zu machen, indem Sparsity- oder Monotonie-Beschränkungen integriert werden. Post-hoc Ansätze dagegen werden erst nach dem Trainingsprozess des Modells eingesetzt. [11]

Eine weitere Unterscheidung findet zwischen lokalen und globalen Methoden statt. Dabei beschäftigen sich globale Methoden mit der Interpretation des Modells als Ganzes bzw. auf dem gesamten Datenraum, während lokale Methoden jeweils für die Erklärung einer einzelnen Vorhersage verwendet werden. Ansätze können nur für bestimmte Machine Learning Modelle verwendbar sein, wenn sie beispielsweise strukturelle Eigenschaften ausnutzen, oder für alle Modelle, wenn sie keine Annahmen über das zu erklärende Modell treffen. Man bezeichnet diesen Unterschied als modell-spezifisch oder modell-agnostisch. Interpretierbare Modellklassen sind automatisch ante-hoc und modell-spezifisch. [11]

Diese drei Einordnungen werden meistens verwendet um Ansätze im Feld XAI zu unterscheiden. Weitere Unterteilungen können jedoch auch noch danach vorgenommen werden, ob der Ansatz Daten zur Verfügung haben muss, für welche Datentypen der Ansatz ausgelegt ist (z.B. tabellarische Daten, Text oder Bilder) und welche Form die Ausgabe hat (z.B. Visualisierung, Beispiele, interpretierbare Modelle) [11, 18, 30].

Post-hoc Erklärbarkeitsmethoden haben dabei den Vorteil, dass sie das Modell selbst nicht verändern und damit keine direkte Auswirkung auf die Vorhersagegüte des Machine Learning Modells haben. Häufig ist ein Trade-off zwischen der Vorhersagegüte und der In-

terpretierbarkeit notwendig, welche durch diese Methoden verbessert wird [3]. Der Vorteil von modell-agnostischen Ansätzen liegt in ihrer Flexibilität. Sie können nicht nur frei für schon bekannte nicht interpretierbare Modellklassen eingesetzt werden, sondern sind auch direkt anwendbar für zukünftig entwickelte Methoden [39]. Außerdem sind sie anwendbar für den Fall, dass das Modell nur in einer Form zur Verfügung steht, bei der lediglich die Vorhersagen für übergebene Eingaben zurückgegeben werden und kein Zugriff auf das Modell besteht. Die Intuition hinter lokalen Erklärbarkeitsmethoden ist, dass die Entscheidungsgrenze des Blackbox-Modells eine globale Erklärung kompliziert, während die Form der Entscheidungsgrenze in der Nähe eine spezifischen Instanz eine einfachere Struktur hat, welche die Erklärungen vereinfacht [40].

Ein Ansatz für lokale Erklärungen sind dabei lokale Surrogat-Modelle. Dabei wird ein interpretierbares Modell auf der Nachbarschaft einer zu erklärenden Instanz x gelernt [11]. Diese Nachbarschaft wird häufig durch Perturbationen erzeugt [40]. Dabei werden als Label für die Punkte der Nachbarschaft die Vorhersagen des Blackbox-Modells f verwendet [11]. Wie gut ein Surrogat-Modell sich an das Blackbox-Modell (in der Nachbarschaft von x) anpasst wird mit der Fidelity gemessen. Für diese kann die Accuracy oder der F1-Score mit Bezug auf die Blackbox-Vorhersagen berechnet werden [18].

Die in dieser Arbeit untersuchten Ansätze sind alle lokale modell-agnostische post-hoc Ansätze. Einige der Methoden verwenden außerdem lokale Surrogat-Modelle.

2.2 Adversarial Attacks

Unter Adversarial Attacks bekannt sind Angriffe auf ein Modell um Adversarial Examples zu erzeugen [22]. Ein Adversarial Example ist dabei eine Perturbation einer Eingabe, welche von einem Modell misklassifiziert wird [6]. In den letzten Jahren sind Adversarial Examples insbesondere im Bereich von Deep Neural Networks (DNN) mit Bilddaten untersucht worden. Die ersten Adversarial Examples, oder auch Evasion Attacks, für Modelle haben ihren Ursprung allerdings deutlich zuvor in der Spam und Malware Erkennung [6].

Für Bilddaten sind Adversarial Examples häufig für Menschen kaum merkbare Perturbationen von normalen Eingabebildern, die zu einer falschen Vorhersage führen. Dabei kann diese Misklassifikation gezielt oder ungezielt sein. Bei gezielter Misklassifikation wird für das Adversarial Example eine bestimmten Klasse vorhergesagt, bei ungezielter eine beliebige, die nicht mit der richtigen Klasse übereinstimmt. Diese Misklassifikationen haben üblicherweise hohe Konfidenz. [22]

Für die Angriffe werden häufig Gradienten ausgenutzt, wodurch die Angriffe auch für andere gradientenbasierte Modelle wie Support Vector Machines geeignet sind. Allerdings existieren auch Angriffe für beliebige Modelle, die nur einen Blackbox-Zugriff auf das Modell brauchen. Diese können beispielsweise auf dem Erlernen eines gradientenbasierten Surrogat-Modells basieren. Adversarial Examples können außerdem teilweise zwischen

Modellen übertragen werden, fungieren also als Adversarial Example für mehrere Modelle. Neben dem Erzeugen von Adversarial Examples, gibt es auch weitere mögliche Angriffe auf Modelle, beispielsweise Poisoning-Attacken, die das Training eines Modells angreifen. [6]

Diese Anfälligkeit von DNNs für Adversarial Examples kann große Probleme verursachen, gerade in sicherheitskritischen Anwendungskontexten wie dem autonomen Fahren. Durch das Ändern eines einzigen Pixels einer Eingabe kann eine rote Ampel als eine grüne Ampel erkannt werden [51]. Im Bereich Adversarial Defense werden daher Gegenmaßnahmen entwickelt. Beim Adversarial Training werden Adversarial Examples für das Training verwendet, um Robustheit gegen sie zu gewinnen [22]. Auch diese Ansätze sind nicht neu, ähnliche Ansätze gibt es ungefähr so lange wie Adversarial Examples [6]. Neben Ansätzen, welche direkt Adversarial Examples inkorporieren oder zu erkennen versuchen, gibt es auch allgemeine Ansätze die Robustheit von Modellen zu verbessern [22]. Die Entwicklung neuer Angriffe und neuer Verteidigungsstrategien befinden sich in einem Wettrüsten [6].

Das Feld der Adversarial Attacks hat Verbindungen zu Explainable Machine Learning. Zum einen könnten die durch XAI-Methoden gewonnen Erkenntnisse verwendet werden, um Adversarial Attacks zu entwickeln, andererseits könnten die XAI-Methoden aber auch dabei helfen Adversarial Examples zu erkennen [3]. Eine Parallele der Felder ergibt sich auch in der Verwendung von Surrogat-Modellen. Außerdem ähneln Adversarial Examples konzeptionell Counterfactual Examples [30]. Counterfactual Explanations beschreiben die Änderungen die notwendig sind, um ein anderes Klassifikationsergebnis zu erreichen, wobei diese Änderungen möglichst klein sein sollen [49]. Ein Counterfactual Example ist ein Datenpunkt der diese minimalen Änderungen erfüllt [17]. Ähnliche Techniken wie für das Finden von Adversarial Examples können auch für das Generieren von Counterfactuals verwendet werden [49].

Im Kontext des Angriffs von Slack et al. [45] geht der Angriff nicht von den Daten, sondern dem Vorhersagemodell aus. Der Angreifer ist in diesem Fall der Modellentwickler. Dies ist eine ähnliche Perspektive wie für Backdoor-Angriffe auf Modelle. Bei diesen wird ein Modell manipuliert, um bei speziellen Eingaben ein bestimmtes Verhalten zu zeigen, dass der Angreifer ausnutzen kann, wenn sich das Modell in Verwendung befindet [6]. Im Angriff von Slack et al. ist allerdings das Ziel einen in einem Modell enthaltenen Bias bei einer Untersuchung durch Erklärbarkeitsmethoden zu verstecken [45]. Dieser Angriff wird im nächsten Abschnitt genauer erläutert.

2.3 Scaffolding-Angriff

Slack et al. [45] haben einen Angriff auf perturbationsbasierte post-hoc Erklärbarkeitsmethoden entwickelt. Sie gehen dabei von der im Folgenden beschriebenen Situation aus. Es gibt einen Angreifer der ein Interesse daran hat, dass ein Modell f, welches einen Bias

enthält, eingesetzt wird. Der Angreifer muss allerdings einen Blackbox-Zugriff auf dieses Modell bereitstellen, beispielsweise für Kunden oder regulatorische Entitäten. Er geht dabei davon aus, dass diese möglicherweise post-hoc Erklärbarkeitsmethoden verwenden, um das Modell zu untersuchen. Stellen sie dabei einen Bias fest ist es unwahrscheinlich, dass das Modell eingesetzt wird bzw. werden darf. Der Angreifer möchte diesen Bias bei der Untersuchung also verstecken. Bei Anwendung des Modells soll es jedoch das ursprüngliche Verhalten beibehalten.

Mit dem Angriff von Slack et al. ist genau dies möglich. Die stattdessen angezeigten Erklärungen können sogar beliebig gewählt werden. Dabei nutzen sie aus, dass die Perturbationen, die bei den Erklärbarkeitsmethoden wie LIME und SHAP (siehe Abschnitte 3.2.1 und 3.2.2) eingesetzt werden, deutlich anders verteilt sind als die echten Daten, also oft Punkte erzeugen, die nicht der echten Datenverteilung X_{dist} entsprechen. Diese Punkte sind *out-of-distribution* (OOD). Der Angriff manipuliert die Erklärungen, indem die Rückgaben für OOD-Punkte verändert werden, wodurch die Erklärbarkeitsmethoden getäuscht werden.

Die Autoren nennen ihren Angriff *Scaffolding*. Dieser Name stammt daher, dass sie einen neuen Klassifizierer e bauen, der f als Teil enthält, aber vorher noch die Entscheidung fällt, ob f oder ein anderer Klassifizierer ψ, dessen Erklärungen angezeigt werden sollen bei einer Untersuchung, für eine Vorhersage verwendet werden soll. Sie bauen also quasi ein Gerüst um f, um den Bias von f zu verschleiern. Der neue Klassifizierer e ist wie folgt definiert:

$$e(x) = \begin{cases} f(x), & x \in X_{dist} \\ \psi(x), & \text{sonst} \end{cases} \tag{2.1}$$

Für die Umsetzung von e muss diese Fallunterscheidung durchgeführt werden können. Dafür muss ein OOD-Klassifizierer gelernt werden, der diese Entscheidung fällt. Um diesen lernen zu können, benötigt der Angriff eine Trainingsmenge X, welche aus der echten Verteilung X_{dist} stammt. Die Punkte in X werden perturbiert um eine Menge X_p zu erzeugen. Diese Perturbationen erfolgen dabei entsprechend den Perturbationen, die eine Erklärbarkeitsmethode erzeugt. Dies ist die angegriffene Methode. Den Punkten in X wird das Label *False* zugewiesen, den Punkten in X_p das Label *True*. Der OOD-Klassifizierer wird auf der Menge $X \cup X_p$ mit den entsprechenden Labels trainiert.

Für diesen OOD-Klassifizierer kann jede ausreichend genaue Methode verwendet werden. Für die Experimente von Slack et al. und auch für die Experimente in dieser Arbeit werden dafür Random Forests (siehe Abschnitt 3.1.2) verwendet. Der Klassifizierer mit Bias, f, entscheidet basierend auf einem einzigen Feature, je nach Datensatz *race* oder *gender*. Für ψ werden verschiedene Modelle verwendet. Für zwei Datensätze entscheidet ψ einmal basierend auf einem künstlich erzeugten, mit dem Feature von f unkorreliertem, binären Feature und einmal basierend auf einer XOR-Verknüpfung zwei solcher Features.

Für einen dritten Datensatz wird ein echtes Feature verwendet. Dies wird auch für die Experimente in dieser Arbeit übernommen.

Diese Auswahlen sind recht extrem, in einem echten Beispiel muss ein mit Bias behafteter Klassifizierer nicht nur allein auf einem sensitiven Feature basieren. Diese Extreme zeigen jedoch, dass der Angriff auch für diese Situation funktioniert, was Slack et al. für LIME und SHAP bestätigen können. Beide Methoden werden von dem Angriff getäuscht, dabei ist LIME etwas anfälliger als SHAP [45].

Von der Konstruktion des Angriffs aus ist es nicht notwendig, dass die angegriffene Erklärbarkeitsmethode lokal ist, allerdings ist der Evaluationsprozess von Slack et al. auf lokale Methoden ausgelegt. Diese Einschränkung wird für eine bessere Vergleichbarkeit in dieser Arbeit beibehalten. Interessant ist außerdem, dass für den Angriff für die Modelle f und ψ auch nur ein Blackbox-Zugriff vorausgesetzt wird. Diese Modelle könnten also auch von anderen Entitäten entwickelt und vom Angreifer nur verwendet werden.

2.4 Ähnliche Ansätze

Neben dem von Slack et al. entwickelten Angriff gibt es noch weitere Arbeiten, die zeigen, dass Erklärungen angegriffen werden können. Einige davon spezialisieren sich dabei auf DNNs für Bilddaten, die durch Pixel-Importance Bilder erklärt werden. Dombrowski et al. [15] zeigen, dass diese Erklärbilder beliebig manipuliert werden können, ohne dass sich der Netzwerk-Output stark ändert, indem ähnlich wie für Adversarial Examples Eingaben des Modells für einen Menschen kaum wahrnehmbar perturbiert werden. Sie zeigen dies für verschiedene Gradienten- und Propagations-basierte Erklärbarkeitsmethoden. Außerdem leiten sie eine theoretische obere Grenze für diese Art von Täuschung her und nutzen die Erkenntnisse daraus, um eine Defensivmaßnahme zu entwickeln, um die Robustheit gegen diese Art von Manipulation zu erhöhen.

Heo et al. [20] und Dimanov et al. [14] stellen Methoden vor, die einen Fine-Tuning Schritt nach dem Training eines DNN ergänzen, bei dem mit einer besonderen Loss-Funktion verschiedene Arten der Manipulation der Erklärungen umgesetzt werden. Dabei stellen Heo et al. verschiedene Arten der Manipulation der Erklärungsbilder vor, darunter gezielte und ungezielte Ansätze. Die ungezielten Ansätze dienen dabei dazu die Erklärungen unaussagekräftig zu machen, während bei ihrem gezielten Ansatz die Erklärungen für zwei Klassen getauscht werden. Dimanov et al. dagegen konzentrieren sich nicht auf Bilddaten, sondern tabellarische Daten, bei denen sie durch den veränderten Loss die Relevanz eines bestimmten Features in den Erklärungen gezielt verringern. Neben den Methoden, die auch für die Erklärungsbilder verwendet werden, zeigen sie auch eine Täuschung von LIME und SHAP. Beide Methoden zeigen ebenfalls keine starken Auswirkungen auf die Ausgaben des Modells.

Anders et al. [2] zeigen theoretisch und praktisch, dass ein Modell so manipuliert werden kann, dass die Erklärungen verschiedener Methoden beliebig geändert werden können. Die Schwachstelle liegt dabei darin, dass die Daten-Mannigfaltigkeit meist deutlich weniger dimensional ist als die Mannigfaltigkeit in die sie eingebettet ist. Da das Training eines Modells nur die Richtungen entlang der Daten-Mannigfaltigkeit festlegt, können dazu orthogonale Richtungen frei bestimmt werden. Diese dominieren in der Erzeugung der Erklärungen, wodurch die Erklärungen beliebig gewählt werden können. Sie zeigen Beispiele und Vorgehen für logistische Regression und DNNs mit Bilddaten. Außerdem stellen sie eine Defensive gegen diesen Angriff vor, bei dem die Erklärungen tangential zur Daten-Mannigfaltigkeit projiziert werden.

Abseits von den für DNNs häufig verwendeten Erklärbarkeitsmethoden, zeigen Lakkaraju und Bastani [26], dass ein Erklärbarkeitsverfahren MUSE, welches ein regelbasiertes globales Surrogat-Modell lernt, ebenfalls irreführende Erklärungen produzieren kann. Diese Erklärungen haben bei menschliche Domänen-Experten erhöhtes Vertrauen in ein auf sensitiven Features basierendes Modell verursacht. Die hier ausgenutzte Schwachstelle ist, dass sensitive Features aus anderen Features rekonstruiert werden können, sodass sie selber nicht Teil einer Erklärung sein müssen, auch wenn die Entscheidung auf ihnen beruht.

Baniecki et al. [5] hingegen zeigen, dass auch Partial Dependence falsche Erklärungen produzieren kann. Hierbei nutzen sie einen Data Poisoning Angriff auf die Trainingsmenge, über welche die Partial Dependence berechnet werden soll. Dabei stellen sie zwei Ansätze vor. Der erste Ansatz basiert auf einem genetischen Algorithmus und benötigt lediglich Blackbox-Zugriff auf das zu erklärende Modell, ist also modell-agnostisch. Der zweite Ansatz ist Gradienten-basiert, also nur für Modelle mit differenzierbaren Ausgaben, wie DNNs geeignet. Sie stellen sowohl einen gezielten Angriff mit gewünschter Erklärung vor, als auch eine Robustheitsprüfung, die eine maximal andere Erklärung erzeugt. Ihr Ansatz lässt sich dabei auch auf andere globale post-hoc Erklärbarkeitsverfahren verallgemeinern, solange sie auf einem Datensatz basieren.

Die hier kurz vorgestellten Ansätze greifen verschiedene Erklärungen auf unterschiedliche Weisen an. Am ähnlichsten zu dem Angriff von Slack et al. sind dabei die Ansätze, die eine Manipulation des Modells vorsehen, ohne die Daten oder die Erklärbarkeitsmethode zu verändern. Dies sind die Angriffe von Heo et al. [20], Dimanov et al. [14] und Anders et al. [2], welche den Fokus jedoch alle auf DNNs und die dafür häufig verwendeten Erklärbarkeitsmethoden legen. Dimanov et al. zeigen jedoch auch eine Täuschung von LIME und SHAP in diesem Kontext.

Kapitel 3

Methoden

Für die Experimente dieser Arbeit werden fünf verschiedene perturbationsbasierte Erklärbarkeitsmethoden aus dem Feld XAI verwendet. Die ersten beiden, LIME und KernelSHAP wurden dabei bereits von Slack et al. [45] auf den in Abschnitt 2.3 vorgestellten Angriff untersucht und dienen zum Vergleich. Die noch nicht untersuchten Methoden Anchors, LORE und EXPLAN werden im Weiteren vorgestellt. Zunächst wird jedoch auf einige Methoden des Machine Learning eingegangen, die als Grundlagen für diese benötigt werden. Random Forests als die für die OOD-Klassifizierer verwendeten Modelle werden dort ebenfalls vorgestellt.

3.1 Basismethoden

Dieser Abschnitt behandelt drei Methoden aus dem Machine Learning, welche als Komponenten der Erklärbarkeitsmethoden bzw. des Angriffs verwendet werden. Dabei handelt es sich zunächst um Entscheidungsbäume, die sowohl für LORE (Abschnitt 3.2.4), als auch EXPLAN (Abschnitt 3.2.5) verwendet werden. Außerdem, darauf aufbauend, werden Random Forests vorgestellt, welche als Modell für die in Abschnitt 2.3 erwähnten OOD-Klassifizierer verwendet werden und auch in EXPLAN genutzt werden. Abschließend wird Agglomerative Clustering erklärt, welches ebenfalls für EXPLAN verwendet wird.

3.1.1 Entscheidungsbäume

Ein Entscheidungsbaum ist ein Vorhersagemodell, welches eine Baumstruktur aufweist. Die Vorhersage beginnt dabei immer in der Wurzel des Baums. Jeder Knoten, der kein Blatt ist, enthält eine Prüfung, von dessen Ergebnis der weitere Verlauf im Baum abhängt. Für jedes Prüfungsergebnis existiert ein Pfad zu einem Kindknoten. Eine vorherzusagende Instanz wandert jeweils weiter zu dem Kindknoten, welcher dem Prüfungsergebnis der Instanz entspricht. Handelt es sich bei dem Kindknoten um einen weiteren Prüfungsknoten, so wird

dieser Prozess fortgesetzt. Wird ein Blatt erreicht, wird mithilfe dessen eine Vorhersage für die Instanz getroffen. [35]

Häufig bestehen die Prüfungen an den Prüfungsknoten für kontinuierliche Features aus dem Test, ob $x_i \leq a$ für einen Index i eines kontinuierlichen Features und einen Schwellwert a. Für diskrete Features sind Kindknoten für alle möglichen Werte des Features möglich, es kann aber auch Prüfungen der $x_i \in S$, für S eine Teilmenge von allen möglichen Feature-Werten vom Feature i, geben. Mit dieser Art von Prüfungen unterteilen Entscheidungsbäume den Instanzraum in Hyperrechtecke. [35]

Ein Entscheidungsbaum kann sowohl für Klassifikations-, als auch für Regressionaufgaben verwendet werden. Nach dem Lernen des Baums besteht der primäre Unterschied zwischen Klassifikations- und Regressionsbäumen darin, was die Blätter enthalten. Für Klassifikationsbäume sind die Blätter einer Klasse zugeordnet, möglicherweise enthalten sie auch eine Klassenverteilung oder ein einfaches Vorhersagemodell [35, 19]. Für Regressionsbäume enthalten die Blätter konstante reelle Zahlen [10]. Im Weiteren werden Klassifikationsbäume im Fokus stehen.

Entscheidungsbäume werden schrittweise rekursiv aufgebaut. Dabei gibt es diverse Algorithmen, darunter CART [10] und C4.5 [35]. Algorithmen unterscheiden sich in folgenden Punkten:

- Welche Prüfungen sind möglich?
- Wie wird gemessen wie gut eine Aufteilung ist?
- Was enthalten die Blätter?
- Wann ist ein Knoten ein Blatt?
- Wie wird Pruning durchgeführt?

Zum ersten Punkt wurden übliche Prüfungen oben bereits erwähnt, jedoch stellen Breiman et al. beispielsweise auch die Verwendung von beliebigen Hyperebenen als Trennlinien für Aufteilungen vor, sowie das Nutzen von eingeschränkten Booleschen Ausdrücken über mehrere Variablen [10]. Auch der dritte Punkt wurde bereits erwähnt.

Die Konstruktion des Baums startet mit dem Wurzelknoten und einer Menge an Trainingsinstanzen. Für einen Knoten muss zuerst geprüft werden, ob er die Voraussetzungen erfüllt ein Blatt zu sein. Knoten, die nur Instanzen einer einzigen Klasse enthalten, also rein sind, werden ein Blatt der entsprechenden Klasse. Ist ein Knoten nicht rein, können die Daten noch weiter aufgeteilt werden. Es ist möglich auch einen unreinen Knoten zu einem Blatt zu erklären, wenn beispielsweise ein Mindestmaß an Reinheit erreicht wurde, oder eine maximale Tiefe für den Baum festgelegt wurde. Soll der Knoten weiter aufgeteilt werden, kommen mehrere Prüfungen in Frage. Unter den Kandidaten soll die beste Aufteilung

gefunden und ausgewählt werden. Intuitiv ist dabei das Ziel, dass die Klassenverteilungen in den Kindknoten reiner werden sollen als im aktuellen Knoten. [10, 35]

Für das Berechnen der Güte einer Aufteilung gibt es verschiedene Kriterien. Dazu zählt das *gini*-Kriterium, welches von CART präferiert wird [10]

$$gini(T) = \sum_{i \neq j} p(C_j|T)p(C_i|T) \tag{3.1}$$

$$= \left(\sum_j p(C_j|T) \right)^2 - \sum_j p^2(C_j|T) \tag{3.2}$$

$$= 1 - \sum_j p^2(C_j|T) \tag{3.3}$$

sowie der *information gain*

$$gain(X) = info(T) - info_X(T) \tag{3.4}$$

wobei

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot info(T_i), \tag{3.5}$$

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} \cdot \log_2 \left(\frac{freq(C_j, S)}{|S|} \right), \tag{3.6}$$

und die von C4.5 verwendete *gain ratio* [35]

$$gain\ ratio(X) = gain(X)/split\ info(X) \tag{3.7}$$

mit

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \log_2 \left(\frac{|T_i|}{|T|} \right). \tag{3.8}$$

Dabei ist $p(C_j|T)$ die Wahrscheinlichkeit, dass im Knoten T eine Instanz der Klasse C_j angehört, welche empirisch mit $freq(C_j, S)/|S|$ geschätzt werden kann. S ist die Menge der Trainingsinstanzen am Knoten T, $|S|$ die Anzahl dieser und $freq(C_j, S)$ gibt an, wie viele Instanzen der Klasse C_j in S auftreten. X bezeichnet eine Aufteilung der Daten in T auf die Kindknoten T_i. Es gibt neben diesen weitere mögliche Kriterien, beispielsweise das Twoing-Kriterium [10]. Es wird jeweils die Prüfung und Aufteilung gewählt, welche das ausgewählte Kriterium maximiert.

