Informatik
Hauptcampus

HOCHSCHULE TRIER

Erklärung von Klassifikatoren des maschinellen Lernens durch vielfältige kontrafaktische Erklärungen

Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations

Fabian Wagner

Seminar

Betreuer: Prof. Dr. Jörn Schneider, Marvin Schneider

Trier, 11.02.2026

Kurzfassung

Inhaltsverzeichnis

1 Einleitung und Problemstellung 1

2 Theoretische Grundlagen 2
2.1 Grundlegende Begriffe 2
2.1.1 Formaler Rahmen der Klassifikation 2
2.1.2 Kontrafaktische Erklärungen 3
2.1.3 Einordnung von Erklärungsmodellen 3
2.2 Definition von DiCE 3

3 Methodisches Vorgehen 4
3.1 Zentrale Konzepte 4
3.2 Optimierung der Counterfactuals 6

4 Evaluation der Methodik 8
4.1 Evaluationsmetriken 8
4.2 Bewertung der Evaluationsqualität 10
4.2.1 Bewertung der Evaluationsmetriken 10
4.2.2 Näherung der Entscheidungsgrenze 10
4.3 Vergleich mit anderen Ansätzen 11
4.3.1 Baselines und LIME 11
4.3.2 Ergebnisse 12

5 Anwendbarkeit und Bedeutung in der Praxis 13
5.1 Relevanz in der Praxis 13
5.2 Demonstration eines Beispiels 13
5.2.1 Initialisierungsschritte 14
5.2.2 Generierung von Erklärungen 15

6 Zusammenfassung und Ausblick 18

Literaturverzeichnis 19

Eigenständigkeitserklärung 20

1

Einleitung und Problemstellung

Künstliche Intelligenz etabliert sich zu einem festen Bestandteil der Arbeitswelt und dem Alltag. Neben dem Einsatz als Chatbot oder Assisstenten treffen KI-Systeme vermehrt Entscheidungen, welche folgenschwere Konsequenzen auf das Leben haben. Die Nachvollziehbarkeit einer solchen Entscheidung ist für den sicheren und fairen Einsatz von KIs von zentraler Bedeutung. Erklärbare KI (xAI) befasst sich mit der Fragestellung die getroffene Klassifikation einer KI für den Menschen verständlich und nachvollziehbar zu machen.

Ziel der Seminararbeit ist die Analyse und Evaluation der Methodik von DiCE [MST20] sowie die Bewertung für die Anwendbarkeit der Methode in der Praxis. Der Ansatz von DiCE beruht auf der Erklärung durch Beispiele, die ähnlich zur ursprünglichen Eingabe sind, jedoch nicht zur gleichen Klassifikation durch die KI führen. Als erstes werden grundlegende Begriffe erklärt, welche für das Verständnis der Arbeit notwendig sind. Anschließend wird das methodische Vorgehen von DiCE untersucht. Hierzu werden die Konzepte Diversity, Proximity, Sparsity, User Constraints sowie Feasibility erläutert und deren Abhängigkeiten betrachtet. Die Bewertung von DiCE erfolgt mithilfe von Evaluationsmetriken für die einzelnen Konzepte, welche zuvor definiert werden. Weiterhin wird ein Beispiel demonstriert, an dem die Verständlichkeit einer solchen Erklärung durch Counterfactuals verdeutlicht werden soll. Abschließend folgt eine Zusammenfassung und kritische Diskussion der vorgestellten Methodik.

2

Theoretische Grundlagen

In diesem Kapitel werden zunächst wichtige Begriffe aus dem Bereich der erklärbaren KI vorgestellt und in dem Kontext von DiCE erläutert.

2.1 Grundlegende Begriffe

Das Framework DiCE wird primär im Bereich der überwachten Klassifikation eingesetzt, weshalb der Fokus der folgenden Definitionen auf diesem Teilbereich des maschinellen Lernens liegt.

2.1.1 Formaler Rahmen der Klassifikation

Im Kontext dieser Arbeit wird ein **Modell des maschinellen Lernens** als eine Funktion betrachtet, die Eingabedaten einer bestimmten Kategorie zuordnet. Die Eingabe erfolgt in Form eines **Feature-Vektors**, welcher die numerischen und kategorischen Merkmale eines Objekts zusammenfasst. Das Ergebnis der Verarbeitung ist die Zuordnung zu einer diskreten **Klasse**, wie in Abbildung 2.1 exemplarisch gezeigt wird. Während ML-Modelle auch für kontinuierliche Vorhersagen genutzt werden können, konzentriert sich die Generierung von Counterfactuals auf die Veränderung der Klassenzugehörigkeit.

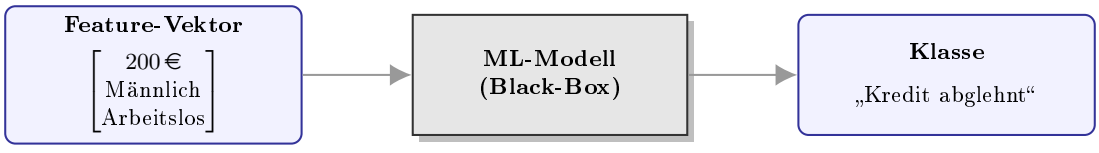

Abbildung 2.1: Visualisierung der Datenverarbeitung: Eine ursprüngliche Eingabe als Feature-Vektor wird durch das ML-Modell einer Klasse zugeordnet. Das ML-Modell lehnt den Antrag auf einen Kredit von einer männlichen, arbeitslosen Person mit einem Einkommen von 200€ ab.