Ein Vorteil von Entscheidungsbäumen ist ihre Interpretierbarkeit. Diese gilt allerdings nur für relativ kleine Bäume [30]. Werden Knoten spät, oder erst wenn sie rein sind, zu Blättern, so können jedoch schnell große Bäume entstehen. Diese sind außerdem instabil und können sich stark ändern, bei nur kleinen Veränderungen in den Trainingsdaten. Für die bessere Interpretierbarkeit und zur Stärkung der Robustheit und Verallgemeinerungsstärke von Bäumen können diese nach ihrem Aufbau daher zurückgeschnitten, also

verkleinert werden. Diesen Prozess nennt man Pruning. Pruning kann auf verschiedene Weisen passieren, Breiman et al. stellen beispielsweise das cost-complexity-pruning vor [10].

Ein weiterer Vorteil von Entscheidungsbäumen ist, dass sie häufig Möglichkeiten haben mit fehlenden Feature-Werten umzugehen. Im Fall von CART geschieht dies über sogenannte Surrogat-Variablen [10] und in C4.5 über eine Gewichtung und Weiterleitung an alle Kindknoten, sowie einer Zusammenfassung aller erreichten Blätter [35].

Für LORE und EXPLAN wird in der praktischen Umsetzung YaDT (Yet another Decision Tree builder) [41] verwendet. YaDT ist eine effiziente C++ Implementation eines stark an C4.5 angelehnten Entscheidungsbaum Algorithmus. Der Algorithmus basiert also wie C4.5 auch auf dem *gain ratio*-Kriterium. Die Software wird vom Autor frei zur Verfügung gestellt[1].

3.1.2 Random Forests

Ein Forest beschreibt ein Ensemble von Bäumen. Ein Random Forest basiert dabei in irgendeiner Weise auf einem Zufallsvektor. Dieser Zufall kann auf verschiedene Arten eingebracht werden. Beispielsweise kann jeder Entscheidungsbaum mit einer Datenmenge trainiert werden, welche zufällig mit Zurücklegen aus den Trainingsdaten gezogen wird, also einem Bootstrap Sample. Andere Möglichkeiten sind die verwendeten Variablen für die Prüfungen an den Knoten zufällig zu wählen, oder zufälliges Rauschen in die Label zu inkorporieren. [9]

Breiman stellt die folgende allgemeine Definition eines Random Forests auf.

Definition 3.1.1. Ein Random Forest ist ein Klassifizierer bestehend aus einer Gruppe von Baum-artigen Klassifizierern $\{h(x, \Theta_k), k = 1, \ldots\}$ wobei die $\{\Theta_k\}$ unabhängig identisch verteilte Zufallsvektoren sind und jeder Baum eine Stimme für die beliebteste Klasse für die Eingabe x abgibt. [9]

Die Definition umfasst nur Random Forests zur Klassifikation, Random Forests können jedoch auch für Regressionsaufgaben verwendet werden, die Vorhersage des Random Forest ist dann der Mittelwert der Baumvorhersagen [9].

Breiman stellt zwei Varianten von Random Forests vor, die nicht nur zufällige Trainingsdatensätze nutzen, sondern zusätzlich zwei verschiedene Arten von auf Zufall basierenden Knotenprüfungen verwenden. Er baut dabei auf das vorher von ihm entwickelte Bagging auf, welches bereits Bootstrap Samples der Trainingsdaten verwendet um jeden Baum zu trainieren [8].

Beide Varianten ziehen wie bei Bagging für jeden Baum ein neues Bootstrap Sample der gleichen Größe wie der ursprüngliche Trainingsdatensatz. Bei der ersten Variante,

[1] http://pages.di.unipi.it/ruggieri/software.html

Forest-RI, werden zusätzlich an jedem Knoten F Features zufällig ausgewählt, die für die Aufteilung an diesem Knoten in Frage kommen. Unter diesen wird die beste Aufteilung gewählt. Bei der zweiten Variante, Forest-RC, werden F zufällige Linearkombinationen von L zufällig ausgewählten Features mit Koeffizienten zufällig gleichverteilt aus $[-1, 1]$ ausgewählt und die beste Aufteilung mit diesen gesucht. Beide Varianten folgen für den Aufbau der Bäume CART [10] (siehe auch Abschnitt 3.1.1), jedoch ohne Pruning. [9]

Durch die Verwendung von Bootstrap Samples für die Bäume ist für jeden Baum etwa ein Drittel der ursprünglichen Trainingsdaten nicht verwendet worden, die sogenannten out-of-bag (OOB) Instanzen. Diese können verwendet werden um Schätzungen nahe derer eines Testdatensatzes zu erhalten. Random Forests nach Breiman berechnen dabei jeweils OOB-Schätzungen für die Generalisierungsfehlerrate, also die Fehlerrate auf ungesehenen Instanzen, die Stärke der Entscheidungsbaum-Klassifizierer, sowie ihrer Korrelation. Außerdem kann eine Feature Importance berechnet werden, die auf dem Anstieg des Vorhersagefehlers bei Permutation der Feature-Werte basiert. [9]

Eine Aussage die Breiman trifft ist, dass Random Forests nicht für Overfitting anfällig sind [9]. Dies basiert er auf dem Beweis, dass die Generalisierungsfehlerrate von Random Forests für eine zunehmende Anzahl an Bäumen zu einem festen Wert konvergiert. Sie sind also nicht für den typischen Verlauf anfällig, dass ab einem gewissen Punkt der Generalisierungsfehler wieder beginnt anzusteigen. Hastie et al. üben jedoch Kritik an dieser zu generellen Aussage, da dieser Wert zu dem Random Forests konvergieren durchaus die Daten overfitten kann [19].

Breiman zeigt außerdem eine obere Grenze für diesen Generalisierungsfehler. Diese hängt ab von der Stärke der einzelnen Klassifizierer des Random Forest und der Korrelation zwischen ihnen. Das Ziel ist es also die Korrelation gering zu halten, während die Stärke möglichst hoch ist. [9]

Diese Maße können teilweise kontrolliert werden, durch die Wahl der Anzahl an Features F bei Forest-RI. In Breimans Experimenten nahm die Korrelation zu für ein größeres F. Die Stärke nahm für kleine F zu, blieb aber danach relativ konstant. Die Wahl von $F = 1$ war in den Experimenten nicht viel schlechter als die Wahl eines höheren F. [9]

Die konkret beste Wahl von F ist abhängig vom verwendeten Datensatz und sollte mit optimiert werden [19]. Allerdings erreichen Random Forests ähnlich gute Ergebnisse wie Adaboost bei deutlich geringerem Aufwand und Laufzeit, insbesondere für Datensätze mit sehr vielen Variablen [9]. Hastie et al. erhalten jedoch bessere Ergebnisse für Boosting Modelle als mit Random Forests [19].

Der Großteil der Experimente für Random Forests wurden von Breiman auf Forest-RI durchgeführt. Dies ist auch die Variante von Random Forests[2], die in Scikit-learn [34] umgesetzt ist und für die Experimente in dieser Arbeit verwendet wird.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

3.1.3 Agglomerative Clustering

Agglomerative Clustering ist eine Unterform des hierarchischen Clusterings. Bei hierarchischem Clustering muss anders als für beispielsweise k-Means, keine Anzahl von gewünschten Clustern vorgegeben werden. Stattdessen wird eine hierarchische Struktur von Clustern erzeugt, wobei auf der obersten Stufe ein einzelnes Cluster mit allen Datenpunkten steht und auf der untersten Stufe N Cluster mit jeweils einem einzelnen Datenpunkt. Die Hierarchie hat insgesamt N Stufen aus denen sich $N-1$ unterschiedliche Partitionen der Daten ablesen lassen. Die Stufen unterscheiden sich dabei jeweils darin, dass die nächst höhere Stufe genau ein Cluster hat, welches die Vereinigung zweier Cluster der aktuellen Stufe darstellt. Alle anderen Cluster bleiben gleich. Diese hierarchische Struktur lässt sich auf zwei grundlegende Arten konstruieren. Zum einen mit dem agglomerativen (bottom-up) Ansatz und zum anderen mit dem divisiven (top-down) Ansatz. Beim agglomerativen Ansatz wird mit jeweils einer Instanz pro Cluster begonnen und in jedem Schritt zwei Cluster verschmolzen. Bei dem divisiven Ansatz wird mit einem Cluster aller Datenpunkte begonnen und in jedem Schritt ein Cluster in zwei Teile partitioniert. Beide Ansätze werden wiederholt bis das jeweils andere Extrem erreicht ist. [19]

Hier liegt der Fokus auf dem ersten Ansatz, dem Agglomerative Clustering. Das grobe Vorgehen des Verfahrens wurde bereits beschrieben, die verschiedenen Ansätze im Bereich Agglomerative Clustering unterscheiden sich jedoch darin, wie die Entscheidung, welche zwei Cluster in jedem Schritt miteinander verschmolzen werden sollen, getroffen wird.

Dabei ist die grobe Idee, in jedem Schritt die beiden Cluster zu wählen, die sich am meisten ähneln. Zu diesem Zweck werden Unähnlichkeitsmaße zwischen Clustern definiert. Diese basieren üblicherweise auf Unähnlichkeitsmaßen zwischen Datenpunkten. Für ein Unähnlichkeitsmaß $d : I \times I \to \mathbb{R}^+$ gilt für alle Instanzen $i, j, k \in I$:

$$d(i,j) \geq 0, \tag{3.9}$$

$$d(i,j) = 0 \iff i = j, \tag{3.10}$$

$$d(i,j) = d(j,i). \tag{3.11}$$

Im Allgemeinen gilt nicht die Dreiecksungleichung $d(i,j) \leq d(i,k) + d(k,j)$, mit welcher das Unähnlichkeitsmaß zu einer Distanz wird. [19, 32]

Beispiele für Unähnlichkeitsmaßen über Cluster A, B, wobei $d(i,i')$ ein Unähnlichkeitsmaß auf Instanzen ist, sind [19]:

- Single Linkage, wobei die Clusterunähnlichkeit die kleinste Unähnlichkeit zwischen Punkten der beiden Cluster ist

$$d_{SL}(A,B) = \min_{\substack{i \in A \\ i' \in B}} d(i,i'), \tag{3.12}$$

- Complete Linkage, wobei die Clusterunähnlichkeit die größte Unähnlichkeit zwischen Punkten der beiden Cluster ist

$$d_{CL}(A,B) = \max_{\substack{i\in A \\ i'\in B}} d(i,i'), \tag{3.13}$$

- Group Average, wobei die Clusterunähnlichkeit die durchschnittliche Unähnlichkeit zwischen Punkten der beiden Cluster ist

$$d_{GA}(A,B) = \frac{1}{|A||B|}\sum_{i\in A}\sum_{i'\in B} d(i,i'), \tag{3.14}$$

  wobei $|A|$ bzw. $|B|$ jeweils die Anzahl der Punkte im jeweiligen Cluster bezeichnet.

Neben diesen gibt es noch viele weitere Methoden Clusterunähnlichkeiten zu berechnen. In jedem Schritt des Verfahrens wird über die ausgewählte Clusterunähnlichkeit berechnet, welche beiden Cluster sich am ähnlichsten sind. Diese beiden Cluster A und B werden daraufhin zu einem Cluster C verschmolzen. Für den nächsten Schritt müssen nur die Unähnlichkeiten des neuen Clusters C zu allen anderen Clustern H neu bestimmt werden. Clusterunähnlichkeiten zwischen Clustern die nicht A oder B sind bleiben unverändert. [31]

Lance und Williams [27] haben für eine Teilmenge der Clusterunähnlichkeitsmaße eine allgemeine Update-Gleichung aufgestellt:

$$d(C,H) = \alpha_A d(A,H) + \alpha_B d(B,H) + \beta d(A,B) + \gamma|d(A,H) - d(B,H)|. \tag{3.15}$$

Diese charakterisiert für bestimmte Werte von α_A, α_B, β und γ die oben vorgestellten Clusterunähnlichkeitsmethoden. Dabei müssen die Koeffizienten wie folgt gewählt werden [31]:

- Single Linkage: $\alpha_A = \alpha_B = 1/2$, $\beta = 0$ und $\gamma = -1/2$,
- Complete Linkage: $\alpha_A = \alpha_B = 1/2$, $\beta = 0$ und $\gamma = 1/2$,
- Group Average: $\alpha_A = \frac{|A|}{|A|+|B|}$, $\alpha_B = \frac{|B|}{|A|+|B|}$ und $\beta = \gamma = 0$.

Neben den bereits genannten Methoden kann mit der Update-Gleichung von Lance und Williams auch das Kriterium von Ward [50] berechnet werden, welches in jedem Schritt die Verschmelzung wählt, welches die Summe der quadrierten Fehler minimal erhöht, bzw. die minimale Veränderung der Varianz verursacht. Dabei müssen in der obigen Formulierung der Update-Gleichung die initialen Unähnlichkeiten zwischen Punkten die quadrierte euklidische Distanz sein. [32]

Die Koeffizienten für das Ward-Kriterium sind dann [13]:

$$\alpha_A = \frac{|A|+|H|}{|A|+|B|+|H|},\ \alpha_B = \frac{|B|+|H|}{|A|+|B|+|H|},\ \beta = \frac{-|H|}{|A|+|B|+|H|} \text{ und } \gamma = 0. \tag{3.16}$$

Diese Unähnlichkeitsmethoden erfüllen die Reduzierbarkeitseigenschaft, falls

$$d(C, H) \geq \min\{d(A, H), d(B, H)\}, \tag{3.17}$$

für alle $H \neq A, B, C$ [13]. Alle hier erwähnten Methoden erfüllen diese Eigenschaft [31]. Diese Eigenschaft ermöglicht es, die aus dem hierarchischen Clustering resultierende Baumstruktur grafisch darzustellen, da sie sicherstellt, dass eine später erfolgende Clusterverschmelzung nicht bei einer niedrigeren Unähnlichkeit erfolgt als vorherige.

Diese Baumstruktur ergibt sich daraus, dass jeweils eine Verbindung zwischen den Clustern, die verschmolzen werden, und dem Cluster, zu dem sie verschmolzen werden, existiert. Die Baumwurzel ist dabei das Cluster mit allen Datenpunkten und die Blätter die Cluster mit jeweils einer Instanz. Dieser Baum kann grafisch dargestellt werden, indem die Clusterknoten jeweils auf der Höhe der Unähnlichkeit der beiden Cluster aus dem es entstanden ist gezeichnet werden. Die Blätter werden auf Höhe der Null gezeichnet. Diese Darstellung nennt sich Dendrogramm. [19].

Ein Dendrogramm kann auch dazu verwendet werden aus dem hierarchischen Clustering ein flaches Clustering zu bestimmen. Dazu kann eine horizontale Linie gewünschter Höhe durch das Dendrogramm gezogen werden. Jede Linie, die dadurch geschnitten wird, definiert ein Cluster des resultierenden flachen Clusterings. Die Höhe der Linie beschreibt dabei den maximalen Wert der Unreinheit bis zu dem das agglomerative Clustering durchgeführt werden müsste um das flache Clustering zu erhalten. [19]

Um ein Clustering mit einer bestimmten Anzahl an Clustern zu erhalten, kann der entsprechende Schritt der Hierarchie extrahiert werden.

3.2 Erklärbarkeitsmethoden

Die nun vorgestellten fünf Erklärbarkeitsmethoden sind alles lokale modell-agnostische Methoden, die auf Perturbationen basieren und in den letzten fünf Jahren entwickelt wurden. Die Ansätze werden dabei in chronologischer Reihenfolge ihrer jeweiligen Veröffentlichung vorgestellt. Die ersten beiden behandelten Methoden, LIME und KernelSHAP, wurden dabei bereits von Slack et al. [45] verwendet, um ihren Angriff zu testen, und werden in dieser Arbeit als Vergleich genutzt. Zusätzlich zu diesen werden Anchors, LORE und EXPLAN untersucht.

3.2.1 LIME

Local Interpretable Model-agnostic Explanations (LIME) ist eine von Ribeiro et al. [39] entwickelte Erklärbarkeitsmethode. Wie am Namen zu erkennen ist, handelt es sich um ein lokales und modell-agnostisches Verfahren. Das Vorgehen besteht dabei darin, ein interpretierbares Modell zu lernen, welches eine lokale Annäherung des zu erklärenden Modells

darstellt. Das Ziel ist dabei eine möglichst gute Annäherung zu erreichen, ohne dass das Erklärungsmodell zu komplex wird. Einen besonderen Fokus legen Ribeiro et al. darauf, dass der Repräsentationsraum der Erklärungen interpretierbar sein muss, auch wenn das zu erklärende Modell eine andere Repräsentation verwendet [39]. Im Folgenden vereinfachen wir dies jedoch zum Raum der Datenpunkte, da wir in dieser Arbeit nur tabellarische Daten verwenden, ohne komplexe Vorverarbeitung, die eine Interpretation der einzelnen Features erschweren würde. Diese Unterscheidung ist jedoch relevant für Text- und Bilddaten, für welche die Methode ebenfalls eingesetzt werden kann.

Sei f ein Blackbox-Modell, G eine Klasse interpretierbarer Modelle, beispielsweise lineare Modelle oder Entscheidungsbäume, und $g \in G$ ein Erklärungsmodell für f. Sei x die zu erklärende Instanz und π_x ein Näherungsmaß, welches den Abstand eines Punktes zu x misst. Die Komplexität des Erklärungsmodell wird mit $\Omega(g)$ bezeichnet.

Das LIME-Erklärungsmodell ergibt sich als Ergebnis der folgenden Optimierung [39]:

$$\xi(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g). \tag{3.18}$$

Dabei ist L eine Loss-Funktion, welche angibt wie gut g sich an f in der Umgebung von x anpasst. Diese Umgebung nennen wir Z. Dann ist dieser Loss definiert wie folgt [39]:

$$L(f, g, \pi_x) = \sum_{z \in Z} [f(z) - g(z)]^2 \, \pi_x(z). \tag{3.19}$$

Das Näherungsmaß wird also verwendet, damit Punkte die sehr nah an x liegen stärker gewichtet werden. Dies soll sicherstellen, dass das Erklärungsmodell in der Nähe von x mit dem zu erklärenden Modell übereinstimmt.

Diese allgemeine Formulierung lässt verschiedene Klassen von interpretierbaren Modellen, Messungen der Komplexität der Erklärungsmodelle und Näherungsmaße zu. In ihrer Arbeit und auch im dazu entwickelten Code[3] beschränken sich Ribeiro et al. jedoch auf lineare Modelle mit wenigen verwendeten Features als interpretierbare Modelle, deren Komplexität anhand der Anzahl dieser Features gemessen wird. Ein solches lineares Modell $g(z) = w \cdot z$, hat einen Gewichtsvektor w, der nur wenige von Null verschiedene Einträge hat. Die zu diesen Einträgen korrespondierenden Features sind diejenigen, die von dem Modell verwendet werden. Der Bias-Term des linearen Modells wird hier in die Vektorschreibweise inkorporiert indem ein konstantes Feature ergänzt wird. Für das Näherungsmaß verwenden sie einen exponentiellen Kernel, $\pi_x(z) = exp(-d(x, z)^2/\sigma^2)$, wobei σ die Kernel-Breite ist und d ein Distanzfunktion auf den Daten [39]. Für tabellarische Daten ist diese Distanzfunktion dabei die euklidische Distanz.

Die Nachbarschaft für die Berechnung der Loss-Funktion wird durch Perturbationen von x erzeugt. Die Autoren gehen auf die Perturbation im Fall von tabellarischen Daten nicht ein, allerdings ist aus dem Code ersichtlich, dass für kontinuierliche Features ein

[3] https://github.com/marcotcr/lime/tree/0.1.1.36

normalverteilter Zufallswert, mit Erwartungswert und Varianz der Trainingsdaten, gezogen wird und für diskrete Features ein Feature-Wert zufällig, der Verteilung in den Trainingsdaten entsprechend, gewählt wird. Dabei ist insbesondere relevant, ob das kategoriale Feature danach mit x übereinstimmt oder nicht.

Ribeiro et al. beschreiben die Verwendung von K-Lasso zur Optimierung des Ausdrucks 3.18. Dabei verwenden sie Lasso [19], um die K Features auszuwählen, welche für das lineare Modell verwendet werden sollen. Die Gewichte werden danach nach der Kleinste Quadrate Methode [19] bestimmt. Das K wird dabei fest gewählt und nicht mit optimiert. [39]

Der Code wurde seit seiner Publikation weiter aktualisiert und diese Art der Optimierung ist in der verwendeten Version zwar noch im Code vorhanden, die Default-Einstellungen benutzen jedoch bei einem $K <= 6$ Forward Selection [19] zur Auswahl und bei einem $K > 6$ die Features mit den höchsten Produkten aus Gewichten und Feature-Werten von x, bei einem Modell mit allen Features.

Das Ergebnis von LIME ist dann das resultierende lineare Modell, bestehend aus den dafür verwendeten Features und ihren Gewichten.

LIME lässt sich neben seiner Verwendung als lokales Erklärbarkeitsmodell auch mithilfe des Submodular Pick Verfahren zu einer globalen Erklärbarkeitsmethode erweitern [39]. Dabei werden basierend auf einer submodularen Funktion eine kleine Menge an möglichst repräsentativen Datenpunkten ausgewählt und ihre einzelnen lokalen Erklärungen zusammengestellt, um zu einem Verständnis des zu erklärenden Modells auf dem gesamten Definitionsbereich beizutragen.

3.2.2 KernelSHAP

SHapley Additive exPlanations (SHAP) ist eine Erklärbarkeitmethode, welche sechs andere zuvor entwickelte Methoden in einem gemeinsamen Rahmen vereint [29]. Lundberg und Lee zeigen dabei, dass diese Methoden, darunter auch das in Abschnitt 3.2.1 vorgestellte LIME, sich alle als eine Additive Feature Attribution Methode darstellen lassen. Diese charakterisieren sie dabei wie in der folgenden Definition.

Definition 3.2.1. Additive Feature Attribution Methoden haben ein Erklärungsmodell, welches eine lineare Funktion binärer Variablen ist:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{3.20}$$

mit $z' \in \{0,1\}^M$, M die Anzahl der vereinfachten Features und $\phi_i \in \mathbb{R}$ [29]

Es handelt sich dabei wie bei LIME auch um lokale Erklärungen eines Blackbox-Modells f durch ein Erklärungsmodell g. Dieses arbeitet dabei in einem vereinfachten binären Feature-Raum. Durch eine Instanz-spezifische Mapping Funktion h_x lässt sich aus der vereinfachten Instanz x' wieder die originale Instanz x rekonstruieren, also $x = h_x(x')$.

Lundberg und Lee zeigen, dass die Klasse der Additiven Feature Attribution Methoden eine eindeutige Lösung hat welche drei wünschenswerte Eigenschaften erfüllt. Diese drei Eigenschaften sind dabei wie folgt [29]:

1. **Local Accuracy**

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{3.21}$$

   Das Erklärungsmodell $g(x')$ stimmt mit dem originalen Modell $f(x)$ überein bei $x = h_x(x')$.

2. **Missingness**

$$x'_i = 0 \Longrightarrow \phi_i = 0 \tag{3.22}$$

   Missingness beschränkt Features mit korrespondierendem $x'_i = 0$ darauf keinen Einfluss zugeordnet zu bekommen.

3. **Consistency** Sei $f_x(z') = f(h_x(z')$ und $z' \setminus i$ bezeichne $z'_i = 0$ zu setzen. Für zwei beliebige Modelle f und f' gilt, falls

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \tag{3.23}$$

   für alle $z' \in \{0,1\}^M$, dann ist $\phi_i(f', x) \geq \phi_i(f, x)$

Das einzige Erklärungsmodell einer Additive Feature Attribution Methode, welches diese Eigenschaften erfüllt, ist nach Theorem 1 von Lundberg und Lee das mit folgenden Gewichten [29]:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} \left[f_x(z') - f_x(z' \setminus i \right], \tag{3.24}$$

wobei $|z'|$ die Anzahl der nicht-null Einträge von z' ist, und $z' \subseteq x'$ die Vektoren z' bezeichnet, dessen nicht-null Einträge eine Teilmenge der nicht-null Einträge von x' sind.

Diese Werte sind aus dem Bereich der Spieltheorie auch als Shapley Werte [44] bekannt. Dort beschreiben sie eine faire Verteilung des Gewinns eines Spiel auf die Spieler. Die Eigenschaften, mit denen Shapley Werte dort definiert werden, lassen sich aus den hier vorgestellten Eigenschaften Local Accuracy und Consistency ableiten [30].

SHAP Werte sind Shapley Werte für die bedingte Erwartungsfunktion des originalen Modells, also die Lösung für Gleichung 3.24 für $f_x(z') = f(h_x(z')) = E\left[f(z)|z_S\right]$, wobei S die Menge der Indexe der nicht-null Elemente von z' beschreibt. Dabei ist das Mapping $h_x(z') = z_S$ implizit, wobei z_S bei Indexen nicht in S fehlende Werte hat. Um auch Modelle erklären zu können, die nicht mit fehlenden Werten umgehen können, wird $f(z_S)$ mit $E\left[f(z)|z_S\right]$ angenähert.

Lundberg und Lee stellen mehrere Methoden vor, um SHAP Werte zu berechnen, darunter fällt auch KernelSHAP, welches an LIME angelehnt ist. Dabei werden die Features als unabhängig angenommen, wodurch sich die Schätzung des Erwartungswertes vereinfacht. Sei $\bar{S}$ die Komplementmenge von S, dann vereinfacht es sich zu:

$$f(h_x(z')) = E\left[f(z)|z_S\right] = E_{z_{\bar{S}}|z_S}\left[f(z)\right] \approx E_{z_{\bar{S}}}\left[f(z)\right]. \tag{3.25}$$

KernelSHAP nutzt die Formulierung des Optimierungsproblem in LIME wie in Gleichung 3.18, wählt jedoch Ω, $\pi_{x'}$ und $L(f, g, \pi_{x'})$ so, dass die Lösung genau die Shapley Werte sind. Dafür müssen sie wie folgt gewählt werden [29]:

$$\begin{aligned}
\Omega(g) &= 0, \\
\pi_{x'}(z') &= \frac{M-1}{\binom{M}{|z'|}|z'|(M-|z'|)}, \\
L(f, g, \pi_{x'}) &= \sum_{z' \in Z}\left[f(h_x(z')) - g(z')\right]^2 \pi_{x'}(z')
\end{aligned}$$

Dabei gilt $\pi_{x'}(z') = \infty$ für $|z'| \in \{0, M\}$, wodurch $f_x(\emptyset) = \phi_0$ und $f(x) = \sum_{i=0}^{M} \phi_i$ sichergestellt wird.

Die Lösung kann wie bei LIME mit linearer Regression berechnet werden. Die Gewichte müssen jedoch anders interpretiert werden. Eine Python-Implementation von KernelSHAP wurde ebenfalls bereitgestellt[4].

Perturbationen entstehen hier beim Schätzen der $f_x(z')$. Insbesondere durch die möglicherweise nicht zutreffende Annahme der Feature-Unabhängigkeit kann es zu unrealistischen Datenpunkten kommen [30].

3.2.3 Anchors

Anchors ist eine Erklärbarkeitsmethode, die wie LIME (siehe Abschnitt 3.2.1) von Ribeiro et al. [40] entwickelt wurde. Auch bei Anchors handelt es sich um eine lokale, modellagnostische Methode, jedoch lernt sie nicht wie bei LIME lokal lineare Modelle, sondern erzeugt hochpräzise Entscheidungsregeln. Regeln sind für Menschen leichter zu verstehen als lineare Entscheidungsgrenzen. Außerdem haben die Autoren festgestellt, dass es im Fall von linearen Modellen zu Unklarheit über den Geltungsbereich der lokalen Erklärung kommen kann. In ihren Experimenten waren Menschen besser in der Lage zu entscheiden, ob das Modell der lokalen Erklärung für ein neues Beispiel verwendet werden kann, wenn es sich dabei um eine Regel, also einen Anchor handelt, als um ein lineares Modell wie bei LIME. Zusätzlich sind Anchors auch in der Lage sich nicht-linearen Entscheidungsgrenzen anzupassen. Wie auch LIME können Anchors für verschiedene Datentypen und unterschiedliche Arten von Vorhersagen verwendet werden. [40]

[4] https://github.com/slundberg/shap/tree/0.32.1

Ein Anchor ist dabei eine Regel, die die Vorhersage einer Instanz verankert. Das bedeutet, dass die Instanz die Regel erfüllt und Veränderungen von Features, die nicht dafür sorgen, dass die Bedingungen der Regel verletzt werden, mit hoher Wahrscheinlichkeit, keine Änderung der Vorhersage hervorrufen. Die hohe Präzision der Regeln ist dabei Teil der Definition. Sei A eine Entscheidungsregel. $A(x) = 1$ gilt, wenn x alle Prädikate der Regel erfüllt. $\mathcal{D}$ sei die Perturbationsverteilung für x und $\mathcal{D}(\cdot|A)$ die bedingte Verteilung für Punkte, die die Prädikate von A erfüllen. A ist ein Anchor falls gilt:

$$E_{\mathcal{D}(z|A)}\left[\mathbf{1}_{f(x)=f(z)}\right] \geq \tau,\ A(x) = 1, \tag{3.26}$$

wobei $\mathbf{1}_{f(x)=f(z)}$ den Wert 1 annimmt, falls $f(x) = f(z)$ und 0 sonst. Die Verteilung der Perturbationen (und Erklärungen) muss dabei wieder, wie bereits bei LIME, interpretierbar sein. [40]

Dies ist dabei eine Kondition über die Präzision der Regel, da gilt [40]

$$\text{prec}(A) = E_{\mathcal{D}(z|A)}\left[\mathbf{1}_{f(x)=f(z)}\right]. \tag{3.27}$$

Da diese genau zu berechnen jedoch komplex ist, wird die Definition eines Anchors gelockert, indem die hohe Präzision nur mit hoher Wahrscheinlichkeit erfüllt sein muss [40]:

$$P(\text{prec}(A) \geq \tau) \geq 1 - \delta. \tag{3.28}$$

Neben der Präzision kann auch die Abdeckung einer Regel bestimmt werden, also welchen Anteil des Eingaberaums eine Regel umfasst. Die Abdeckung (Coverage) ist definiert als [40]:

$$\text{cov}(A) = E_{\mathcal{D}(z)}\left[A(z)\right]. \tag{3.29}$$

Erfüllen mehrere Regeln die Bedingungen an einen Anchor, so wird diejenige mit der höchsten Abdeckung bevorzugt. Damit ergibt sich das kombinatorische Optimierungsproblem [40]:

$$\max_{A \text{ s.t. } P(\text{prec}(A)\geq\tau)\geq 1-\delta} \text{cov}(A). \tag{3.30}$$

Eine Lösung für dieses Problem existiert immer, da die Regel, die x in allen Features genau definiert, immer eine Präzision von 1 hat, allerdings mit einer sehr geringen Abdeckung [40].

Für eine effiziente Lösung dieses Optimierungsproblems stellen die Autoren einen Algorithmus vor, der auf einem Algorithmus für ein pure-exploration multi-armed bandit Problem, KL-LUCB, basiert. Dieser wird dabei kombiniert mit Beam Search eingesetzt. Es wird mit einer leeren Regel begonnen. In jeder Iteration werden zunächst Kandidaten aus der Menge der besten Kandidaten der letzten Iteration generiert. Dabei wird jeder Regel aus dieser Menge ein weiteres Prädikat hinzugefügt, welches nicht bereits Teil der Regel war. Für diese neuen Kandidaten wird daraufhin mit KL-LUCB ihre Präzision geschätzt.

Dabei werden durch den Algorithmus nur so viele perturbierte Punkte aus der Verteilung $\mathcal{D}(\cdot|A)$ gezogen, wie notwendig sind, um die Schätzung mit ausreichender Sicherheit durchzuführen. Möglicherweise werden weitere Punkte benötigt um sicher zu stellen, ob eine Regel den Grenzwert τ erreicht, oder nicht. Aus den Kandidaten werden jeweils die B besten ausgewählt und für die nächste Iteration verwendet. Zusätzlich wird die Abdeckung der Regeln berechnet. Da bei der Hinzunahme weiterer Prädikate die Abdeckung einer Regel nicht steigen kann, wird dies ebenfalls verwendet um den Suchraum einzuschränken. Wurde bereits ein zulässiger Anchor mit einer höheren Abdeckung gefunden, so muss eine Regel nicht weiter verfolgt werden. Am Ende des Algorithmus wird der Anchor mit der höchsten Abdeckung zurückgegeben. Dieser Algorithmus ist dabei eine Erweiterung eines einfachen Greedy-Ansatzes darum B Regeln gleichzeitig zu verwalten, um nicht von schrittweise optimalen, aber insgesamt suboptimalen Entscheidungen abhängig zu sein. Außerdem wurde die explizite Berücksichtigung der Abdeckung ergänzt. [40]

Für den Fall von tabellarischen Daten verwenden die Autoren einen Validierungsdatensatz als $\mathcal{D}$. Um Beispiele aus $\mathcal{D}(\cdot|A)$ zu erzeugen, werden die Features, die in A vorkommen, auf Werte festgesetzt, die A erfüllen, und die restlichen Features zusammenhängend von einer Zeile aus $\mathcal{D}$ gezogen. [40]

Die Laufzeit des Algorithmus zum Bestimmen von Anchors ist stark abhängig von der Anzahl an möglichen Prädikaten, welche die maximale Regellänge und Iterationszahl bedingt [30]. Dies sorgt für hohe Laufzeiten auf Datensätzen mit vielen Features. Außerdem kann es auf stark unbalancierten Datensätzen und in der Nähe von Entscheidungsgrenzen teilweise zu sehr langen Regeln kommen, welche wenig hilfreich für einen Anwender sind [40].

Der von den Autoren veröffentlichte Code[5] enthält neben der Rückgabe des gefundenen Anchors und seiner Präzision und Abdeckung auch die Option partielle Regeln und ihre jeweiligen Werte zu extrahieren. Damit kann die Konstruktion des Anchors nachverfolgt werden und potentiell kann auch nachträglich die Präzisionsgrenze erweitert werden, um so eine kürzere Regel zu erhalten.

3.2.4 LORE

Die von Guidotti et al. [17] entwickelten LOcal Rule-based Explanations (LORE) sind eine weitere lokale modell-agnostische Erklärbarkeitsmethode, welche auf Perturbationen basiert. Die Perturbationen treten dabei in der durch einen genetischen Algorithmus erzeugten Nachbarschaft einer zu erklärenden Instanz x auf. Auf der generierten Nachbarschaft wird daraufhin ein Entscheidungsbaum (siehe Abschnitt 3.1.1) als lokales Modell gelernt. Schließlich wird aus dem Baum eine Entscheidungsregel extrahiert, sowie eine Menge an Counterfactual Rules. Wie bereits bei Anchors, verwenden die Autoren Regeln wegen ihrer

[5] https://github.com/marcotcr/anchor

leichteren Verständlichkeit für Anwender. Einen weiteren Vorteil sehen sie in der Ergänzung der Erklärung um Counterfactuals. Für die Entscheidungsbäume als lokales Modell entscheiden sie sich, da es relativ einfach ist aus ihnen sowohl eine Entscheidungsregel, als auch Counterfactual Rules zu bestimmen. [17]

Durch die Verwendung eines genetischen Algorithmus zur Erzeugung einer Nachbarschaft um die Instanz x, erreichen die Autoren eine balancierte Trainingsmenge für das lokale Modell, welche um die Entscheidungsgrenze um Instanz x herum eine höhere Dichte aufweist. Um eine balancierte Nachbarschaft zu erreichen, durchlaufen sie den genetischen Algorithmus zweimal, um zwei Datensätze $Z_=$ und $Z_\neq$ zu erzeugen. Sie beschränken sich auf binäre Klassifikationen. Dabei beschreibt $Z_=$ die Instanzen der Nachbarschaft, welche von dem Blackbox-Modell f, welches erklärt werden soll, mit der gleichen Klasse vorhergesagt werden wie die Instanz x und $Z_\neq$ umfasst die Instanzen, die zur anderen Klasse zugeordnet werden. Dabei unterscheiden sich die Durchläufe des genetischen Algorithmus in der Wahl der $fitness$-Funktion, welche bewertet wie gut eine Instanz zur Vorgabe passt und wesentlich bestimmt ob eine Instanz Teil der nächsten Generation wird und ob sie am Ende für das Ergebnis des Algorithmus ausgewählt wird. Die verwendeten $fitness$-Funktionen sind wie folgt definiert:

$$fitness^x_=(z) = \mathbf{1}_{f(x)=f(z)} + (1 - d(x,z)) - \mathbf{1}_{x=z} \tag{3.31}$$

$$fitness^x_\neq(z) = \mathbf{1}_{f(x)\neq f(z)} + (1 - d(x,z)) - \mathbf{1}_{x=z}. \tag{3.32}$$

Dabei beschreibt $d(x,z)$ eine Distanzfunktion auf den Daten. Guidotti et al. verwenden dafür ein gewichtetes Mittel aus dem Simple Matching Coefficient für kategoriale Features und einer normalisierten euklidischen Distanz für kontinuierliche Features. Sie konnten in Experimenten jedoch keine starken Unterschiede bei der Verwendung anderer Distanzen finden. Die $fitness$-Funktion begünstigt Datenpunkte der gewünschten Klasse, welche möglichst nahe an x liegen, entsprechend der gewählten Distanz, aber nicht genau x sind. [17]

Der genetische Algorithmus beginnt mit einer Population P_0 von N Kopien von x. Für diese werden initial einmal die $fitness$-Werte berechnet. Danach beginnt die Evolutionsschleife. Diese beginnt mit der Auswahl der besten Exemplare, das heißt diejenigen mit den höchsten $fitness$-Werten, für die Population P_{i+1}. Der nächste Schritt ist das $Crossover$, mit der Crossover-Wahrscheinlichkeit pc. Hier verwenden Guidotti et al. ein two-point-crossover, wobei jeweils zwei Elterninstanzen und zwei Features zufällig ausgewählt werden, deren Werte zwischen diesen Instanzen jeweils gewechselt werden. Dieser Schritt resultiert in der Population P'_{i+1}. Diese werden entsprechend der Mutationswahrscheinlichkeit pm mutiert, was bedeutet, dass einige Feature-Werte durch zufällig aus der Verteilung der Daten gezogene Werte ersetzt werden. Die Verteilung der Daten nähern sie dabei durch Datenpunkte aus einem übergebenen Datensatz an. Der Schleifendurchlauf schließt ab mit einer weiteren Berechnung der $fitness$-Werte der abschließenden Population P''_{i+1}. Nach

G Schleifendurchläufen endet der Algorithmus und die gefundenen Datenpunkte werden zurückgegeben. [17]

Dieser Algorithmus wird zweimal durchgeführt, jeweils mit einer der beiden *fitness*-Funktionen, und die Ergebnisse werden zusammengefasst zur Nachbarschaft $Z = Z_{=} \cup Z_{\neq}$. Die Parameter des genetischen Algorithmus, N, pc, mc und G werden dabei von Guidotti et al. festgelegt und in der bereitgestellten Python-Implementierung[6] dem Anwender von LORE nicht direkt als Parameter zur Verfügung gestellt. Die Autoren sehen in diesem Parameter-losen Ansatz einen Vorteil gegenüber LIME und Anchors, wo beispielsweise die Anzahl Features K oder die gewünschte Präzision τ festgelegt werden muss (siehe Abschnitte 3.2.1 und 3.2.3). [17]

Permutationen der Daten entstehen während des genetischen Algorithmus bei den Crossover- und Mutations-Operationen. Durch diese können Zusammenhänge in den Daten zerstört werden und so auch unrealistische Datenpunkte entstehen. Auf der erzeugten Nachbarschaft wird ein Entscheidungsbaum mit YaDT gelernt, genaueres dazu findet sich in Abschnitt 3.1.1.

Die Erklärungen, die LORE erzeugt, bestehen aus einem Tupel $\langle r = p \rightarrow y, \Phi \rangle$. Dabei beschreibt $r = p \rightarrow y$ eine Entscheidungsregel. Die Konsequenz der Regel, y, stimmt dabei mit der Klasse überein, die das lokale Modell, der Entscheidungsbaum g, für die Instanz x vorhersagt, also $y = g(x)$. Die Prämisse der Regel, p, besteht dabei aus einer Konjunktion von Prädikaten, wie sie für Prüfungen im Entscheidungsbaum verwendet werden. Eine Regel r ist konsistent mit dem lokalen Modell g, falls $g(z) = y$ für alle Instanzen z, die alle Prädikate der Prämisse erfüllen. Die Regel und das Modell stimmen also auf diesen Instanzen überein. Für eine Menge an Prädikaten δ definiere $p\,[\delta]$ als die Prädikate in p ergänzt um δ, wobei Prädikate über Features in p, für die es auch Prädikate in δ gibt von diesen überschrieben werden. Eine Counterfactual Rule ist dann eine Regel $p\,[\delta] \rightarrow \hat{y}$, wobei $\hat{y} \neq y$ ist. δ heißt dann ein Counterfactual. Die Menge Φ der Erklärungen ist eine Menge an Counterfactual Rules. [17]

Diese Erklärungen werden aus dem berechneten Entscheidungsbaum g erzeugt. Die Entscheidungsregel r ergibt sich dabei als Konjunktion der Prüfungen an den Knoten des Entscheidungsbaums auf dem Weg von der Wurzel zu dem Blatt, in welches die Instanz x eingeordnet wird. Die Konsequenz der Regel ist dabei die Klasse des Blatts. Die resultierende Regel ist konsistent mit g. Für die Counterfactual Regeln werden die Pfade von der Wurzel zu Blättern der gegenteiligen Klasse ebenfalls zu Regeln konstruiert. Dadurch ergeben sich direkt die Counterfactual Rules, die Counterfactuals δ müssen nicht explizit berechnet werden. Auch diese Regeln sind konsistent mit g. Unter diesen Regeln werden diejenigen ausgewählt, welche minimal viele Prädikate enthalten, die nicht von x erfüllt werden. Diese bilden die Menge Φ. Neben den Counterfactual Rules berechnet LORE zusätzlich aus diesen und der Instanz x Counterfactual Examples. [17]

[6] https://github.com/riccotti/LORE

Die finale Ausgabe von LORE enthält neben der Regel und den Counterfactuals auch den Baum aus denen diese gewonnen wurden. Für die in dieser Arbeit durchgeführten Experimente wird dieser verwendet, um daraus ein Feature-Ranking zu erzeugen. Genaueres dazu in Abschnitt 5.1.1.

3.2.5 EXPLAN

Der Hauptfokus der Erklärbarkeitsmethode EXPLaining black-box classifiers using Adaptive Neighborhood generation (EXPLAN) [37] liegt darin eine gute Nachbarschaft einer Instanz zu erzeugen, auf der ein lokales Modell gelernt wird. Eine gute Nachbarschaft ist nach Rasouli und Yu dabei kompakt, repräsentativ, divers und balanciert. Es handelt sich bei EXPLAN ebenfalls um einen lokalen und modell-agnostischen Ansatz. Nach der Erzeugung der Nachbarschaft wird wie bei LORE (siehe Abschnitt 3.2.4) ebenfalls ein Entscheidungsbaum als lokales Modell erlernt und anschließend auf die gleiche Weise eine lokale Entscheidungsregel aus dem Baum extrahiert. [37]

Die Qualität der Nachbarschaft hat einen starken Einfluss auf die Güte der Erklärungen und insbesondere die Anpassungsgüte des lokalen Modells, bzw. die Fidelity [37]. Rasouli und Yu haben bereits vor der Entwicklung von EXPLAN mit einer Methode zur Erzeugung von Nachbarschaften, in Kombination mit linearer Regression bzw. Entscheidungsbäumen als lokale Modelle, Verbesserungen gegenüber LIME erzielen können [36].

Diese Methode bildet den ersten Schritt von EXPLAN, die Erzeugung einer dichten Datenmenge. In einem zweiten Schritt wird aus dieser eine repräsentative Datenmenge gebildet, welche anschließend balanciert wird. Abschließend wird dann ein lokales interpretierbares Modell gelernt. [37]

Der erste Schritt, die Erzeugung einer dichten Datenmenge, lässt sich weiter in Schritte unterteilen. Zunächst wird eine Menge von N perturbierten Datenpunkten zufällig erzeugt. Dies geschieht entsprechend einer übergebenen Datenverteilung $\mathcal{D}$. Bei Betrachtung des entwickelten Python-Codes[7] ist erkennbar, dass hierfür alle Features als kategorial angenommen werden und die Häufigkeitsverteilungen der Werte in einem übergebenen Datensatz verwendet werden um für jedes Feature unabhängig voneinander zufällige Feature-Werte zu bestimmen. Der so entstandene Datensatz wird mit S bezeichnet. Auf der Menge $(S, f(S))$, also diesen Datenpunkten und den Vorhersagen des Blackbox-Klassifizierers f auf ihnen als Label, wird daraufhin ein Random Forest (siehe Abschnitt 3.1.2) gelernt. Dieser wird verwendet, um mit der TreeInterpreter-Methode [42] lokale Feature Importance Vektoren pro Instanz $s \in S$ zu berechnen.

Die TreeInterpreter-Methode betrachtet dabei die Änderung der Knotenvorhersage (falls dieser Knoten ein Blatt wäre), also im Fall der Klassifikation die Klassenwahrscheinlichkeiten, zwischen einem Entscheidungsknoten und seinem Kindknoten auf dem

[7] https://github.com/peymanrasouli/EXPLAN

Vorhersage-Pfad einer Instanz als Beitrag des Features, auf dem die Prüfung in diesem Entscheidungsknoten basiert, zur Vorhersage. Die Knotenvorhersage des Wurzelknoten ist dabei der Bias. Falls ein Feature mehrfach auf dem Pfad vorkommt, werden die Beiträge miteinander verrechnet. Die Vorhersage der Instanz ergibt sich aus der Addition dieser Beiträge und dem Bias. Für einen Random Forest werden die Beiträge und der Bias pro Baum bestimmt und über alle Bäume der Durchschnitt gebildet. [42]

Die berechneten Feature Importances $\mathcal{V}$ werden im letzten Schritt verwendet, um die Datenpunkte in S an x anzunähern. Dabei wird für jedes $s \in S$ für Feature Werte $s_j \neq x_j$ geprüft, ob die Instanz-spezifischen Beiträge des Features j für die Klassen von s und x (nur eine Klasse wenn sie übereinstimmen, sonst zwei) annähernd gleich sind. Dieser Vergleich findet auf einer diskretisierten Version von $\mathcal{V}$ statt, auch der Test, ob die Feature-Werte bereits übereinstimmen findet diskretisiert statt. Ist diese Ähnlichkeit gegeben, so wird der Feature-Wert s_j durch x_j ersetzt. Es ergibt sich schließlich eine dichte Datenmenge Z um x. [37]

Die Idee dahinter ist eine Betrachtung der Struktur von Daten nicht nur im Feature-Raum, sondern auch im Vorhersageraum. Durch den Vergleich der Feature Importance Werte von Instanzen werden solche Änderungen gemacht, die die Punkte im Datenraum näher an x bringen, ohne dabei einen großen Einfluss auf ihr Verhältnis im Vorhersageraum zu haben. [36]

Nachdem durch diese Schritte eine dichte Datenmenge um x erzeugt wurde, wird im nächsten Schritt Agglomerative Clustering (siehe Abschnitt 3.1.3) eingesetzt, um eine repräsentative Datenmenge als Untermenge dessen zu erhalten. Dabei wird ein Grenzwert τ für die minimale Menge an Punkten pro Klasse festgelegt. Die Punkte in Z werden entsprechend ihrer Vorhersagen von f in Gruppen G_l pro Label l aufgeteilt. Außerdem wird jeder Gruppe eine Kopie von x hinzugefügt. Für jede Gruppe G_l wird daraufhin Agglomerative Clustering verwendet und daraus eine Aufteilung in zwei Cluster erlangt. Enthält das Cluster C_x welches die Kopie von x enthält noch mindestens τ viele Punkte, so wird $G_l = C_x$ gesetzt und der Clustering-Vorgang wiederholt. Andernfalls wird G_l nicht weiter unterteilt. Ist dies für jedes Label geschehen, werden die resultierenden Gruppen G_l wieder zu einem Datensatz Z' zusammengefasst. Für das Agglomerative Clustering wird jeweils das Ward-Kriterium verwendet. [37]

Diese Menge Z' kann in ihrer Klassenverteilung über f noch unbalanciert sein. Daher wird in einem abschließenden Schritt SMOTE [12] eingesetzt um die Klassen zu balancieren. SMOTE führt dabei Oversampling durch, indem neue Instanzen auf den Verbindungslinien zwischen einer Instanz der gewünschten Klasse und einem seiner nächsten Nachbarn erzeugt werden. Dieser Schritt resultiert in der finalen Nachbarschaft $\mathcal{X}$. [37]

Auf diesem Datensatz wird wie bereits erwähnt ein Entscheidungsbaum als lokales Modell erlernt. EXPLAN verwendet dabei wie LORE auch YaDT um diesen zu erzeugen. Auch wie bei LORE wird aus diesem Baum eine Entscheidungsregel extrahiert, die den

Vorhersagepfad von x im Baum beschreibt. Die Ausgabe von EXPLAN ist diese Regel, sowie der lokale Entscheidungsbaum. [37]

Anders als bei LORE werden bei EXPLAN keine Counterfactual Rules bestimmt, obwohl dies aufgrund des Entscheidungsbaums als lokalem Modell auf die gleiche Weise auch möglich wäre. Permutationen treten bei EXPLAN bereits im ersten Schritt der Nachbarschaftsgenerierung auf. Da diese außerdem auch Interaktionen von Features vernachlässigt, ist es auch hier möglich, dass unrealistische Datenpunkte entstehen. Die weiteren Schritte in der Nachbarschaftserzeugung könnten dies jedoch auch abmildern oder sogar korrigieren.

Kapitel 4

Experimente

Die Experimente dieser Arbeit folgen in Teilen der Struktur von Slack et al. [45]. Der Code für die Experimente wurde in Python geschrieben und basiert auf den Repositories von Slack et al.[1] und Rasouli und Yu[2]. Letzteres umfasst dabei auch den notwendigen Code um Anchors und LORE anzuwenden.

Die verwendeten Datensätze werden im nächsten Abschnitt vorgestellt und entsprechen dabei denen, die bereits für Evaluation von Slack et al. [45] verwendet wurden. Darauf folgt die Beschreibung der Angriffe auf die Erklärbarkeitsmethoden, insbesondere der neu entwickelten Angriffe auf Anchors, LORE und EXPLAN. Dabei wird auch auf die Parameteroptimierung eingegangen. Abschließend werden die durchgeführten Experimente beschrieben.

Für das Training der Adversarial Models für LORE und EXPLAN wird dabei Multiprocessing eingesetzt, um die Laufzeiten zu verkürzen. Bei der Erzeugung der Erklärungen von Anchors, LORE und EXPLAN wurde ebenfalls Multiprocessing eingesetzt. Dies gilt bei den Mix-Versuchen auch für die Erklärungen von LIME und KernelSHAP.

4.1 Datensätze

Für die Experimente dieser Arbeit werden die drei Datensätze verwendet, die auch Slack et al. [45] bereits für ihre Evaluation genutzt haben. Dabei handelt es sich um den **COMPAS** [24] Datensatz von ProPublica, sowie den **Communities and Crime** (CC) [38] Datensatz und eine vorverarbeitete Version des **German Credit** [21, 47] Datensatzes aus dem UCI Machine Learning Repository [16].

Der COMPAS Datensatz enthält Informationen über die Kriminalgeschichte, Gefängniszeiten und demografische Informationen, sowie ihren COMPAS risk score für Angeklagte aus Broward County, Florida. Nach Vorverarbeitung, um fehlende Werte zu ent-

[1] https://github.com/dylan-slack/Fooling-LIME-SHAP
[2] https://github.com/peymanrasouli/EXPLAN

fernen, enthält der Datensatz noch 6172 Datenpunkte. Für die Experimente werden die sechs Variablen `age`, `two_year_recid`, `c_charge_degree`, `race`, `sex`, `priors_count` und `length_of_stay` verwendet. Dabei wird letztere aus der Beginn- und Endzeit des Aufenthalts berechnet. Die Zielvariable ist binär, ob das Risiko als *Hoch* eingestuft wurde. Die sensitive Variable ist hier *race*, wobei diese ebenfalls binarisiert wurde und nur zwischen *African-American* und anderen unterschieden wird. Abweichend von der Datenvorverarbeitung von Slack et al. werden die binären Variablen `c_charge_degree` und `sex` nur in numerische Variablen umgewandelt. Ursprünglich wurde für jeweils ein Niveau eine Indikatorvariable erzeugt.

Der Communities and Crime Datensatz enthält neben sozio-ökonomische Informationen auch Informationen über die Kriminalität und Strafverfolgungsdaten für Gemeinden der USA. Die gewählte Zielvariable ist die Rate an Gewaltverbrechen pro 100.000 Einwohner. Diese wird dabei binarisiert indem der Median als Grenzwert verwendet wird. Alle Datenpunkte, für die dieser Wert fehlt, sowie alle Spalten mit fehlenden Werten werden entfernt. Dies resultiert in 1994 Datenpunkten. Zusätzlich entfernt werden einige Spalten, die den Namen, oder einen Code für die Gemeinde, den Bezirk oder den Staat enthalten. Dies resultiert in 100 Variablen. Das sensitive Feature ist hier `racePctWhite numeric`.

Der German Credit Datensatz enthält demografische und finanzielle Informationen über 1000 Kreditantragssteller. Das sensitive Attribut ist `Gender` und die Zielvariable `GoodCustomer`. Die Variable `PurposeOfLoan` wird nicht verwendet, insgesamt gibt es nach der Vorverarbeitung 28 Variablen.

Den ersten beiden Datensätzen werden außerdem jeweils zwei binäre Variablen mit den Bezeichnungen `unrelated_column_one` und `unrelated_column_two` hinzugefügt, deren Werte zufällig gewählt werden. Sie sind also unkorreliert mit den sensitiven Attributen. Für den dritten Datensatz wird das Feature `LoanRateAsPercentOfIncome` verwendet, weshalb diese Features nicht hinzugefügt werden. Da dieses Feature kontinuierlich ist, wird die binäre Klassifikation anhand dessen, ob der Wert oberhalb oder unterhalb seines Mittelwerts in den Trainingsdaten liegt, ausgeführt. Alle Datensätze werden skaliert und es werden 10% pro Datensatz als Testdatensatz verwendet, der Rest als Trainingsdatensatz.

4.2 Adversarial Models

Der in Abschnitt 2.3 beschriebene Angriff besteht darin ein Adversarial Model zu bestimmen. Dieses umfasst dabei insbesondere das Lernen des OOD-Klassifizierers. Als Eingabe benötigt ein Adversarial Model das Blackbox-Modell f und das Modell ψ welches bei den Erklärungen angezeigt werden soll. Für das Training muss außerdem ein Datensatz übergeben werden. Neben diesen Parametern erhält jedes der Adversarial Models einen Parameter `perturbation_multiplier` und Parameter bezüglich des zu lernenden Random Forest. Für die Einzel- und Mix-Versuche werden Random Forests von Scikit-learn [34] mit 100 Bäumen

und Default-Einstellungen genutzt. Alle Adversarial Models, bis auf das für KernelSHAP, erhalten zusätzlich eine Liste der kategorialen Features. Außerdem kann das Adversarial Model zusätzliche Parameter erhalten die für die konkrete Perturbationsbestimmung der Methoden benötigt werden.

Der grobe Ablauf zum Training des Adversarial Models ist wie folgt. Die übergebenen Trainingsdaten werden `perturbation_multiplier` häufig repliziert, dies geschieht, um die Anzahlen gegenüber den erzeugten Perturbationen auszugleichen. Die Perturbationen werden entsprechend der angegriffenen Methode erzeugt, dabei ist üblicherweise der Parameter `perturbation_multiplier` involviert, um die Anzahl zu kontrollieren. Den Daten werden anschließend neue Label zugewiesen, dabei werden die echten Datenpunkte der Klasse 1 zugeordnet und die Perturbationen der Klasse 0. Hierbei wird außerdem meistens überprüft, ob die Perturbationen Punkte erzeugt haben, die bereits Teil der echten Trainingsdaten waren. Diese werden dann mit dem Label 1 versehen. Der so konstruierte Datensatz wird in einen Trainings- und Testdatensatz aufgeteilt. Der Test-Anteil beträgt dabei 20%. Mit dem neuen Trainingsdatensatz wird der Random Forest als OOD-Klassifizierer trainiert. Der Testdatensatz wird verwendet, um für die Sensitivitätsexperimente die Güte des OOD-Klassifizierers zu bestimmen.

Die Adversarial Models für LIME und KernelSHAP werden für die Experimente unverändert verwendet. LIME erhält als zusätzliche Parameter die Standardabweichung der Normalverteilung mit Mittelwert 0 und Standardabweichung 0.3, aus welcher zufällige Werte auf die Datenpunkte im übergebenen Datensatz addiert werden. Dies wird bei LIME mit einem `perturbation_multiplier` von 30, mehrfach wiederholt. Dabei werden für LIME nur Perturbationen über kontinuierliche Features erzeugt und ausgewertet, mit rein kategorialen Datensätzen kann der Angriff aktuell nicht umgehen. Zu diesem Zweck erhält das Adversarial Model für LIME die Liste der diskreten Features. Es ist außerdem das einzige Modell, welches bei der Zuweisung der neuen Label keine Überprüfung enthält, ob die Perturbationen echte Datenpunkte replizieren. Das Adversarial Model für KernelSHAP erhält als zusätzliche Parameter eine Background-Distribution, bzw. eine Anzahl k für k-Means-Clustering, um diese zu erzeugen, sowie eine Anzahl an Perturbationen. Diese Perturbationen werden durch iterierte zufällige Ersetzungen von Feature-Werten durch zufällige Werte aus der Background-Distribution erzeugt. Der Parameter `perturbation_multiplier` wird hier nur für die Replikation der Trainingsdaten verwendet. Das Adversarial Model für KernelSHAP verwendet als einziges keine Liste der kategorialen Features.

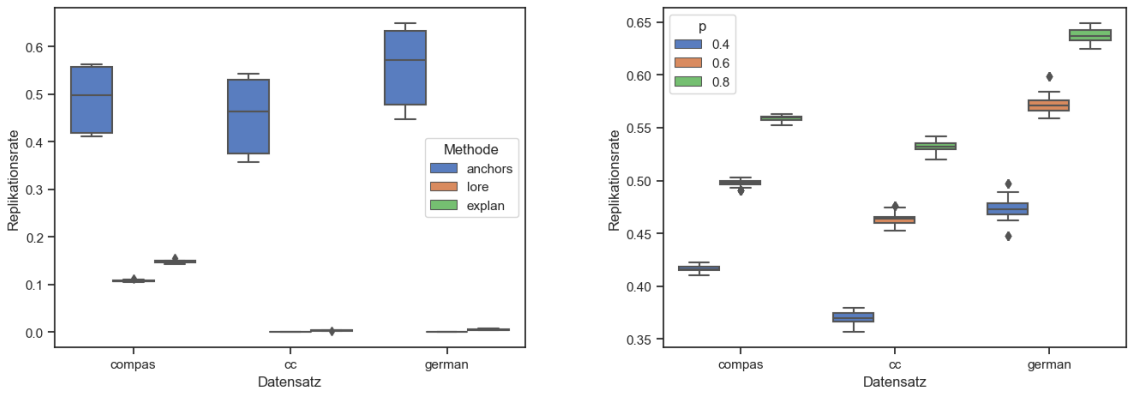
Die trainierten Adversarial Models, bzw. ihre Klassifikationsfunktion werden dann den Erklärbarkeitsmethoden als Blackbox übergeben. Diese Klassifikationsfunktion enthält den Schritt zunächst den gelernten OOD-Klassifizierer anzuwenden, und entsprechend dem erhaltenen Ergebnis die Vorhersage von f oder ψ zurückzugeben. In den nächsten Abschnitten werden die konstruierten Angriffe für Anchors, LORE und EXPLAN beschrieben.

4.2.1 Anchors

Die zusätzlichen Eingabeparameter für das Adversarial Model für Anchors bestehen aus `p` und `n_samples_per_tuple`. Die Perturbationen von Anchors entstehen bei dem Samplen von Daten aus einer entsprechend eines Anchor-Kandidaten bedingten Verteilung. Diese Anchor-Kandidaten können als Tupel von Konditionen angesehen werden. Die Erzeugung von Perturbationen zum Trainieren des OOD-Klassifizierers, konstruiert pro Punkt der übergebenen Datenmenge eine Anzahl solcher Tupel und zieht Daten aus der entsprechenden bedingten Verteilung. Die Anzahl dieser Tupel wird als aufgerundeter Quotient der Parameter `perturbation_multiplier` und `n_samples_per_tuple` festgelegt. Pro Tupel werden genau `n_samples_per_tuple` viele Punkte aus der bedingten Verteilung gezogen. Damit ergeben sich etwa $n \cdot$ `perturbation_multiplier` viele Perturbationspunkte, wobei n die Anzahl der Datenpunkte in der übergebenen Menge bezeichnet. Diese Tupel können theoretisch jede Länge zwischen 1 und m, der Gesamtanzahl an möglichen Konditionen, haben. Dabei werden diese Tupel im Prozess von Anchors inkrementell aufgebaut. Da der Suchraum von Anchors im Verlauf des Algorithmus jedoch eingegrenzt wird, werden tendenziell für kürzere Tupel häufiger Daten aus der Verteilung gezogen und damit diese Art von Perturbationen erzeugt. Daher werden die Längen der Tupel aus einer geometrischen Verteilung bestimmt. Der Parameter `p` dieser Verteilung kann variiert werden. Mit dieser vorgegebenen Länge wird ohne zurücklegen eine Menge an möglichen Konditionen gezogen, um das Tupel zu konstruieren.

Ein zusätzlicher Schritt, der nur für Anchors verwendet wird ist ein Subsample Schritt. Bei diesem werden die Perturbationen, die bereits in dem übergebenen Datensatz vorkommen, also keine echten Perturbationen sind, entfernt. Aus der replizierten übergebenen Datenmenge wird ein Subsample gezogen, welches die gleiche Größe wie die Menge der verbliebenen Perturbationen hat. Dieser Schritt wurde für den Angriff auf Anchors eingeführt, da die Replikationsrate, das heißt der Anteil der Perturbationen, der bereits in der übergebenen Datenmenge enthalten ist, von diesem Angriff ohne diese Veränderung besonders hoch ist, siehe Abbildung 4.1. Dadurch ist das Verhältnis zwischen echten Daten und perturbierten Daten unausgewogen, es ist jedoch ein Gleichgewicht gewünscht. Dieses Problem tritt über alle Datensätze auf und ist deutlich stärker ausgeprägt als bei den Angriffen auf die anderen Methoden, wie aus Abbildung 4.1(a) ersichtlich wird. Einen großen Einfluss auf die Replikationsrate hat dabei die Wahl des Verteilungsparameter `p`. Dies ist in Abbildung 4.1(b) veranschaulicht.

Diese hohen Replikationsraten für Anchors liegen darin begründet, dass, insbesondere für höhere Werte von `p`, sehr viele der Tupel die Länge 1 haben. Bei dem Ziehen von Beispielen aus der bedingten Wahrscheinlichkeit, werden dabei ganze Reihen aus der Datenverteilung, bzw. den Trainingsdaten, gezogen und nur das Feature, welches in der Kondition des Tupels vorkommt, wird an diese angepasst. Da einige Features in den Daten



(a) Vergleich Erklärbarkeitsmethoden.

(b) Auswirkung von Parameter `p`.

Abbildung 4.1: Veranschaulichung der Verteilung der Replikationsraten der Adversarial Models bei der Parameteroptimierung. Die Replikationsraten für Anchors werden vor dem Subsampling berechnet. (a) zeigt den Vergleich über die Erklärbarkeitsmethoden die angegriffen werden und die drei Datensätze und (b) zeigt die Auswirkungen des Parameters `p` des Adversarial Models für Anchors.

binär oder niedrig kategorial sind und die Grenzen für Konditionen auf kontinuierlichen Features diese nur in wenige Bereiche einteilen, kann es häufig auftreten, dass dieses eine Feature bereits der Kondition entspricht und nicht angepasst werden muss. Damit ergibt sich eine Perturbation die einem echten Datenpunkt entspricht. Mit zunehmend längeren Tupeln wird diese Wahrscheinlichkeit geringer.

4.2.2 LORE

Für den Angriff auf LORE werden keine weiteren Parameter verwendet, für die Methode werden auch keine Hyperparameter dem Nutzer zugänglich gemacht. Der Angriff besteht daraus, den genetischen Algorithmus für die Nachbarschaftsbestimmung von LORE für jeden Punkt der Trainingsdaten einmal durchzuführen. Von der so erzeugten Nachbarschaft werden dann `perturbation_multiplier` viele Datenpunkte zufällig ausgewählt die für die Perturbationen genutzt werden. Damit erreicht man auch hier wieder $n \cdot$ `perturbation_multiplier` viele Perturbationspunkte. Der genetische Algorithmus benötigt zur Auswertung der *fitness*-Funktionen einen Blackbox-Klassifizierer. Dafür wird hier f übergeben, dies kann natürlich nur eine Annäherung an das echte Verhalten sein, welches hier nachgeahmt werden soll, um möglichst echte Perturbationen zu erhalten, da während des Trainings des Adversarial Model es nicht selbst als Blackbox-Klassifizierer verwendet werden kann.

4.2.3 EXPLAN

Für das Adversarial Model für EXPLAN werden ähnlich wie bei LORE ebenfalls keine zusätzlichen Parameter benötigt. EXPLAN stellt dem Nutzer bei der Anwendung zwar die Parameter N und τ zur Verfügung, da für die Experimente allerdings die Default-Einstellungen für diese verwendet werden, ergibt eine Variation dieser für das Training des Adversarial Models wenig Sinn. Der Angriffsverlauf ist analog zu dem für LORE darin, dass für jeden Punkt die Nachbarschaft bestimmt wird, in diesem Fall über die ersten drei Schritte von EXPLAN. Danach findet auch wieder eine Auswahl von `perturbation_multiplier` vielen Punkten daraus statt. Auch hier wird bei der Erzeugung der Nachbarschaft ein Blackbox-Klassifizierer benötigt, für den ebenfalls f übergeben wird.

4.2.4 Parameteroptimierung

Die Wahl der Parameter für die Adversarial Models wurde über eine Parameteroptimierung festgelegt. Für alle drei Modelle wurde dabei der Parameter `perturbation_multiplier` aus der Menge $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ gewählt. Für Anchors wurden neben diesem auch die Parameter `p` und `n_samples_per_tuple` optimiert. Dabei waren die Optionen für `p` die Menge $\{0.4, 0.6, 0.8\}$ und für `n_samples_per_tuple` die Menge $\{1, 2, 5\}$. Zusätzlich zu diesen Parametern für das Adversarial Model für Anchors wurden außerdem drei Optionen für den Anteil eines Validierungssplit für das Lernen des Anchor-Erklärers verwendet, die Optionen hier waren 10%, 20% oder 30% des Trainingsdatensatzes. Wie später in Abschnitt 5.1.2 beschrieben wurden außerdem zwei verschiedene Optionen der Umwandlung der Anchors-Regeln zu einem Ranking ausgetestet. Für Anchors ergeben sich insgesamt 540 verschiedene Kombinationen. Für die Adversarial Models für LORE und EXPLAN wurden keine weiteren Parameter berücksichtigt, es ergeben sich also nur zehn Durchläufe.

Für die Optimierung wurden Experimente wie für die Einzelexperimente (siehe Abschnitt 4.3.1) durchgeführt, allerdings nur für den Fall einer einzelnen für ψ verwendeten Variable. Die Parameter der Angriffe wurden für jeden Datensatz einzeln optimiert. In jedem einzelnen Durchlauf wurden für die Auswahl die Fidelity und eine Fooled-Heuristik auf den Testdaten berechnet, eine genaue Beschreibung der Maße befindet sich in Abschnitt 5.2. Ein Angreifer hat potentiell die Möglichkeit die Daten, mit denen das modifizierte Modell e untersucht wird, selber bereitzustellen. In diesem Fall ist eine Überanpassung kein Problem für den Angreifer, sondern vielleicht sogar vorteilhaft. Daher wird hier kein extra Validierungsdatensatz für die Parameteroptimierung verwendet, sondern der Testdatensatz.

Für die Auswahl der Parameterkombination wird eine Pareto-Front über die berechneten Maße gebildet. Für die Kombinationen von Anchors wird dabei jedoch von den jeweils sechs Läufen, die identische Parameter für das Adversarial Model verwenden, der schlechteste Wert der Fooled-Heuristik verwendet (die Fidelity ist identisch, da sie das gleiche

Angegriffene Methode	Parameter	compas	cc	german
Anchors	`perturbation_multiplier`	1	1	2
	`p`	0.8	0.8	0.6
	`n_samples_per_tuple`	1	1	1
LORE	`perturbation_multiplier`	8	2	4
EXPLAN	`perturbation_multiplier`	3	1	3

Tabelle 4.1: Durch Parameteroptimierung bestimmte Parameter.

Adversarial Model benutzen) und diese somit zu einem Datenpunkt verschmolzen. Dieser stellt den "worst-case" über diese Optionen für den Angreifer dar, da dieser keine Kontrolle über den Anteil an Validierungsdaten und die Umwandlungsmethode hat. Die resultierenden Pareto-Fronten sind in Abbildung 4.2 dargestellt. Dabei wurde statt der Fidelity die entsprechende Fehlerrate verwendet, die sich als $1 - \text{Fidelity}$ ergibt. Der Idealpunkt liegt in der linken unteren Ecke bzw. bei $(0, 0)$ und wird für den Angriff auf EXPLAN auf Communities and Crime erreicht. Die Auswahl der Parameterkombinationen wurde dabei durch visuelle Betrachtung der Pareto-Fronten und Abwägung der Maße getroffen und ist rot markiert. Dabei ist zu beachten, dass die Fooled-Heuristik die subjektive Wahrnehmung der Täuschung nicht linear darstellt. Dabei sind die gleichen Differenzen im niedrigen Bereich ausdrucksstärker als in höheren Bereichen. Die ausgewählten Parameter-Einstellungen sind in Tabelle 4.1 aufgeführt.

4.3 Versuchsaufbau

Es werden drei verschiedene Arten an Experimenten durchgeführt. In den ersten Experimenten, den Einzelversuchen, wird jeweils die Anfälligkeit einer Erklärbarkeitsmethode auf den für sie spezialisierten Angriff untersucht. Die Sensitivitätsversuche testen, welche Auswirkung die Güte des OOD-Klassifizierers auf die Täuschung hat. In den Mix-Versuchen werden die Erklärbarkeitsmethoden jeweils auf ihre Anfälligkeit für nicht speziell auf sie ausgerichtete Angriffe untersucht.

Die ersten beiden Versuchsarten werden dabei entsprechend des Vorgehens von Slack et al. [45] durchgeführt. Die Mix-Versuche wurden in ihrer Arbeit erwähnt, aber nicht umfassend aufgeführt. Außerdem wird hier eine größere Vielfalt an Methoden untersucht.

Wie aus Abbildung 4.3 ersichtlich ist, haben der Anteil der Validierungsdaten und die Transformationsmethode keine große Auswirkung auf den Täuschungseffekt (gemessen an der Fooled-Heuristik, siehe Abschnitt 5.2.3). Für die Versuche mit Anchors als Erklärmethode wird daher der Anteil an Validierungsdaten auf 20% festgelegt und die erste der beiden in Abschnitt 5.1.2 verwendeten Umwandlungsmethoden für Anchors verwendet. Die

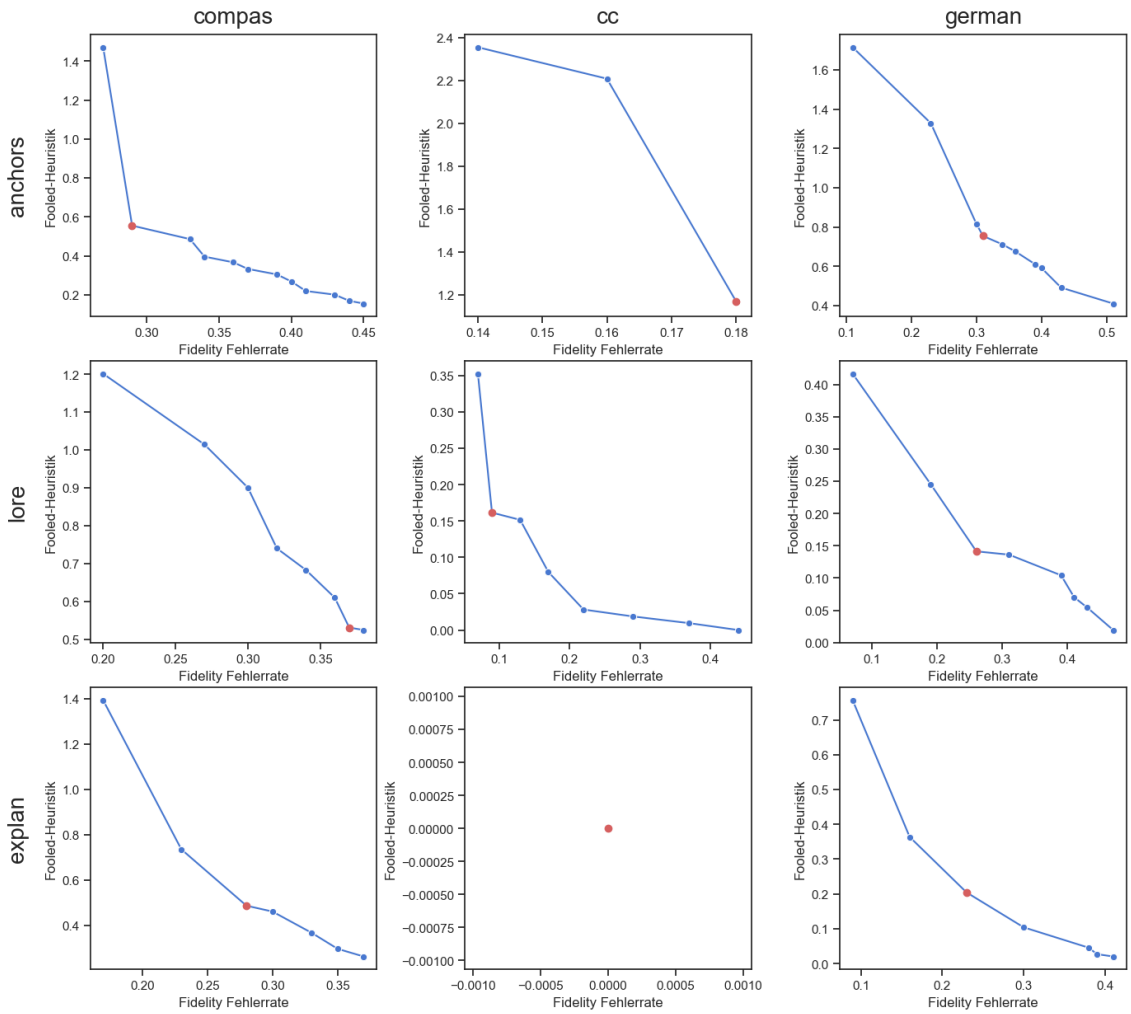
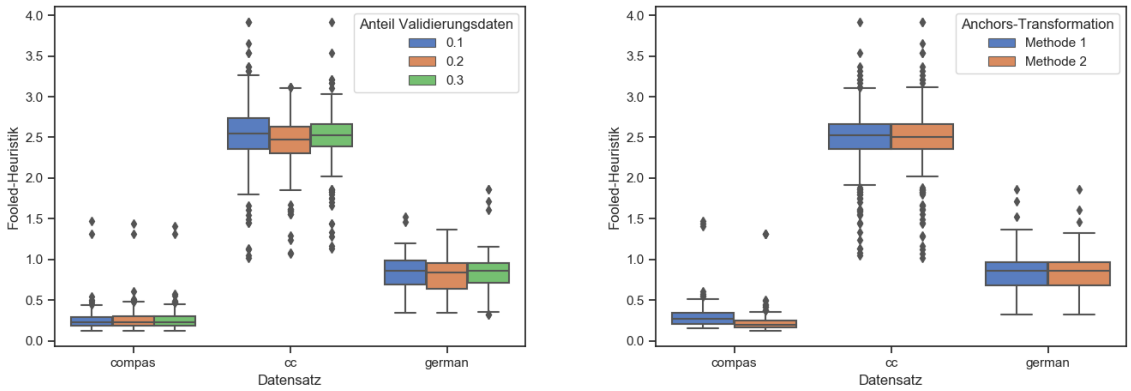


Abbildung 4.2: Pareto-Fronten der Parameteroptimierung, der Idealpunkt liegt bei $(0, 0)$. Die Spalten stellen die Datensätze dar und die Zeilen die angegriffene Erklärbarkeitsmethode. Die Achsen sind dabei zwischen den Grafiken unterschiedlich skaliert. Die ausgewählte Kombination ist rot markiert.



(a) Anteils Validierungsdaten.

(b) Transformations-Methode.

Abbildung 4.3: Vergleich der Effekte der nicht unter Kontrolle des Angreifers stehenden Optimierungsparameter für Anchors auf die Fooled-Heuristik über die Parameteroptimierung. (a) zeigt den Vergleich des Anteils Validierungsdaten und (b) zeigt den Vergleich der beiden Anchors-Transformationen.

Wahl des Validierungsanteil hat nur einen geringen Einfluss für höhere Werte der Fooled-Heuristik. Die Transformationsmethode hat auf COMPAS allerdings für niedrige Werte der Heuristik einen kleinen Einfluss. Im niedrigen Bereich der Heuristik hat dies eine stärkere Auswirkung auf das Täuschungsergebnis als in höheren Bereichen. Hier wurde die Methode so gewählt, dass der Angriff schwieriger wird.

4.3.1 Einzelversuche

Die Einzelversuche werden, wie in Abschnitt 2.3 beschrieben, für COMPAS und CC einmal mit der ursprünglichen mit einem Bias behafteten Funktion f, sowie jeweils einmal mit dem trainierten Adversarial Model mit ψ basierend auf einem einzelnen synthetischen Feature und ψ basierend auf zwei synthetischen Features durchgeführt. Für den German Credit Datensatz entfällt die Variante mit ψ basierend auf zwei Features und das für das erste Adversarial Model verwendete Feature ist `LoanRateAsPercentOfIncome`. Diese Experimente werden dabei für alle in Abschnitt 3.2 vorgestellten Erklärbarkeitsmethoden als Erklärer durchgeführt. Es wird jeweils das Adversarial Model verwendet, welches diesen Erklärer angreift. Die jeweils verwendeten Parameter sind in Tabelle 4.1 aufgeführt.

Es wird für jeden Punkt des Testdatensatzes eine Erklärung erzeugt. Die Ergebnisse werden für die Evaluation aggregiert wie in Kapitel 5 beschrieben wird. Diese Versuche dienen dazu zu evaluieren, wie effektiv die in Abschnitt 4.2 konstruierten Angriffe den jeweils angegriffenen Erklärer täuschen können.

4.3.2 Sensitivitätsversuche

Die Sensitivitätsversuche werden nur für die drei in dieser Arbeit neu untersuchten Erklärbarkeitsmethoden Anchors, LORE und EXPLAN durchgeführt. Sie dienen dabei dazu festzustellen wie stark der Täuschungserfolg von der Güte des OOD-Klassifizierers abhängt. Dabei werden Versuche wie für die Einzelversuche mit einem synthetischen Feature für ψ durchgeführt. Es wird nur der COMPAS Datensatz verwendet. Um OOD-Klassifzierer möglichst unterschiedlicher Güte zu erhalten, wird die Ausdrucksstärke des gelernten Random Forest eingeschränkt. Dabei wird die Anzahl der Bäume, die maximale Baumtiefe und die Anzahl an Beispielen pro Knoten bei dem dieser noch weiter aufgeteilt werden darf variiert. Die Anzahl an Bäumen wird aus der Menge $\{1, 2, 4, 8, 16, 32, 64\}$, die maximale Tiefe aus der Menge $\{1, 2, 4, 8, None\}$ und die minimale Anzahl Beispiel aus der Menge $\{2, 4, 8, 16, 32, 64\}$ gewählt. Insgesamt ergeben sich so 210 Durchläufe des Experiments. Im Gegensatz zu den Einzelexperimenten werden allerdings nicht die Erklärungsergebnisse aller Testdaten aggregiert, sondern nur die der ersten 100. Die Güte des OOD-Klassifizierers wird über den beim Trainieren des Adversarial Model bestimmten Testdatensatz aus echten Daten und Perturbationen evaluiert. Die Parameter für das Adversarial Model werden ebenfalls wie in Tabelle 4.1 gewählt.

4.3.3 Mix-Versuche

Durch die Mix-Versuche wird untersucht wie anfällig die Erklärbarkeitsmethoden für Angriffe sind, die nicht speziell auf sie ausgerichtet sind. Dies dient auch dazu herauszufinden, welche Methoden in der Lage sind diese Angriffe zu durchschauen. Die Versuche finden dabei grundsätzlich wie die Einzelversuche statt, auch die Parameter der Adversarial Models werden wie in Tabelle 4.1 aufgeführt verwendet. Die Mix-Versuche werden außerdem auch für beide verschiedenen Adversarial Models für die Datensätze COMPAS und CC durchgeführt. Sie unterscheiden sich von den Einzelversuchen darin, dass nicht die passenden Kombinationen von Adversarial Model und Erklärer zusammen getestet werden, sondern jeweils alle anderen möglichen Kombinationen. Alle drei Datensätze werden bei diesen Versuchen verwendet.

Kapitel 5

Evaluation

Da es sich bei allen in dieser Arbeit untersuchten Erklärbarkeitsmethoden um lokale Methoden handelt, werden Erklärungen immer nur für einzelne Datenpunkte erzeugt. Um ein Gesamtbild zu erhalten, werden die Erklärungen daher über die Testdaten aggregiert. Dabei wird bestimmt in welchem Anteil der Erklärungen der Testdaten (für Sensitivitätsversuche nur 100 Punkte daraus) ein Feature auf dem ersten, zweiten oder dritten Platz der Erklärung steht. LIME und KernelSHAP produzieren ein lineares Modell, welches den Features Gewichte zuweist, aus denen zu diesem Zweck ein Ranking (der Absolutbeträge) erstellt wird. Das Ergebnisformat der neu untersuchten Methoden Anchors, LORE und EXPLAN enthält allerdings keine direkte Umformungsmöglichkeit in ein Ranking. Wie aus diesen dennoch ein Ranking gewonnen werden kann, um dieses Evaluationsformat verwenden zu können, wird im nächsten Abschnitt beschrieben. Das Evaluationsformat von Slack et al. [45] soll dabei für eine bessere Vergleichbarkeit der Ergebnisse verwendet werden. Da nur die Plätze des Rankings betrachtet werden, ist nur ein qualitatives und nicht unbedingt ein quantitatives Ranking notwendig.

Neben der Aggregation der Erklärungen wird außerdem die Fidelity der Adversarial Model auf den Testdaten und eine Fooled-Heuristik bestimmt. Für die Sensitivitätsexperimente wird die Güte des OOD-Klassifizierers außerdem mit dem F1-Score gemessen. Auf diese Maße wird in Abschnitt 5.2 eingegangen. In Abschnitt 5.3 werden die Ergebnisse der Experimente evaluiert.

5.1 Ergebnistransformationen

Für die Bestimmung der Platzierungen der Features in einer Erklärung muss aus dieser ein Ranking der Features konstruiert werden. Die Methoden Anchors, LORE und EXPLAN erzeugen jeweils Erklärungen in Form von Regeln. LORE und EXPLAN extrahieren diese dabei aus einem Entscheidungsbaum. Dieser Baum wird verwendet, um aus diesem Feature Importances zu berechnen, da die Reihenfolge der Entscheidungsknoten in einem Baum

nicht zwingend als (qualitatives) Ranking ihrer Wichtigkeit für die Entscheidung angesehen werden können. Das Vorgehen dabei ist im nächsten Abschnitt beschrieben. Für Anchors werden zwei miteinander verwandte Varianten der Umwandlung der Anchors-Regeln im darauf folgenden Abschnitt vorgestellt. Die erste dieser Varianten wird dabei für die Experimente verwendet. Dies wird in Abschnitt 4.3 begründet.

5.1.1 Feature Importance für Entscheidungsbäume

Es gibt neben der hier vorgestellten Feature Importance Methoden diverse weitere Methoden die Wichtigkeit der Variablen für einen Entscheidungsbaum zu bestimmen. Darunter auch die von Breiman [9] für Random Forests beschriebene auf Feature-Permutationen beruhende Methode und die von EXPLAN benutzte TreeInterpreter [42] Methode für lokale Feature Importance. Da es sich im Fall der Ergebnisbäume von LORE und EXPLAN bereits um lokale Modelle handelt, wird hier allerdings die globale Feature Importance dieser Bäume bestimmt.

Für die Experimente dieser Arbeit wird dabei die Mean Decrease Impurity (MDI) Methode verwendet. Bei dieser Methode wird die Feature Importance eines Features in einem Baum über die Addition der Verbesserung in einem Aufteilungs-Kriterium für jeden Knoten in dem dieses Feature für die Prüfung verwendet wird, gewichtet mit dem Anteil der Daten die diesen Knoten erreichen, definiert. In Formeln ergibt sich die Feature Importance einer Variablen m wie folgt [28]:

$$Imp(m) = \sum_{T:v(X_T)=m} p(T)\Delta i(X_T). \tag{5.1}$$

Dabei ist T ein Knoten des Baumes, der kein Blatt ist, X_T die Aufteilung am Knoten T und $v(X_T)$ die für diese verwendete Variable. Die Wahrscheinlichkeit, dass dieser Knoten besucht wird $p(T)$ wird geschätzt durch N_T/N, wobei N_T die Anzahl der Datenpunkte ist, die den Knoten T erreicht und N die Gesamtzahl der Datenpunkte ist. $\Delta i(X_T)$ bezeichnet die Verbesserung im Kriterium durch die Knotenaufteilung.

Die MDI ist eigentlich als Maß für einen Random Forest definiert und wird dafür über alle Bäume gemittelt [28]. Da wir hier nur einen Baum haben, entfällt dieser Schritt. Dieses Maß wird bei Verwendung des *gini*-Kriteriums auch Gini Importance genannt [28] und wird auch für den DecisionTreeClassifier[1] in Scikit-learn [34] verwendet. Da LORE und EXPLAN allerdings YaDT [41] verwenden, wurde sie selber implementiert. Das verwendete Kriterium ist dabei entsprechend des Lernprozesses die *gain ratio* (siehe Abschnitt 3.1.1). Für die Berechnung der Feature Importance werden die Daten verwendet mit denen der Baum konstruiert wurde, also die Nachbarschaften der jeweiligen Methoden.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

5.1.2 Anchors-Transformationen

Für die Umwandlung der Anchors-Erklärungen in eine Ranking werden zwei Varianten vorgestellt. Die letztendlich für die Experimente verwendete Variante ist die erste der beiden. Der Konstruktionsprozess der Anchors-Regeln baut diese iterativ auf, wobei schrittweise die Kandidaten mit der höchsten Präzision ausgewählt werden. Die nacheinander aufgenommenen Konditionen können also als absteigend in ihrer Relevanz angesehen werden. Allerdings kann es vorkommen, dass ein Feature für mehrere Konditionen verwendet wird, welche in der Tupel-Repräsentation des Anchors auftreten. Daher müssen diese Ränge zusammengefasst werden.

Beide hier beschriebenen Methoden weisen den Positionen der Konditionen im iterativ aufgebauten Tupel einen Wert zu. Sie unterscheiden sich darin wie sie dies tun. Das Ranking ergibt sich durch Aufsummieren der Werte für jede Variable über alle Konditionen in denen sie vorkommt und anschließendes absteigendes Sortieren. Diese Werte sollten nicht als direkte Feature Importance Werte interpretiert werden und dienen nur dazu ein qualitatives Ranking zu erhalten in dem sich keine Features doppeln.

Die erste Variante weist einer Position $i \in \{0, 1, \ldots, |t| - 1\}$ den Wert $|t| - i$ zu. Dabei steht $|t|$ für die Länge des Tupels. Die zweite Variante weist den Wert $1/(i + 1)$ zu.

5.2 Evaluationsmaße

Für den Vergleich der Experimentergebnisse und für die Parameteroptimierung werden verschiedene Maße verwendet. Dabei wird die Anpassungsgüte des Adversarial Models an die ursprüngliche Blackbox f mit der Fidelity gemessen. Die Güte des OOD-Klassifizierers wird in den Sensitivitätsexperimenten mit dem F1-Score bewertet. Um die Güte der Täuschung der Erklärbarkeitsmethode in einer Maßzahl zusammenzufassen, damit darüber eine Optimierung stattfinden kann, wurde eine Fooled-Heuristik konstruiert, die hier ebenfalls beschrieben wird.

5.2.1 Fidelity

Die Fidelity beschreibt die Anpassungsgüte an ein anderes Modell [18]. Hier wird damit die Anpassungsgüte des Adversarial Model an die ursprüngliche Funktion f auf den Testdaten gemessen. Dabei wird die Fidelity als die Accuracy mit Bezug auf die Vorhersagen von f als Label berechnet. Die Accuracy ist wie folgt definiert [46]:

$$Accuracy = \frac{\text{Anzahl korrekter Vorhersagen}}{\text{Anzahl gesamte Vorhersagen}}. \tag{5.2}$$

Neben der Accuracy können auch andere Maße für eine Fidelity verwenden, wie beispielsweise der im nächsten Abschnitt erklärte F1-Score.

5.2.2 F1-Score

Der F1-Score ist das harmonische Mittel zwischen Precision und Recall. Er ist wie folgt definiert [46]:

$$\text{F1-}Score = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{2pr}{p+r}. \tag{5.3}$$

Dabei bezeichnet p die Precision und r den Recall, welche wie folgt berechnet werden [46]:

$$p = \frac{TP}{TP + FP}, \tag{5.4}$$

$$r = \frac{TP}{TP + FN}. \tag{5.5}$$

TP steht für True Positive, FP für False Positive und FN für False Negative. Das Positive bzw. Negative gibt dabei an, welche Klasse für einen Punkt dieser Gruppe vorhergesagt wurde und das True bzw. False, ob dieser Vorhersage zutrifft. Die Abkürzungen repräsentieren jeweils die Anzahl der Daten, die in diese Kategorie fallen. Da der F1-Score bei den Sensitivitätsexperimenten für den beim Trainieren der Adversarial Model erzeugten Testdatensatz aus Datenpunkten und Perturbationen berechnet wird, beziehen sich die Anzahlen auf diesen. Die Precision gibt also an welcher Anteil der der positiven Klasse zugeordneten Punkte dieser tatsächlich angehören. Der Recall beschreibt welcher Anteil der Punkte aus der positiven Klasse auch diese vorhergesagt bekommen.

5.2.3 Fooled-Heuristik

Die Fooled-Heuristik wurde konstruiert um das Täuschungsergebnis des Angriffs für die Parameteroptimierung in einer Zahl zusammenzufassen. Die Intuition hinter der Formulierung ist, dass das Feature auf dem die mit Bias behaftete Funktion f basiert möglichst nicht in der aggregierten Form der Erklärungen vorkommen soll, während das bzw. die Features, die für ψ verwendet werden möglichst an erster Stelle stehen sollen. Die Verteilung eines Features über die Plätze wird dabei als vierstellige diskrete Wahrscheinlichkeitsverteilung angesehen. Die Prozentwerte des Vorkommen pro Rang werden dabei als Wahrscheinlichkeiten für diesen interpretiert, der vierte Wert umfasst die Restwahrscheinlichkeit. Diese umfasst den Anteil der Erklärungen bei denen das Feature gar nicht, oder nur auf Rang 4 oder niedriger enthalten ist. Die Zielverteilung für das f-Feature ist also $(0, 0, 0, 1)$ und für die Features für ψ ist es $(1, 0, 0, 0)$.

Für jedes relevante Feature wird die KL-Divergenz $KL(p \parallel q)$ seiner Verteilung q von der entsprechenden Zielverteilung p berechnet. Die resultierenden Divergenzen werden über ein harmonisches Mittel miteinander verrechnet, um die Fooled-Heuristik zu erhalten.

Die Kullback-Leibler-Divergenz (KL-Divergenz, auch: Relative Entropie) ist ein asymetisches Abstandsmaß zwischen Wahrscheinlichkeitsverteilungen [7]. Sie ist definiert als [25, 7]:

$$KL(p \parallel q) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \tag{5.6}$$

für Wahrscheinlichkeitsverteilungen p und q. Bei diskreten Verteilungen über den gleichen Bereich $\mathcal{X}$ wird das Integral zur Summe und es gilt:

$$KL(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right). \tag{5.7}$$

Die KL-Divergenz nimmt Werte größer oder gleich 0 an. Außerdem gilt [25]:

$$KL(p \parallel q) = 0 \iff p(x) = q(x) \forall x. \tag{5.8}$$

Die Fooled-Heuristik verhält sich nicht linear zur subjektiven Täuschungsqualität. Die gleichen absoluten Unterschiede im niedrigen Bereich wirken ausschlaggebender als in höheren Bereichen.

5.3 Ergebnisse

In den nächsten Abschnitten werden die Ergebnisse der in Kapitel 4 beschriebenen Experimente evaluiert. Dabei wird die Aufteilung in Einzelversuche, Sensitivitätsversuche und Mix-Versuche beibehalten. Aufgrund von Ausführungsunterschieden können sich die Fidelity und Fooled-Heuristik der Versuche von der Parameteroptimierung unterscheiden.

5.3.1 Einzelversuche

Abbildung 5.1 zeigt die Ergebnisse der Einzelversuche auf dem COMPAS Datensatz. Unter den Versuchen mit dem ursprünglichen mit Bias behafteten Klassifizierer f ist LIME die einzige Methode, die andere Features neben `race` für die Erklärungen verwendet. Dabei werden hier jedoch nur die Ränge betrachtet, die Gewichte die LIME diesen Features zuweist könnten sehr gering sein. Für den Angriff mit einem Feature lassen sich alle Erklärbarkeitsmethoden relativ gut täuschen, dabei benutzt gerade LORE jedoch sehr häufig nicht für f oder ψ relevante Features als wichtigstes Feature der Erklärungen. Auffällig ist dabei, dass das Feature `length_of_stay` fast 40% des ersten Rangs ausmacht. Dieses Feature hat dabei weder mit dem Zielwert, noch mit der Variable `race` eine starke Korrelation. KernelSHAP, Anchors und EXPLAN haben etwa gleichviel Vorkommen des sensitiven Features auf dem zweiten Rang, jedoch ist das Feature bei KernelSHAP vollständig vom ersten Rang verdrängt. KernelSHAP wird also etwas besser getäuscht. Keine der drei neu untersuchten Methoden erreicht eine vollständige Verdrängung von `race` vom ersten Platz der Erklärungen. Diese Betrachtungen müssen immer unter Berücksichtigung der Fidelity des Adversarial Models stattfinden, diese ist für alle Einzelversuche in Tabelle 5.1 aufgeführt. Eine gute Täuschung bei niedriger Fidelity heißt zwar, dass die Methode getäuscht wurde, aber auch, dass der Bias von f in einer Anwendung des Adversarial Models abgeschwächt wird. Andererseits kann immer eine gute Fidelity erreicht werden, wenn dafür keine Täuschung stattfindet. Die neu untersuchten Adversarial Models haben, im Vergleich

	COMPAS		CC		German
	ein Feature	zwei Features	ein Feature	zwei Features	ein Feature
LIME	1.00	0.99	1.00	1.00	1.00
SHAP	0.87	0.86	0.97	0.96	0.98
Anchors	0.68	0.66	0.82	0.82	0.61
LORE	0.63	0.65	0.92	0.90	0.81
EXPLAN	0.72	0.74	1.00	1.00	0.77

Tabelle 5.1: Fidelity-Werte der Adversarial Models in den Einzelversuchen. Die Spalten beschreiben dabei die verschiedenen Angriffe über die Datensätze und die Zeilen die von dem Adversarial Model angegriffene Erklärbarkeitsmethode. Alle Werte sind auf zwei Nachkommastellen gerundet.

zu den von Slack et al. entwickelten, merkbar geringere Fidelity-Werte auf COMPAS und German. Auf Communities and Crime ist dieser Unterschied weniger stark bis gar nicht vorhanden. Die zu der Abbildung 5.1 zugehörigen Werte der Fooled-Heuristik finden sich in den Tabellen 5.3 und 5.4 für einen numerischen Vergleich dieser Darstellungen. Aus Tabelle 5.3 lässt sich erkennen, dass, entsprechend der Heuristik, die Täuschung für den Angriff mit einem Feature bei LIME am besten ist und bei Anchors am schlechtesten, allerdings nur knapp.

Für den Angriff mit zwei Features auf dem COMPAS Datensatz sind die Ergebnisse für LORE und EXPLAN für den ersten Rang recht vergleichbar zu den Ergebnissen für den Angriff mit einem Feature. Lediglich der Anteil des Features `unrelated_column_one` wird mit dem Feature `unrelated_column_two` geteilt. Das Verhältnis zwischen beiden Features ist für alle Methoden auf allen Rängen recht ausgeglichen. Die Täuschung von LIME, LORE und EXPLAN wirkt etwas besser, da das Feature `race` insgesamt etwas seltener auftritt. Dagegen hat sich der Angriff für KernelSHAP und Anchors im Verhältnis verschlechtert. Gerade Anchors verwendet kaum die für ψ verwendeten Features für Erklärungen und das sensitive Feature nimmt etwa die Hälfte des ersten Rangs ein. Auch für diesen Angriff verwendet LORE das Feature `length_of_stay` sehr häufig, es nimmt sogar über 50% des ersten Rangs ein. Ein Betrachten der Fooled-Heuristik in Tabelle 5.4 ergibt, dass LIME wieder am besten getäuscht wurde und Anchors am schlechtesten.

Die Versuche auf dem Communities and Crime Datensatz stellen sich etwas extremer dar. Wie aus Abbildung 5.2 ersichtlich ist, werden alle Methoden außer Anchors sowohl für den Angriff mit einem, als auch mit zwei Features deutlich besser getäuscht. Dies gilt insbesondere für LIME und EXPLAN, welche bei einer Fidelity von 100% das sensitive Feature fast vollständig von allen drei Rängen verdrängt haben. KernelSHAP erreicht bei dem Angriff mit nur einem Feature eine bessere Täuschung als auf COMPAS, das Feature `racePctWhite numeric` nimmt jedoch noch einen recht großen Teil des zweiten Rangs ein. Anchors dagegen werden eher schlecht getäuscht für den Angriff mit einem Feature und

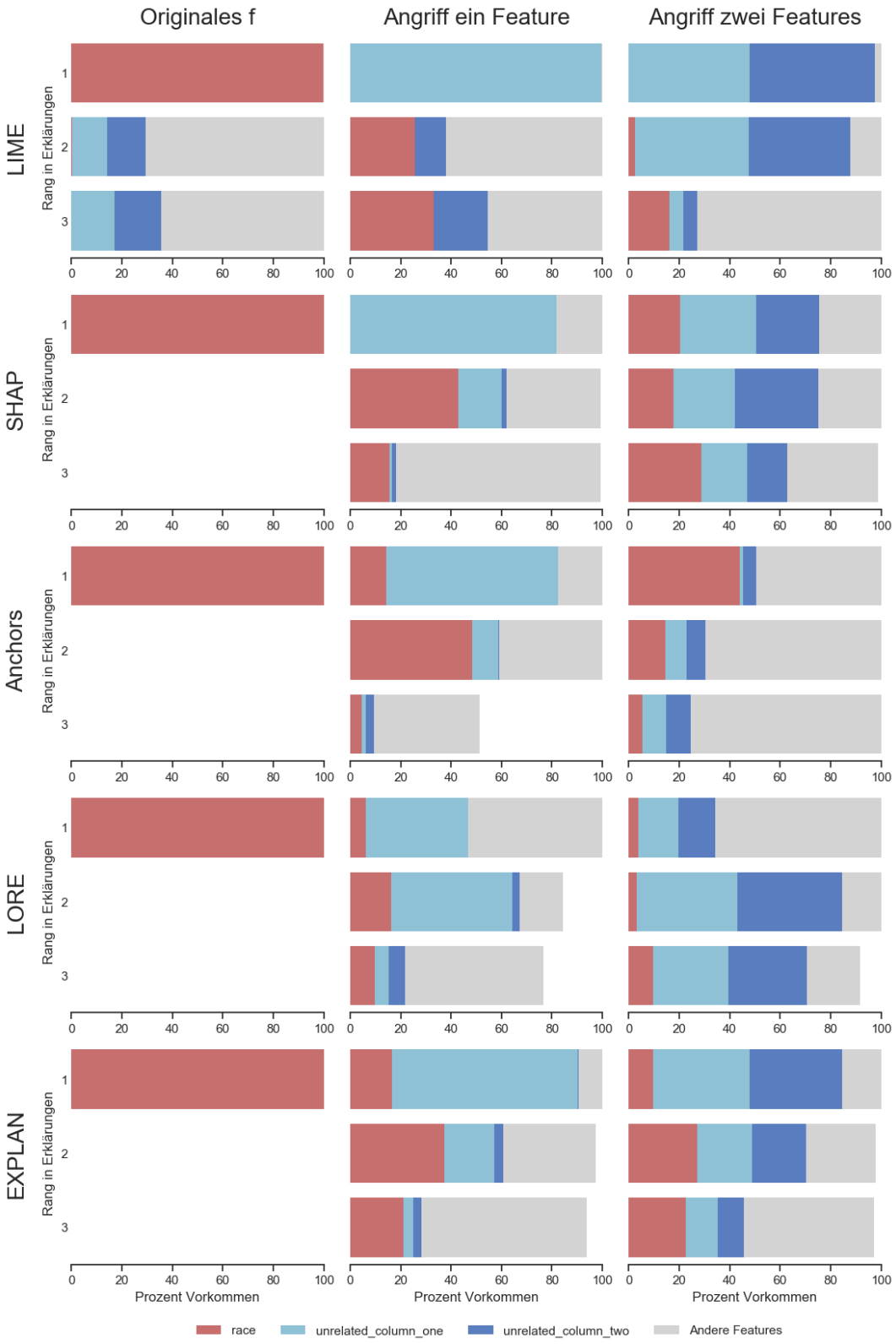


Abbildung 5.1: Darstellung der Erklärungsaggregation für die Einzelversuche auf COMPAS. Die Spalten stellen die verschiedenen Angriffe dar und die Zeilen die angegriffenen Erklärbarkeitsmethoden, die auch zur Erzeugung der Erklärungen verwendet wurden.

fast gar nicht für den Angriff mit zwei Features. LORE verwendet wieder häufig nicht relevante Features, bündelt dies jedoch hier nicht so stark, was möglicherweise daran liegt, dass der Communities and Crime Datensatz deutlich mehr Variablen enthält. Die Balance der beiden Feature von ψ ist bei allen Methoden außer LIME gegeben, welche das Feature `unrelated_column_two` deutlich häufiger für Erklärungen verwendet. Die Werte der Fooled-Heuristik auf dem Communities and Crime Datensatz finden sich in den Tabellen 5.5 und 5.6. Aus diesen ist ersichtlich, dass bei dem Angriff mit einem Feature EXPLAN minimal besser getäuscht wird als LIME, während Anchors wie in der Abbildung 5.2 erkennbar, den höchsten Wert erhält. Für den Angriff mit zwei Features gilt dies ebenfalls.

Abbildung 5.3 zeigt die Ergebnisse für den German Credit Datensatz. Die Verteilungen der Features ähneln denen des Angriffs mit einem Feature auf dem Communities and Crime Datensatz. Die Täuschung von LIME und EXPLAN ist allerdings etwas schlechter als dort. Dafür ist die Täuschung von Anchors etwas besser. Die Werte der Fooled-Heuristik aus Tabelle 5.7 sagen aus, dass KernelSHAP am besten getäuscht wurde und Anchors am schlechtesten. Dass der Wert der Heuristik hier 0 ist, also ihren besten Wert annimmt, obwohl das sensitive Feature auf den Rängen 2 und 3 auftritt, ist ein Anzeichen dafür, dass die Heuristik nicht exakt dem subjektiven Eindruck entspricht.

Die erzielten Ergebnisse für LIME und KernelSHAP weichen von den von Slack et al. [45] erzielten Ergebnissen ab. Auf dem COMPAS Datensatz erhalten sie für LIME deutlich geringere Anteile des Features `race` auf dem zweiten und dritten Rang. Für KernelSHAP auf diesem Datensatz erhalten sie dafür einen größeren Anteil des sensitiven Features auf COMPAS für den Angriff mit einem Feature. Bei dem Angriff mit zwei Features stellen sie auch leicht höhere Anteile des sensitiven Features dar, insbesondere enthält ihr erster Rang allerdings keine nicht relevanten Features. Auf dem Communities and Crime Datensatz unterscheidet sich vor allem das LIME Ergebnis für zwei Features deutlich, dort sind die ersten beiden Ränge vollständig mit den beiden synthetischen Features in etwa ausgeglichenem Verhältnis gefüllt. Für die KernelSHAP Ergebnisse zeigen sich hier geringere Anteile des sensitiven Features. Die Ergebnisse für LIME auf dem German Credit Datensatz stimmen mit den hier erreichten Ergebnissen überein, für KernelSHAP gibt es jedoch Abweichungen. Das Feature `Gender` ist deutlich stärker auf dem zweiten Rang vertreten. Der erste Rang in der Grafik von Slack et al. erreicht außerdem nicht 100%, was impliziert, dass es einige Testinstanzen mit leeren Erklärungen geben muss. Dies konnte in den für diese Arbeit durchgeführten Experimenten nicht beobachtet werden.

Insgesamt wurde über alle Einzelexperimente hinweg Anchors am schlechtesten getäuscht. Dabei hat Anchors außerdem meist die schlechteste Fidelity (siehe Tabelle 5.1). Daher kann diese schlechte Täuschung nicht daher stammen, dass der OOD-Klassifizierer fast immer eine Entscheidung für das ursprüngliche f fällt, welches eine hohe Fidelity bei schlechter Täuschung verursachen würde. Ein Grund für die Resistenz von Anchors gegenüber dem Angriff könnte darin liegen, dass es weniger oder schlechter zu unterschei-

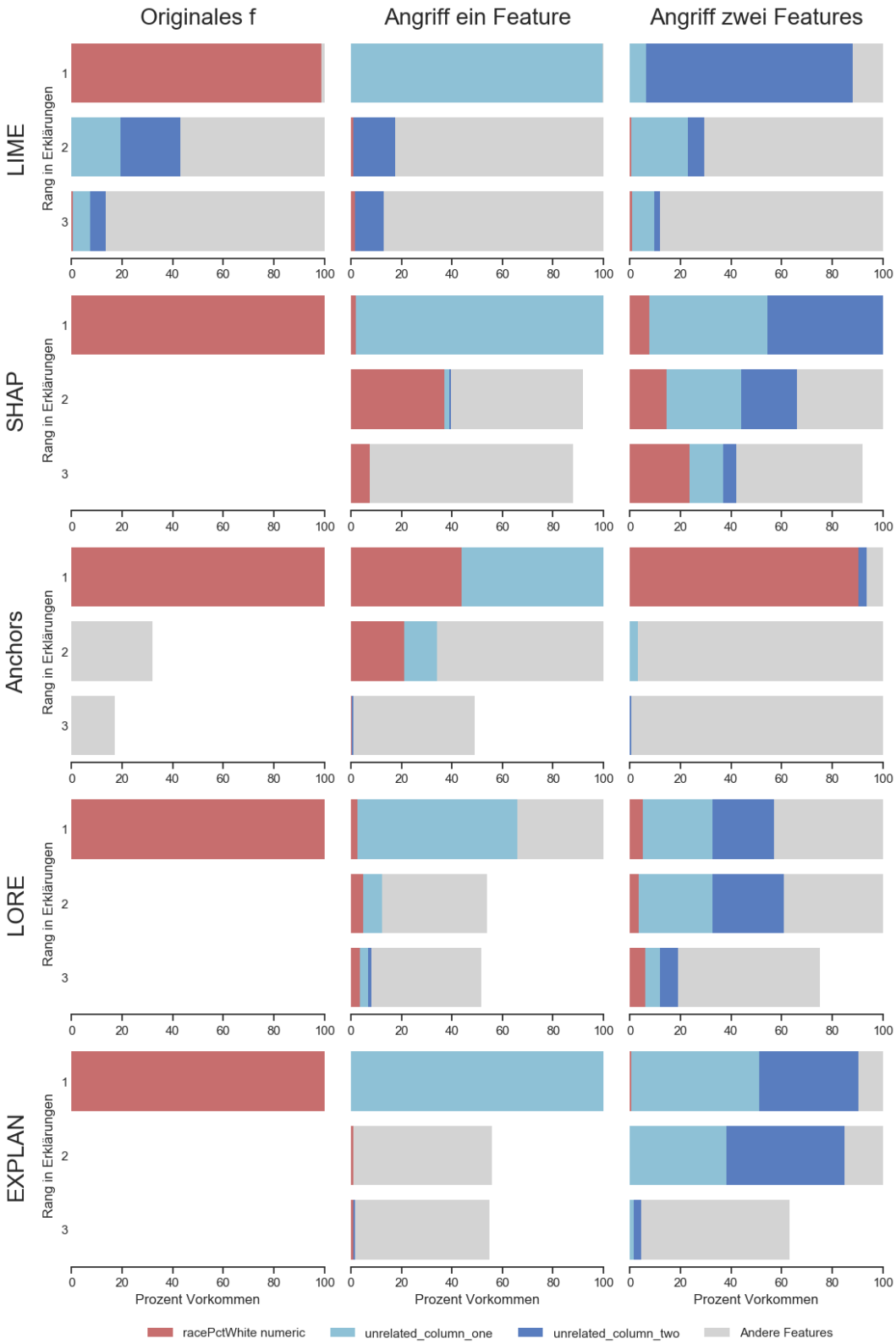


Abbildung 5.2: Darstellung der Erklärungsaggregation für die Einzelversuche auf Communities and Crime. Die Spalten stellen die verschiedenen Angriffe dar und die Zeilen die angegriffenen Erklärbarkeitsmethoden, die auch zur Erzeugung der Erklärungen verwendet wurden.

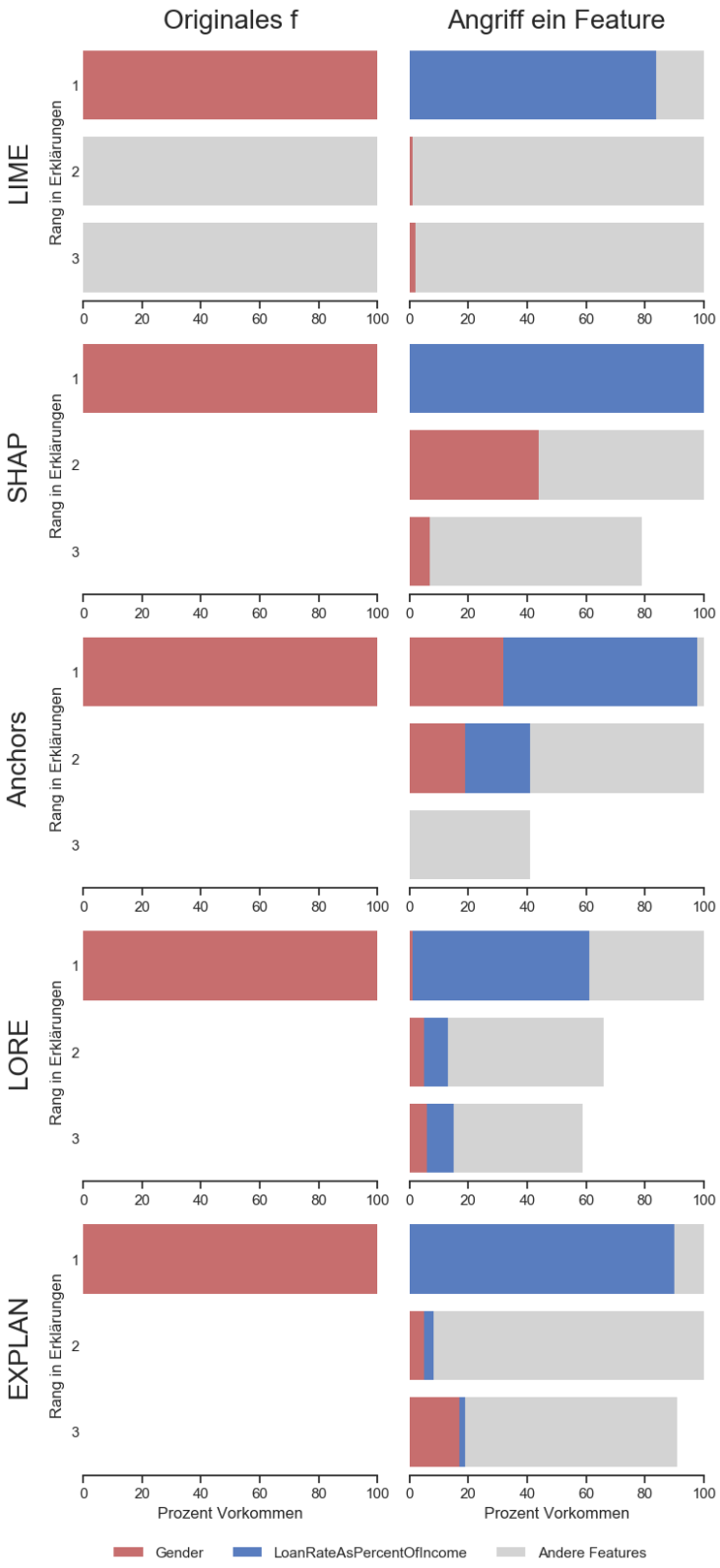


Abbildung 5.3: Darstellung der Erklärungsaggregation für die Einzelversuche auf German Credit. Die Spalten stellen die verschiedenen Angriffe dar und die Zeilen die angegriffenen Erklärbarkeitsmethoden, die auch zur Erzeugung der Erklärungen verwendet wurden.

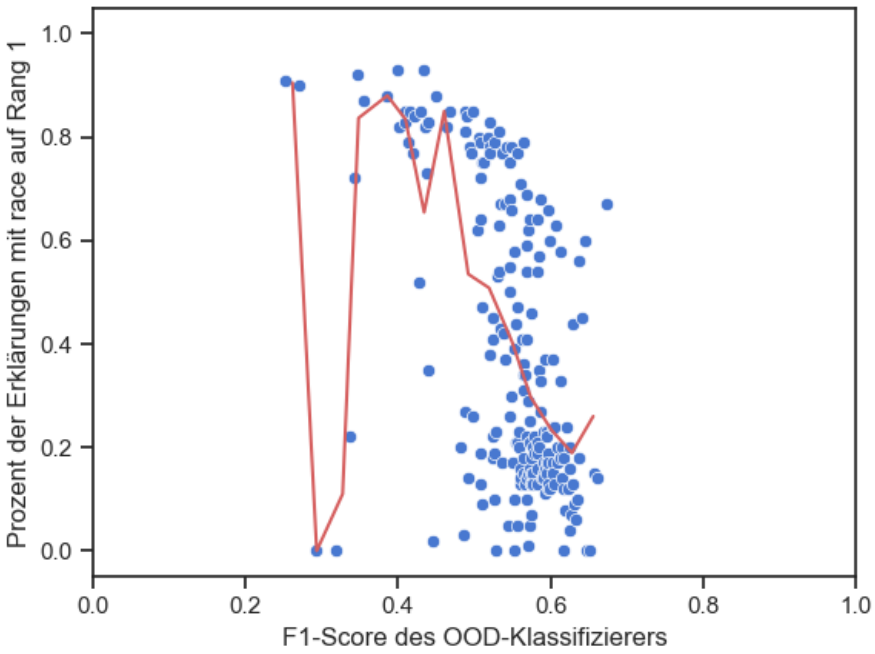


Abbildung 5.4: Ergebnisse der Sensitivitätsversuche für Anchors auf COMPAS. Die rot dargestellte Linie verbindet die Mittelwerte über Gruppierung der Daten in 15 Gruppen entlang der x-Achse.

dende Perturbationen erzeugt. Dafür spricht die hohe Replikationsrate des Angriffs ohne Subsample-Schritt (siehe Abschnitt 4.2.1). Eine andere mögliche Begründung ist, dass der hier konstruierte Angriff nicht geeignet ist, die Perturbationen, die Anchors erzeugt, nachzubilden.

5.3.2 Sensitivitätsversuche

Die Ergebnisse der Sensitivitätsversuche für Anchors sind in Abbildung 5.4 dargestellt. Alle Kombinationen der Einschränkungen für den OOD-Klassifizierer haben in keinem Modell resultiert, das einen F1-Score von weniger als 0.25 erreicht, jedoch wird auch kein höherer F1-Score als etwa 0.7 erreicht. Auf der niedrigen Seite sind die Punkte etwas spärlicher, weichen aber stark voneinander ab bei dem Anteil des Features `race` auf dem ersten Platz der Erklärungen. Danach gibt es einen steilen Abstieg des Anteils für einen F1-Score im Bereich von 0.5 bis 0.6.

Die Verläufe für LORE (Abbildung 5.5) und EXPLAN (Abbildung 5.6) sehen sehr anders aus, sind sich untereinander aber deutlich ähnlicher. Beide zeigen einen relativ gleichmäßigen Abstieg des Anteils von `race` bei höherem F1-Score des OOD-Klassifizierers. Der Abstieg ist für LORE dabei etwas sanfter und beginnt sehr früh. Selbst für einen F1-Score nahe 0 resultiert LORE bereits in Erklärungen die das Feature `race` nur in 60-80% der

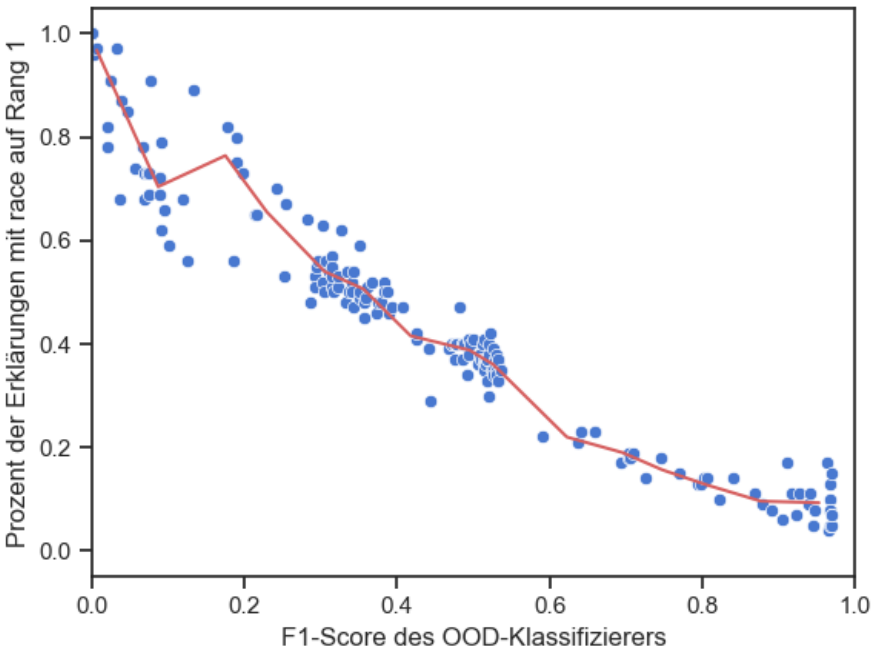


Abbildung 5.5: Ergebnisse der Sensitivitätsversuche für LORE auf COMPAS. Die rot dargestellte Linie verbindet die Mittelwerte über Gruppierung der Daten in 15 Gruppen entlang der x-Achse.

Fälle auf dem ersten Platz positionieren. Im Vergleich dazu erklärt LORE die ursprüngliche Blackbox, wie in Abbildung 5.1, zu sehen ist, ausschließlich mit `race`, entsprechend der echten Entscheidungsgründe von f. Für LORE ist es außerdem möglich, ein Adversarial Model mit nahezu vollem F1-Score von 1 zu erzeugen, allerdings wird für diese das Feature `race` dennoch nicht vollständig vom ersten Erklärungsplatz verdrängt.

EXPLAN reagiert erst etwas später auf den Angriff, bei F1-Scores von etwa 0.2. Der Abstieg danach ist etwas steiler, hier wird allerdings weder ein nahezu idealer F1-Score erreicht, noch ein vollständiges Verdrängen von `race`.

Im Vergleich zu den Ergebnissen von Slack et al. [45] für LIME und KernelSHAP, scheinen die hier untersuchten Methoden deutlich schneller auf Täuschungsversuche zu reagieren. Die von ihnen gefundene recht hohe Genauigkeitsschwelle für LIME erreichen die Methoden nicht annähernd. Der Verlauf von EXPLAN ähnelt dem von KernelSHAP am meisten, jedoch beginnt der Abstieg dort erst bei einem F1-Score von etwa 0.45. Allerdings ist es möglich LIME und KernelSHAP vollständig zu täuschen, mit vollem F1-Score, was sich für die hier untersuchten Methoden nicht ergibt.

5.3.3 Mix-Versuche

Bei den Angriffen auf den COMPAS Daten mit einem Feature lässt sich, wie in Abbildung 5.7 zu sehen ist, LIME von den Adversarial Models für Anchors, LORE und EXPLAN

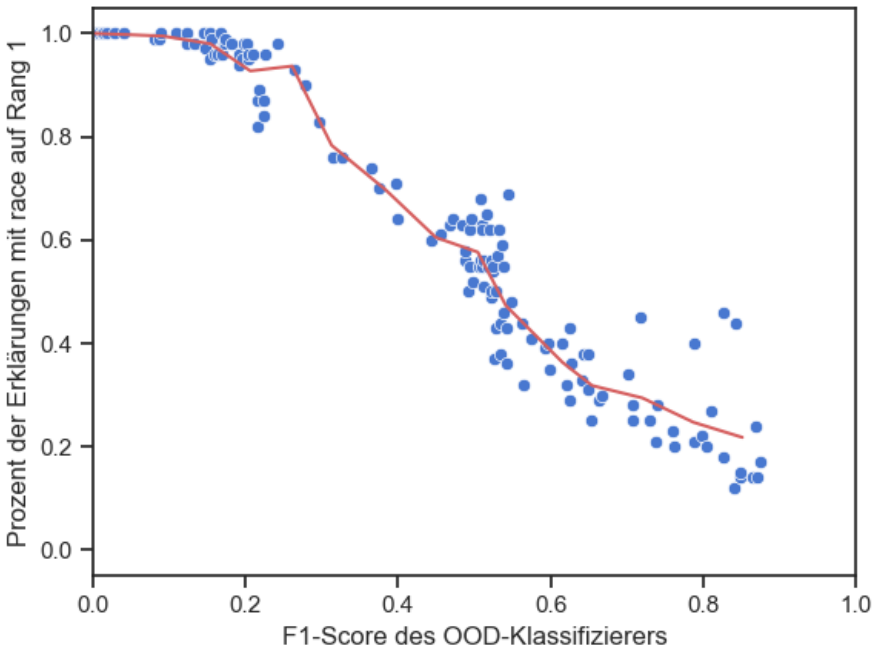


Abbildung 5.6: Ergebnisse der Sensitivitätsversuche für EXPLAN auf COMPAS. Die rot dargestellte Linie verbindet die Mittelwerte über Gruppierung der Daten in 15 Gruppen entlang der x-Achse.

in sehr ähnlicher Weise täuschen. Dabei ist der erste Rang nahezu immer mit dem Feature `unrelated_column_one` belegt und der zweite mit dem sensitiven Feature. Bei dem Adversarial Model für KernelSHAP sind diese Ränge getauscht, LIME wird also weniger getäuscht. Alle anderen Angriffe täuschen LIME schlechter als der auf LIME zugeschnittene Angriff. Der jeweils dritte Rang in diesen Erklärungen wird im Fall von Anchors fast vollständig von `two_year_recid` belegt, im Fall von EXPLAN zu etwa 90% von `priors_count` und im Fall von KernelSHAP zu ungefähr 60% ebenfalls von `priors_count`. Bei LORE ist der Platz aufgeteilt auf mehrere Features.

KernelSHAP lässt sich von den anderen Angriffen auf Anchors, LORE und EXPLAN nahezu gleich gut täuschen wie von dem spezialisierten Angriff, von dem Angriff auf Anchors und EXPLAN etwas weniger. Kaum bis gar nicht täuschen lässt sich KernelSHAP von dem Angriff auf LIME. Anchors lässt sich von den anderen Angriffen nicht sonderlich gut täuschen, von dem auf LIME abgezielten Angriff jedoch gar nicht. Hier entsprechen die Ergebnisse denen der ursprünglichen Funktion f (vergleiche Abbildung 5.1). Auch der Angriff auf KernelSHAP täuscht Anchors fast gar nicht. Die Angriffe auf LORE und EXPLAN haben etwas mehr Erfolg.

LORE lässt sich von den Angriffen auf LIME und KernelSHAP etwa gleich gut täuschen, etwas mehr allerdings von KernelSHAP. Beides täuscht LORE aber nicht stark, das Feature `race` tritt dennoch in etwa der Hälfte der Erklärungen an erster Stelle auf und das gewünschte Feature `unrelated_column_one` tritt nicht sehr häufig auf. Von den Angriffen auf Anchors und EXPLAN lässt es sich jedoch in etwa so gut täuschen wie durch den gezielten Angriff. LORE verwendet auch in diesen Erklärungen, ähnlich wie bei den Einzelversuchen, häufig das Feature `length_of_stay` für Erklärungen. Bein Anchors und EXPLAN wird dieses dabei häufiger als wichtigstes Feature verwendet als `unrelated_column_one`. Bei KernelSHAP tritt es im Rang 1 häufiger auf als `race`. Diese Tendenz von LORE viel das Feature `length_of_stay` zu verwenden gilt auch für den Angriff mit zwei Features.

EXPLAN lässt sich von LIME fast gar nicht täuschen. Auch der Angriff des Adversarial Models für KernelSHAP ist nicht viel besser darin. Die Angriffe auf Anchors und LORE funktionieren jedoch in etwa so gut wie der gezielte Angriff. Allgemein fällt auf, dass sich über alle Methoden die Erklärungen für die Angriffe auf LORE und EXPLAN stark ähneln. Aus Tabelle 5.3 ist ersichtlich, dass entsprechend der Fooled-Heuristik LIME am stärksten von allen Angriffen außer dem auf KernelSHAP getäuscht wird. Die geringste Täuschung wirkt auf KernelSHAP und EXPLAN für den Angriff auf LIME ein. Optisch wirkt die Erklärung von Anchors für den LIME-Angriff am besten, der drastische Unterschied in der Fooled-Heuristik entsteht durch etwa 0.5% die bei Anchors auf dem ersten Platz nicht durch `race` belegt sind. Die zu allen Mix-Experimenten gehörigen Fidelity-Werte finden sich in Tabelle 5.2. Sie entsprechen in etwa den Werten für die Einzelversuche.

Die Ergebnisse für die Mix-Versuche auf den COMPAS Daten für Angriffe mit zwei Features sind in der Abbildung 5.8 dargestellt. Im Gegensatz zu den Ergebnissen für ein

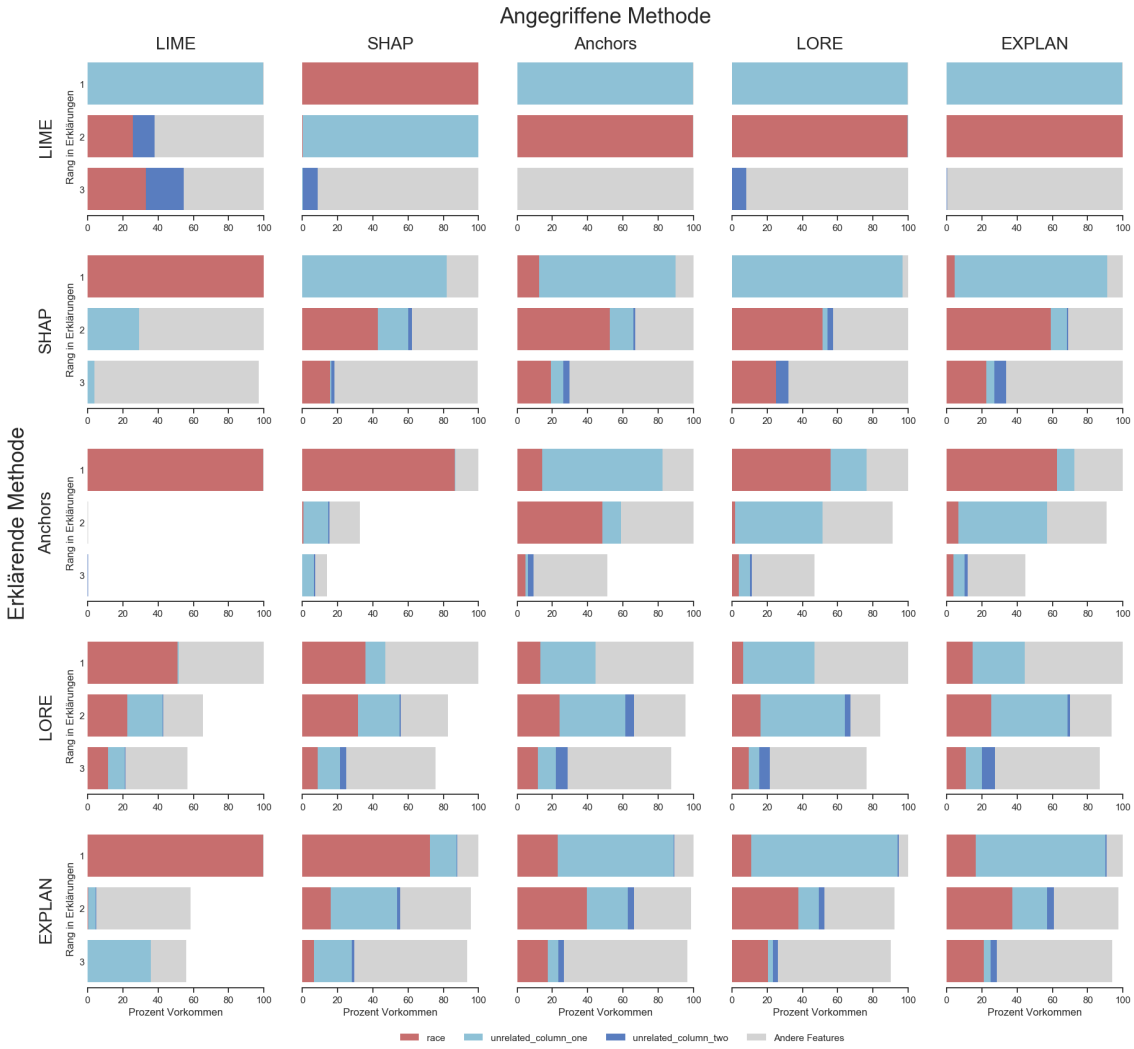


Abbildung 5.7: Erklärungsaggregationen der Mix-Versuche auf COMPAS mit einem Feature `unrelated_column_one` für das Adversarial Model. Auf der Diagonale werden die passenden Ergebnisse der Einzelversuche mit abgebildet.

	COMPAS		CC		German
	ein Feature	zwei Features	ein Feature	zwei Features	ein Feature
LIME	1.00	0.99	1.00	1.00	1.00
SHAP	0.87	0.87	0.96	0.96	0.97
Anchors	0.68	0.66	0.82	0.82	0.61
LORE	0.64	0.67	0.91	0.90	0.80
EXPLAN	0.74	0.74	1.00	1.00	0.73

Tabelle 5.2: Fidelity-Werte der Adversarial Models in den Mix-Versuchen. Die Spalten beschreiben dabei die verschiedenen Angriffe über die Datensätze und die Zeilen die von dem Adversarial Model angegriffene Erklärbarkeitsmethode. Für die Mix-Versuche wird jeweils nur einmal ein Adversarial Model trainiert und von allen nicht angegriffenen Erklärbarkeitsmethoden erklärt. Daher reicht hier eine Spalte und es muss nicht zwischen den Erklärern differenziert werden. Alle Werte sind auf zwei Nachkommastellen gerundet.

		Angegriffene Methode				
		LIME	SHAP	Anchors	LORE	EXPLAN
Erklärer	LIME	0.01	9.81	0.01	0.01	0.01
	SHAP	20.72	0.33	0.45	0.06	0.28
	Anchors	8.48	3.01	0.57	1.20	1.66
	LORE	2.79	1.72	0.85	0.55	0.91
	EXPLAN	20.72	2.31	0.67	0.32	0.50

Tabelle 5.3: Werte der Fooled-Heuristik für die Einzelversuche und die Mix-Versuche auf COMPAS mit einem Feature `unrelated_column_one` für das Adversarial Model. Die Spalten entsprechen den vom Adversarial Model angegriffenen Methoden und die Zeilen den Methoden die zum Erklären verwendet werden. Alle Werte sind auf zwei Nachkommastellen gerundet.

		Angegriffene Methode				
		LIME	SHAP	Anchors	LORE	EXPLAN
Erklärer	LIME	0.39	11.90	5.30	0.57	11.90
	SHAP	20.72	1.21	1.02	0.95	1.50
	Anchors	7.29	4.31	1.94	2.49	2.97
	LORE	5.40	2.26	0.64	0.46	1.01
	EXPLAN	8.35	2.59	0.90	0.83	0.95

Tabelle 5.4: Werte der Fooled-Heuristik für die Einzelversuche und die Mix-Versuche auf COMPAS mit zwei Features `unrelated_column_one` und `unrelated_column_two` für das Adversarial Model. Die Spalten entsprechen den vom Adversarial Model angegriffenen Methoden und die Zeilen den Methoden die zum Erklären verwendet werden. Alle Werte sind auf zwei Nachkommastellen gerundet.

Feature wird LIME hier nur durch den auf LORE ausgerichteten Angriff neben dem auf sich selbst spezialisierten Angriff getäuscht. Dies geschieht vergleichbar gut. Die Erklärungen für die anderen Angriffe ähneln sich stark.

KernelSHAP lässt sich von LIME nicht täuschen, allerdings von den anderen Angriffen in einem ähnlichen Ausmaß wie für den gezielten Angriff. Unter diesen lässt sich KernelSHAP von EXPLAN am wenigsten täuschen. Anchors zeigt wieder relativ hohe Resistenz gegen die Täuschungsversuche. Das Vorkommen des Features `race` ist zwar stark verringert für einige Angriffe, jedoch werden auch die synthetischen Features kaum für Erklärungen verwendet. Der Angriff auf LIME wird von Anchors wieder in etwa wie das ursprüngliche f erklärt.

Die Erklärungen von LORE entsprechen in etwa denen für Angriffe mit einem Feature wobei das Vorkommen des ersten synthetischen Features auf beide verteilt wurde. Diese beiden Features kommen aber allgemein etwas häufiger vor. Für EXPLAN gilt das gleiche. Die Vorkommen der beiden synthetischen Features über alle Erklärungen ist in etwa ausgeglichen. Der Fooled-Heuristik nach findet die beste Täuschung, die nicht mit einem gezielten Angriff erreicht wird, bei LIME Erklärungen für den LORE Angriff statt (siehe Tabelle 5.4). Die schlechteste Angriffskombination ist wieder der Angriff von LIME erklärt durch KernelSHAP.

Von den Angriffen mit einem Feature auf den Communities and Crime Daten lässt sich LIME immer täuschen. Dabei ist der Angriff auf Anchors noch am wenigsten effektiv, da dort das sensitive Feature meist den zweiten Platz einnimmt. Dies kann Abbildung 5.9 entnommen werden. KernelSHAP dagegen lässt sich von dem auf LIME gezielten Angriff kaum täuschen und von den anderen drei Angriffen etwa gleich gut, aber etwas besser. Noch viel weniger effektiv sind die Angriffe bei Erklärungen durch Anchors. Diese lassen sich beinahe gar nicht täuschen. Auch die Angriffe auf LORE sind nicht sonderlich erfolgreich.

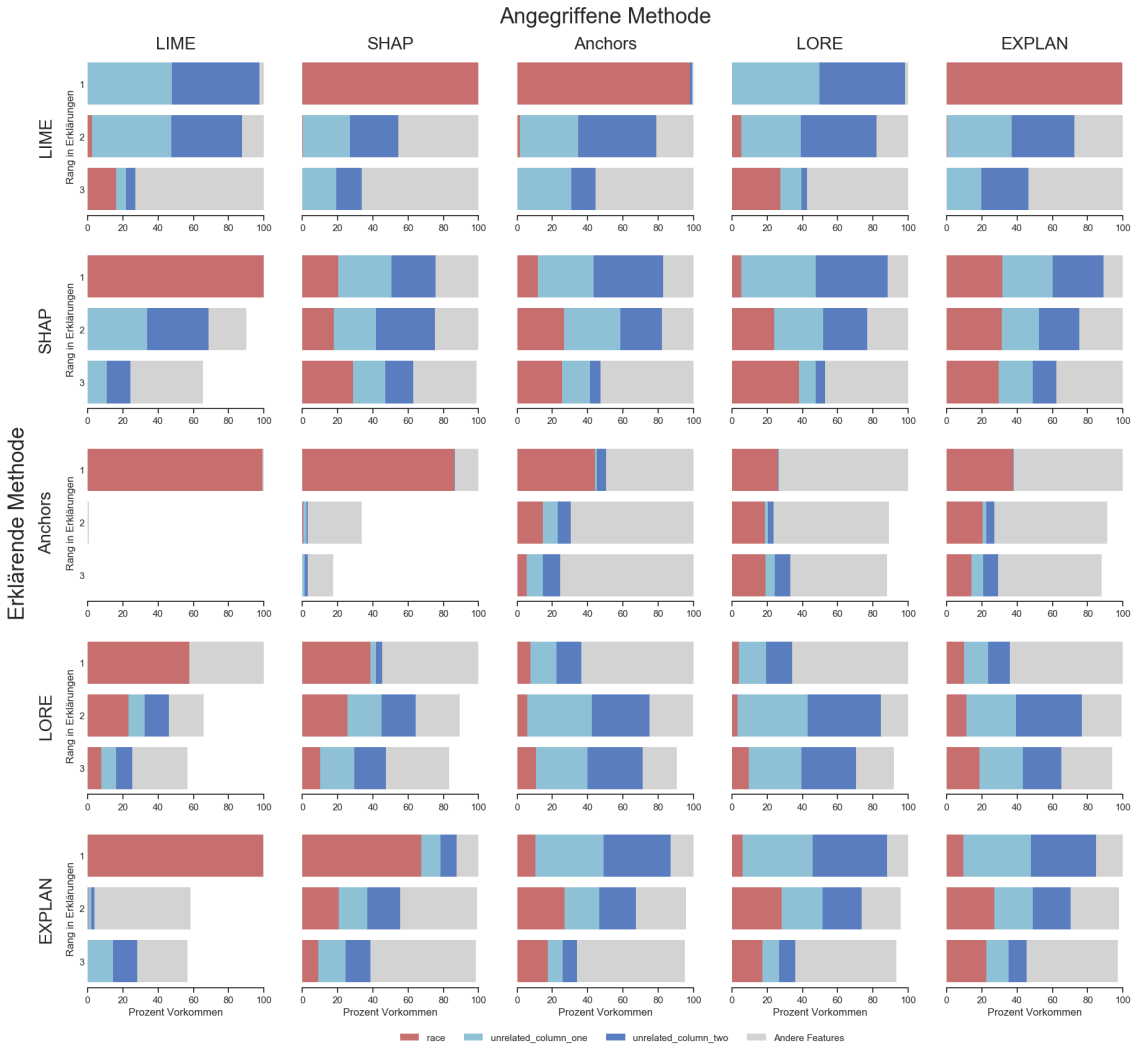


Abbildung 5.8: Erklärungsaggregationen der Mix-Versuche auf COMPAS mit zwei Features `unrelated_column_one` und `unrelated_column_two` für das Adversarial Model. Auf der Diagonale werden die passenden Ergebnisse der Einzelversuche mit abgebildet.

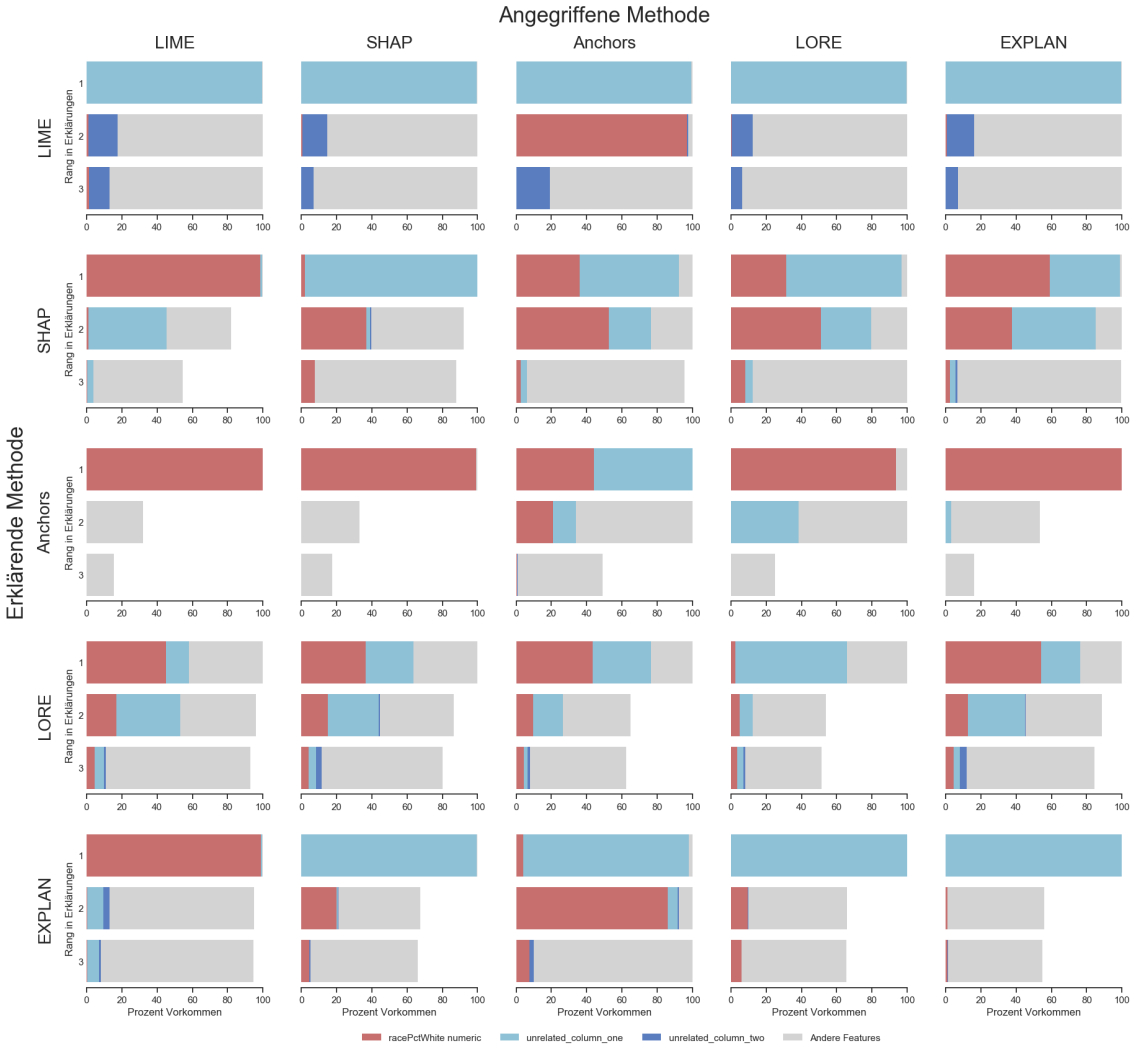


Abbildung 5.9: Erklärungsaggregationen der Mix-Versuche auf Communities and Crime mit einem Feature `unrelated_column_one` für das Adversarial Model. Auf der Diagonale werden die passenden Ergebnisse der Einzelversuche mit abgebildet.

Die beste Täuschung erreicht dort der Angriff auf KernelSHAP und die schlechteste der Angriff auf EXPLAN. EXPLAN dagegen kann nur den Angriff auf LIME durchschauen und wird von allen anderen stark getäuscht, darunter am wenigsten noch von dem Angriff auf Anchors. Aus der Tabelle 5.5 lässt sich auch entnehmen, dass die Angriffe die auf EXPLAN oder LORE abzielen und von EXPLAN erklärt werden am erfolgreichsten sind. Dahingegen sind die Angriffe auf LIME und EXPLAN die von Anchors erklärt werden am wenigsten erfolgreich.

Bei den Angriffen mit zwei Features auf dem Communities and Crime Datensatz lässt sich LIME wieder von vielen Angriffen sehr stark täuschen, allerdings ist es in diesem Fall in der Lage, dem Angriff der auf Anchors spezialisiert ist nicht zu verfallen. Dies ist dargestellt in Abbildung 5.10. Auffällig ist hierbei die Tendenz von LIME das Feature

		Angegriffene Methode				
		LIME	SHAP	Anchors	LORE	EXPLAN
Erklärer	LIME	0.01	0.01	0.01	0.01	0.01
	SHAP	7.54	0.04	0.92	0.72	1.53
	Anchors	20.72	7.54	0.75	4.83	20.72
	LORE	1.42	1.00	0.97	0.19	1.35
	EXPLAN	8.44	0.01	0.12	0	0

Tabelle 5.5: Werte der Fooled-Heuristik für die Einzelversuche und die Mix-Versuche auf Communities and Crime mit einem Feature `unrelated_column_one` für das Adversarial Model. Die Spalten entsprechen den vom Adversarial Model angegriffenen Methoden und die Zeilen den Methoden die zum Erklären verwendet werden. Alle Werte sind auf zwei Nachkommastellen gerundet.

`unrelated_column_two` deutlich zu bevorzugen, wie es bereits bei den Einzelversuchen der Fall war (siehe Abbildung 5.2).

KernelSHAP lässt sich mittelmäßig von den Angriffen auf Anchors, LORE und EXPLAN täuschen, allerdings fast gar nicht durch den Angriff auf LIME. Anchors zeigt dagegen jedoch kaum Änderungen der Erklärungen bei allen Angriffen, ist also gegen diese resistent. LORE lässt sich von den nicht auf LORE ausgerichteten Angriffen etwa gleich gut täuschen, etwas mehr durch den für EXPLAN, etwas weniger durch den für LIME. Alle diese Täuschungen sind allerdings nicht sehr stark. EXPLAN wird von den Angriffen auf KernelSHAP und LORE fast vollständig getäuscht, von dem Adversarial Model für Anchors jedoch deutlich weniger. Der Angriff auf LIME täuscht EXPLAN nicht. Die Werte der Fooled-Heuristik in Tabelle 5.6 zeigen, dass jeder Angriff, außer der auf Anchors gezielte Angriff, Anchors von den Erklärbarkeitsmethoden am wenigsten täuschen. Die beste Täuschung erreicht der Angriff auf EXPLAN auf sein Ziel.

Auf den German Credit Datensatz lässt sich LIME wieder von allen Angriffen außer dem auf KernelSHAP täuschen (siehe Abbildung 5.11). Dabei ist dieser Effekt etwas schwächer für die Angriffe auf Anchors und EXPLAN. Die Angriffe auf Anchors und LORE sind für KernelSHAP in etwa so effektiv wie der darauf ausgerichtete Angriff. Der Angriff auf EXPLAN ist etwas schwächer und der Angriff auf LORE fast gar nicht effektiv. Entsprechend des Verhaltens auf den anderen Datensätzen sind die Angriffe nicht wirksam gegen Anchors, dabei sind die Angriffe für LORE und EXPLAN noch etwas besser als die anderen. LORE lässt sich von den Angriffen auf Anchors und EXPLAN so gut täuschen wie durch einen gezielten Angriff, ist aber resistenter gegen den Angriff auf LIME und insbesondere den auf KernelSHAP. EXPLAN teilt die Resistenz von Anchors gegen den LIME-Angriff und ist recht resistent gegen den KernelSHAP-Angriff, lässt sich jedoch von den Adversarial Models für Anchors und LORE täuschen, davon besser vom ersteren. Anhand der Werte der Fooled-Heuristik aus Tabelle 5.7, lässt sich erkennen, dass die beste

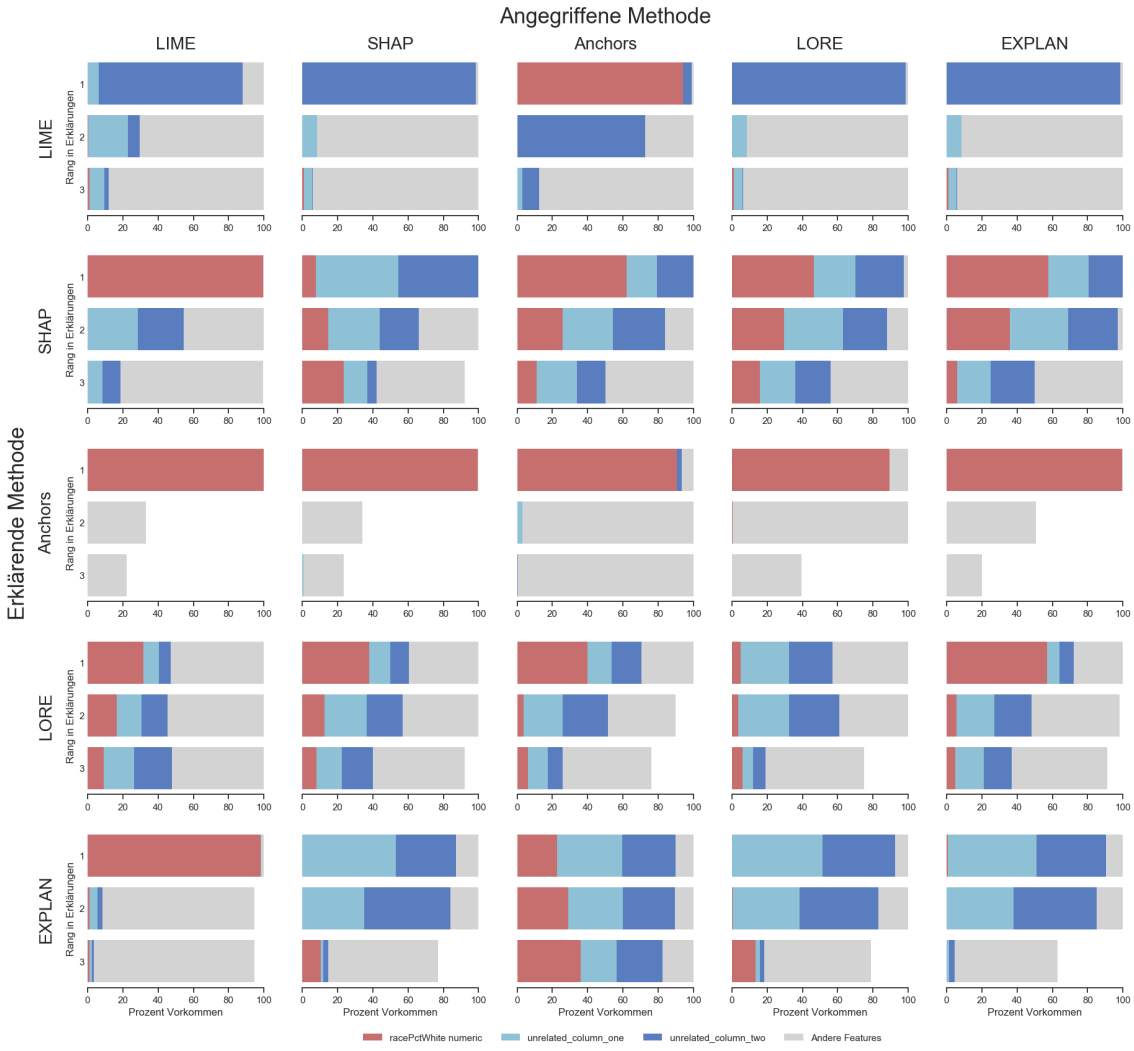


Abbildung 5.10: Erklärungsaggregationen der Mix-Versuche auf Communities and Crime mit zwei Features `unrelated_column_one` und `unrelated_column_two` für das Adversarial Model. Auf der Diagonale werden die passenden Ergebnisse der Einzelversuche mit abgebildet.

		Angegriffene Methode				
		LIME	SHAP	Anchors	LORE	EXPLAN
Erklärer	LIME	0.04	0.02	4.07	0.02	0.02
	SHAP	10.52	0.71	2.11	1.61	2.03
	Anchors	20.72	10.52	3.96	5.65	10.52
	LORE	1.53	1.46	1.19	0.38	1.81
	EXPLAN	10.52	0.26	1.29	0.32	0.01

Tabelle 5.6: Werte der Fooled-Heuristik für die Einzelversuche und die Mix-Versuche auf Communities and Crime mit zwei Features `unrelated_column_one` und `unrelated_column_two` für das Adversarial Model. Die Spalten entsprechen den vom Adversarial Model angegriffenen Methoden und die Zeilen den Methoden die zum Erklären verwendet werden. Alle Werte sind auf zwei Nachkommastellen gerundet.

		Angegriffene Methode				
		LIME	SHAP	Anchors	LORE	EXPLAN
Erklärer	LIME	0.05	20.72	0.35	0.07	0.32
	SHAP	3.17	0	0.06	0.06	0.61
	Anchors	20.72	6.00	0.53	2.08	1.86
	LORE	1.49	2.94	0.23	0.20	0.37
	EXPLAN	20.72	5.57	0.12	0.83	0.15

Tabelle 5.7: Werte der Fooled-Heuristik für die Einzelversuche und die Mix-Versuche auf German Credit mit einem Feature `LoanRateAsPercentOfIncome` für das Adversarial Model. Die Spalten entsprechen den vom Adversarial Model angegriffenen Methoden und die Zeilen den Methoden die zum Erklären verwendet werden. Alle Werte sind auf zwei Nachkommastellen gerundet.

Täuschung für den Angriff auf KernelSHAP mit KernelSHAP als Erklärer auftritt. Die geringste Täuschung tritt für den LIME-Angriff erklärt durch Anchors oder EXPLAN auf.

Die Erklärungen von Anchors und LIME für den Angriff auf EXPLAN, sowie die Erklärungen von Anchors und EXPLAN auf LORE erklären relativ viel über das nicht relevante Feature `Single`. Dies lässt sich dadurch erklären, dass dieses stark mit dem sensitiven Feature `Gender` in den Daten korreliert ist (Korrelationskoeffizient 0.73).

Die Aussage von Slack et al. [45], dass Angriffe auf LIME ineffektiv gegen KernelSHAP sind, kann bestätigt werden. Auch die Aussage, dass die umgekehrte Richtung zu verschiedenen Ergebnissen führt, trifft zu. Auf den Datensätzen COMPAS und German Credit ist LIME in der Lage dem Angriff zu widerstehen. Auf dem Datensatz Communities and Crime lässt sich LIME jedoch täuschen.

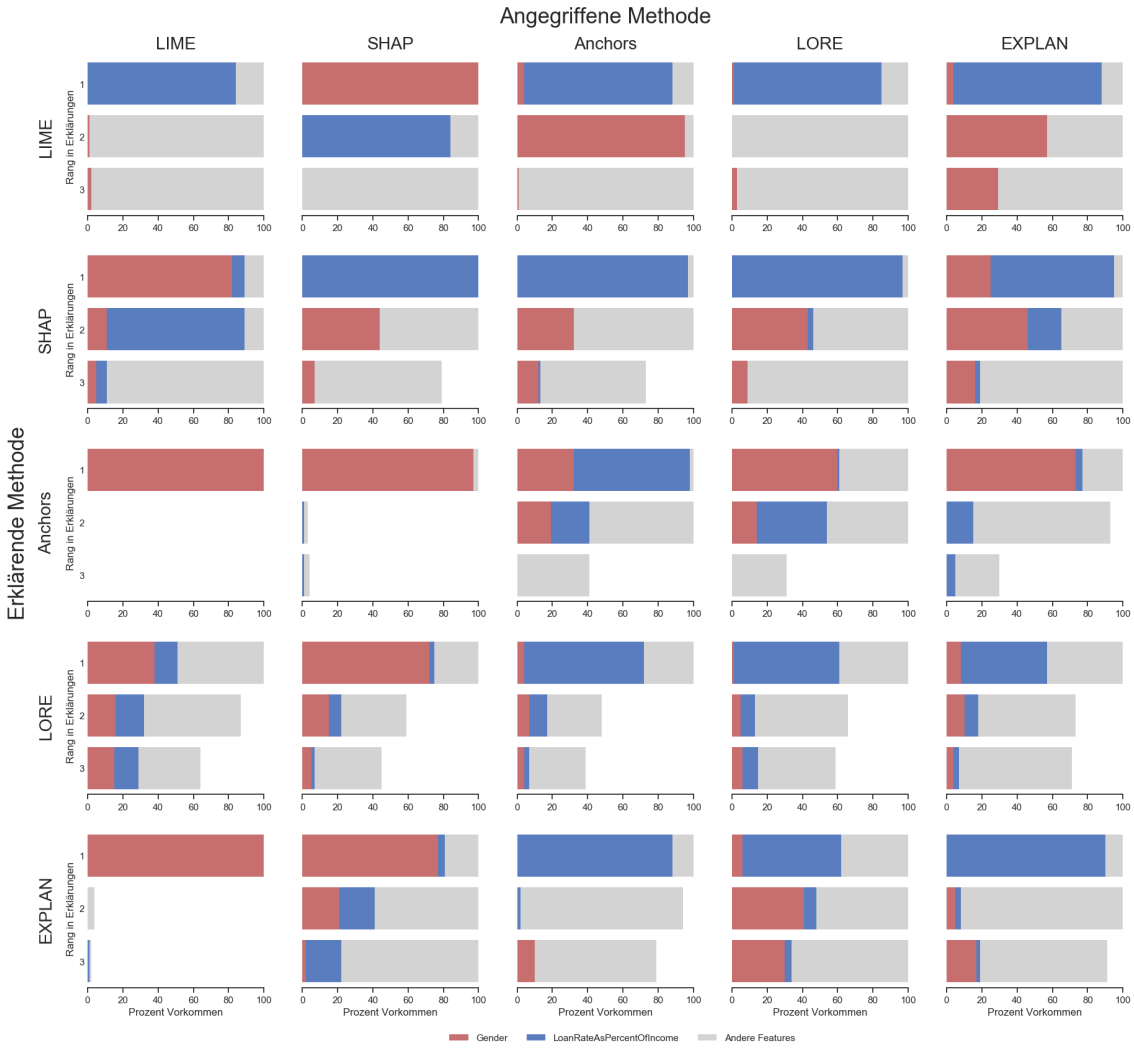


Abbildung 5.11: Erklärungsaggregationen der Mix-Versuche auf German Credit mit einem Feature `LoanRateAsPercentOfIncome` für das Adversarial Model. Auf der Diagonale werden die passenden Ergebnisse der Einzelversuche mit abgebildet.

Unter den hier neu untersuchten Erklärbarkeitsmethoden ist Anchors über alle Kombinationen gesehen der robusteste Ansatz. Er lässt sich nur schlecht von anderen Angriffen täuschen. Diese Aussage ist auch unabhängig von der Konstruktion des Angriffs, anders als die Ergebnisse der Einzelexperimente. Mit diesen zusammen ergibt sich Anchors aus den durchgeführten Experimenten klar als resistenteste Methode. Dies kann, wie bereits erwähnt, zum Teil jedoch auch an einem unpassend konstruierten Angriff liegen. Eine bessere Absicherung gegen einen Angriff erhält man durch den Einsatz mehrerer Erklärbarkeitsmethoden. Leider ist keine der Methoden als klar resistent gegen den auf Anchors ausgerichteten Angriff anzusehen. LIME gelingt es in zwei der fünf Variationen der Mix-Versuche dem Anchors-Angriff zu widerstehen, jeweils für einen Angriff mit zwei Features. Aber auch LIME scheitert für die restlichen drei Fälle. Nach LIME ist KernelSHAP noch am geeignetsten dem Angriff auf Anchors zu entgehen. Die Angriffe auf KernelSHAP und insbesondere LIME werden recht häufig von fast allen anderen Methoden erkannt, am besten jedoch meistens durch Anchors.

Kapitel 6

Fazit und Ausblick

In den Experimenten hat sich die Erklärbarkeitsmethode Anchors als am resistentesten, sowohl gegen den auf sie ausgerichteten Angriff, als auch gegen auf andere Methoden ausgerichtete Angriffe, erwiesen. Ein möglicher Grund für ersteres liegt darin, dass die Perturbationen von Anchors häufig in der Trainingsdatenmenge liegen. Dieses Ergebnis könnte allerdings auch an dem konkreten hier konstruierten Angriff liegen. Der Angriff hat allerdings keinen Einfluss auf die Resistenz gegen die Angriffe auf andere Erklärbarkeitsmethoden. Insgesamt ergibt sich Anchors als beste Wahl, wenn die Möglichkeit besteht, dass der in Abschnitt 2.3 beschriebene Angriff zum Einsatz kommt. Da jedoch auch Anchors angreifbar ist, ist es sinnvoll, mehr als eine Erklärbarkeitsmethode zu verwenden. Dabei hat sich LIME als beste Ergänzung zu Anchors herausgestellt, ist aber trotzdem nicht ausreichend, um das Angriffspotential von Anchors abzudecken. Außerdem hat sich ergeben, dass die drei Methoden Anchors, LORE und EXPLAN schon bei recht wenig genauen OOD-Klassifizierern beginnen auf den Angriff zu reagieren. In diesem Aspekt stellen sich LIME und KernelSHAP als weniger angreifbar heraus.

Die erreichten Ergebnisse sind immer abhängig von der Qualität der konstruierten Angriffe. Es ist möglich, dass effektivere Angriffe diesem Schema entsprechend gefunden werden können. Insbesondere für den Angriff auf Anchors könnte durch eine andere Wahrscheinlichkeitsfunktion für die Länge der Tupel, oder ein gänzlich überarbeitetes Konzept ein besserer Angriff entstehen. Auch die entwickelte Fooled-Heuristik hat sich nicht als ideal erwiesen, da sie in manchen Aspekten nicht ganz der Intuition entspricht und ihre Skala eine Interpretation von Differenzen erschwert.

Allgemein ist die Evaluation über Ränge der verwendeten Features in den Erklärungen recht simplifiziert. Gerade für die ursprünglich von Slack et al. [45] untersuchten Erklärbarkeitsmethoden LIME und KernelSHAP stehen konkrete Werte für die Features bereit. Diese können durchaus Einfluss auf die Wahrnehmung einer Erklärung haben, da ein Feature mit einem Gewicht von 0.01 auf Rang 2 gegenüber einem Feature mit Gewicht 0.5 auf Rang 1 eine andere Relevanz vermittelt, als ein Feature mit Gewicht 0.4 auf Rang

2 im Vergleich dazu. Außerdem lässt sich das genutzte Evaluationsschema nicht auf globale Erklärbarkeitsmethoden übertragen, welche ebenfalls anfällig für den beschriebenen Angriff sein könnten. Da die hier verwendeten neuen Erklärbarkeitsmethoden Regeln und kein direktes Ranking der Features erzeugen, musste damit außerdem umgegangen werden, indem künstlich ein solches Ranking aus ihren Ergebnisformen erzeugt wird. Diese Umwandlungsart kann auch einen Einfluss auf die Auswertung der Angriffe haben.

Da die Schwachstelle die der Angriff ausnutzt das Erzeugen von unrealistischen Perturbationen ist, kann eine Defensive gegen den Angriff darin bestehen, eine bessere Nachbarschaftserzeugung zu konstruieren, welche dieses Problem reduziert oder vollständig vermeidet. Einige Ansätze in diese Richtung, welche die Anfälligkeit für den Angriff möglicherweise verringern, wurden bereits entwickelt [1, 23, 43, 48]. Ein anderer Ansatz besteht darin zu erkennen wann ein Angriff stattgefunden hat. Dabei könnte es Unterschiede in der Verteilung der Zielwerte zwischen echten Daten und Perturbationsdaten geben.

Eine weitere Einschränkung des Angriffs in seiner aktuellen Form ist, dass er für eine Erklärbarkeitsmethode spezialisiert wird und häufig nicht für andere Erklärbarkeitsmethoden funktioniert, wie aus den Ergebnissen der Mix-Versuche erkenntlich ist. Eine Richtung in die der Angriff verbessert werden könnte ist also die Spezialisierung für mehr als eine Erklärbarkeitsmethode.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Aas, Kjersti, Martin Jullum und Anders Løland: *Explaining individual predictions when features are dependent: More accurate approximations to Shapley values.* Artificial Intelligence, 298:103502, 2021.

[2] Anders, Christopher J., Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller und Pan Kessel: *Fairwashing Explanations with Off-Manifold Detergent.* In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Band 119 der Reihe *Proceedings of Machine Learning Research*, Seiten 314–323. PMLR, 2020.

[3] Arrieta, Alejandro Barredo, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins et al.: *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI.* Information Fusion, 58:82–115, 2020.

[4] Baehrens, David, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen und Klaus-Robert Müller: *How to Explain Individual Classification Decisions.* Journal of Machine Learning Research, 11:1803–1831, August 2010.

[5] Baniecki, Hubert, Wojciech Kretowicz und Przemyslaw Biecek: *Fooling Partial Dependence via Data Poisoning.* CoRR, abs/2105.12837, 2021.

[6] Biggio, Battista und Fabio Roli: *Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning.* Pattern Recognition, 84:317–331, 2018.

[7] Bishop, Christopher M.: *Pattern Recognition and Machine Learning.* Springer, 2006.

[8] Breiman, Leo: *Bagging Predictors.* Machine Learning, 24(2):123–140, August 1996.

[9] Breiman, Leo: *Random Forests.* Machine Learning, 45(1):5–32, Oktober 2001.

[10] Breiman, Leo, Jerome H. Friedman, Richard A. Olshen und Charles J. Stone: *Classification And Regression Trees.* 1984.

[11] Burkart, Nadia und Marco F. Huber: *A Survey on the Explainability of Supervised Machine Learning.* Journal of Artificial Intelligence Research, 70:245–317, Januar 2021.

[12] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall und W. Philip Kegelmeyer: *SMOTE: Synthetic Minority over-Sampling Technique.* Journal of Artificial Intelligence Research, 16(1):321–357, Juni 2002.

[13] Day, William H. E. und Herbert Edelsbrunner: *Efficient Algorithms for Agglomerative Hierarchical Clustering Methods.* Journal of classification, 1(1):7–24, 1984.

[14] Dimanov, Botty, Umang Bhatt, Mateja Jamnik und Adrian Weller: *You Shouldn't Trust Me: Learning Models Which Conceal Unfairness From Multiple Explanation Methods.* In: *SafeAI@AAAI*, 2020.

[15] Dombrowski, Ann-Kathrin, Maximilian Alber, Christopher J. Anders, Marcel Ackermann, Klaus-Robert Müller und Pan Kessel: *Explanations Can Be Manipulated and Geometry is to Blame.* Curran Associates Inc., Red Hook, NY, USA, 2019.

[16] Dua, Dheeru und Casey Graff: *UCI Machine Learning Repository*, 2017.

[17] Guidotti, Riccardo, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini und Fosca Giannotti: *Local Rule-Based Explanations of Black Box Decision Systems.* arXiv preprint arXiv:1805.10820, 2018.

[18] Guidotti, Riccardo, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti und Dino Pedreschi: *A Survey of Methods for Explaining Black Box Models.* ACM Computing Surveys, 51(5), August 2018.

[19] Hastie, Trevor, Robert Tibshirani und Jerome Friedman: *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2 Auflage, 2009.

[20] Heo, Juyeon, Sunghwan Joo und Taesup Moon: *Fooling Neural Network Interpretations via Adversarial Model Manipulation.* In: Wallach, Hanna M., Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox und Roman Garnett (Herausgeber): *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Seiten 2921–2932, 2019.

[21] Hofmann, Hans: *Statlog (German Credit Data).* UCI Machine Learning Repository, 1994.

[22] Huang, Xiaowei, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu und Xinping Yi: *A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability.* Computer Science Review, 37:100270, 2020.

[23] Jia, Yunzhe, James Bailey, Kotagiri Ramamohanarao, Christopher Leckie und Michael E. Houle: *Improving the Quality of Explanations with Local Embedding Perturbations.* In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, Seite 875–884, New York, NY, USA, 2019. Association for Computing Machinery.

[24] Julia Angwin, Jeff Larson, Surya Mattu und Lauren Kirchner: *Machine Bias.* ProPublica, 2016.

[25] Kullback, S. und R. A. Leibler: *On Information and Sufficiency.* The Annals of Mathematical Statistics, 22(1):79 – 86, 1951.

[26] Lakkaraju, Himabindu und Osbert Bastani: *"How Do I Fool You?": Manipulating User Trust via Misleading Black Box Explanations.* In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, Seite 79–85, New York, NY, USA, 2020. Association for Computing Machinery.

[27] Lance, G. N. und W. T. Williams: *A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems.* The Computer Journal, 9(4):373–380, 02 1967.

[28] Louppe, Gilles, Louis Wehenkel, Antonio Sutera und Pierre Geurts: *Understanding Variable Importances in Forests of Randomized Trees.* In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, Seite 431–439, Red Hook, NY, USA, 2013. Curran Associates Inc.

[29] Lundberg, Scott M. und Su-In Lee: *A Unified Approach to Interpreting Model Predictions.* In: *Advances in Neural Information Processing Systems*, Seiten 4765–4774, 2017.

[30] Molnar, Christoph: *Interpretable Machine Learning.* 2019. `https://christophm.github.io/interpretable-ml-book/`.

[31] Murtagh, F.: *A Survey of Recent Advances in Hierarchical Clustering Algorithms.* The Computer Journal, 26(4):354–359, 11 1983.

[32] Murtagh, Fionn und Pierre Legendre: *Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?* Journal of Classification, 31(3):274–295, Oktober 2014.

[33] Pastor, Eliana und Elena Baralis: *Explaining Black Box Models by Means of Local Rules.* In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, Seite 510–517, New York, NY, USA, 2019. Association for Computing Machinery.

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

[35] Quinlan, J. Ross: *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., 1993.

[36] Rasouli, Peyman und Ingrid Chieh Yu: *Meaningful Data Sampling for a Faithful Local Explanation Method.* In: Yin, Hujun, David Camacho, Peter Tino, Antonio J. Tallón-Ballesteros, Ronaldo Menezes und Richard Allmendinger (Herausgeber): *Intelligent Data Engineering and Automated Learning – IDEAL 2019*, Seiten 28–38, Cham, 2019. Springer International Publishing.

[37] Rasouli, Peyman und Ingrid Chieh Yu: *EXPLAN: Explaining Black-box Classifiers using Adaptive Neighborhood Generation.* In: *2020 International Joint Conference on Neural Networks (IJCNN)*, Seiten 1–9, 2020.

[38] Redmond, Michael: *Communities and Crime Unnormalized Data Set.* UCI Machine Learning Repository, 2011.

[39] Ribeiro, Marco Tulio, Sameer Singh und Carlos Guestrin: *"Why Should I Trust You?": Explaining the Predictions of Any Classifier.* In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, Seite 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[40] Ribeiro, Marco Tulio, Sameer Singh und Carlos Guestrin: *Anchors: High-Precision Model-Agnostic Explanations.* In: *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[41] Ruggieri, Salvatore: *YaDT: Yet another Decision Tree builder.* In: *16th International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, Seiten 260–265. IEEE Press, 2004.

[42] Saabas, Ando: *Interpreting random forests.* `http://blog.datadive.net/interpreting-random-forests/`, 2014. Letzter Zugriff: 15.09.2021.

[43] Saito, Sean, Eugene Chua, Nicholas Capel und Rocco Hu: *Improving LIME Robustness with Smarter Locality Sampling.* CoRR, abs/2006.12302, 2020.

[44] Shapley, Lloyd S.: *A Value for n-Person Games.* In: *Contributions to the Theory of Games 2.28*, Seiten 307–318. 1953.

[45] Slack, Dylan, 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*, AIES '20, Seite 180–186, New York, NY, USA, 2020. Association for Computing Machinery.

[46] Tan, Pang-Ning, Michael Steinbach, Vipin Kumar und Anuj Karpatne: *Introduction to Data Mining eBook: Global Edition* . Pearson Deutschland, 2019.

[47] Ustun, Berk, Alexander Spangher und Yang Liu: *Actionable Recourse in Linear Classification.* In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, Seite 10–19, New York, NY, USA, 2019. Association for Computing Machinery.

[48] Vres, Domen und Marko Robnik-Sikonja: *Better sampling in explanation methods can prevent dieselgate-like deception.* CoRR, abs/2101.11702, 2021.

[49] Wachter, Sandra, Brent D. Mittelstadt und Chris Russell: *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR.* CoRR, abs/1711.00399, 2017.

[50] Ward, Joe H.: *Hierarchical Grouping to Optimize an Objective Function.* Journal of the American Statistical Association, 58(301):236–244, 1963.

[51] Wicker, Matthew, Xiaowei Huang und Marta Kwiatkowska: *Feature-Guided Black-Box Safety Testing of Deep Neural Networks.* In: Beyer, Dirk und Marieke Huisman (Herausgeber): *Tools and Algorithms for the Construction and Analysis of Systems*, Seiten 408–426, Cham, 2018. Springer International Publishing.