2.1.2 Kontrafaktische Erklärungen

Eine **kontrafaktische Erklärung**, auch Counterfactual (CF) genannt, setzt dort an, wo die ursprüngliche Eingabe zu einer aus Anwendersicht unerwünschten Klasse führt. Ein Counterfactual ist ein modifiziertes Gegenbeispiel zur ursprünglichen Eingabe, welches so gewählt wird, dass das Modell eine **erwünschte Klasse** (CF-Klasse) ausgibt. Das Ziel ist es, dem Anwender die minimal notwendigen Änderungen aufzuzeigen, um ein gewünschtes Ergebnis zu erzielen.[MST20]

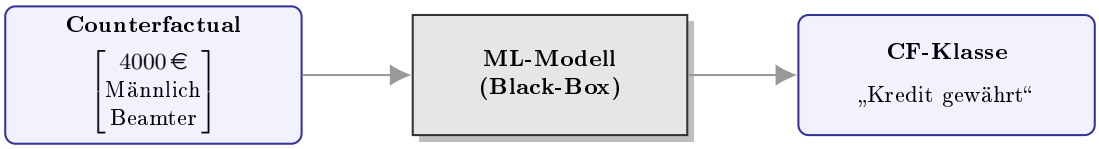

Abbildung 2.2: Beispielhafte kontrafaktische Erklärung im Bezug auf Abbildung 2.1, sodass durch Variation der Eingabe die gewünschte Klasse erreicht wird. Das Counterfacutal zeigt, dass eine Erhöhung des Einkommens und eine Anstellung den gewünschten Kredit durch das Modell ermöglichen würde.

2.1.3 Einordnung von Erklärungsmodellen

Die Notwendigkeit für Verfahren wie DiCE ergibt sich aus der Komplexität moderner **Black-Box**-Modelle. Im Gegensatz zu **White-Box**-Modellen ist deren interne Logik nicht unmittelbar nachvollziehbar[LG19]. Erklärungsansätze lassen sich in zwei Kategorien, global und lokal, unterteilen. **Globale** Ansätze versuchen, das Modellverhalten in seiner Gesamtheit zu interpretieren, während **lokale** Ansätze, zu denen auch die kontrafaktischen Erklärungen zählen, die Entscheidung für einen spezifischen Einzelfall begründen[BATGL+19].

2.2 Definition von DiCE

Diverse **C**ounterfactual **E**xplanations (DiCE) ist ein Erklärungsansatz, um Entscheidungen von ML-Modellen für den Menschen verständlich zu machen. Es handelt sich um ein lokales, post-hoc Verfahren für Black-Box-Modelle. Die Erklärung des Modells findet somit nach der Trainingsphase statt und dient lediglich der Bewertung, Nachvollziehbarkeit sowie dem Aufzeigen von Schwächen oder einem Bias in den Ausgaben des Modells. DiCE generiert zu einer Eingabe verschiedene Counterfactuals, um dem Anwender aufzuzeigen, welche Parameter sich für eine andere Klassifikation durch das ML-Modell ändern müssen. Die Counterfactuals werden so generiert, dass sie möglichst unterschiedlich sind, dabei jedoch möglichst nahe an der ursprünglichen Eingabe bleiben.[MST20]

3

Methodisches Vorgehen

Nachdem die theoretischen Grundlagen definiert wurden, befasst sich dieses Kapitel mit dem methodischen Vorgehen von DiCE.

Die Kernidee lässt sich am besten als ein Suchprozess beschreiben: Ausgehend von einer ursprünglichen Eingabe, die zu einem unerwünschten Ergebnis geführt hat, sucht das Verfahren nach einer optimalen Menge von Alternativszenarien. Anstatt dem Anwender lediglich zu sagen, dass er abgelehnt wurde, generiert DiCE eine Erklärung, was hätte anders sein müssen, um das gewünschte Resultat zu erhalten.

Ein anschauliches Beispiel hierfür ist die Ablehnung eines Kreditantrags durch ein ML-Modell. DiCE kann hier aufzeigen, dass der Kredit bewilligt worden wäre, wenn beispielsweise das Einkommen um 100€ höher oder die Kreditsumme um 5000€ niedriger ausgefallen wäre.

Der methodische Ablauf von DiCE lässt sich in drei Phasen unterteilen:

1. **Initialisierung:** Zufällige Initialisierung der Counterfactuals.
2. **Optimierung:** Optimierung der Counterfactuals durch iterative Minimierung der Verlustfunktion.
3. **Nachbearbeitung:** Anpassung der Ergebnisse zur Steigerung der Sparsität.

In den folgenden Abschnitten werden zunächst die zentralen Konzepte erläutert, bevor die mathematische Formulierung der Verlustfunktion zur Optimierung der Counterfactuals untersucht wird.

3.1 Zentrale Konzepte

Diversität beschreibt, wie sich die generierten Counterfactuals voneinander unterscheiden. Eine hohe Vielfältigkeit zeigt dem Anwender nicht nur mehrere Möglichkeiten zum Erreichen einer anderen Klassifikation auf, wodurch sich die Machbarkeit erhöht, sondern lässt auch größere Rückschlüsse auf das Entscheidungsverhalten des ML-Modells zu. Machbarkeit bedeutet im Kontext von DiCE, ob eine Möglichkeit für einen spezifischen Anwender theoretisch realisierbar ist. Zum Beispiel kann ein Mensch seine ethnische Herkunft nicht ändern, jedoch ist das Erlernen einer neuen Sprache möglich. Um Diversität zu berücksichtigen, wird in DiCE das Konzept der **D**eterminantal **P**oint **P**rocesses (DPP) verwendet, um

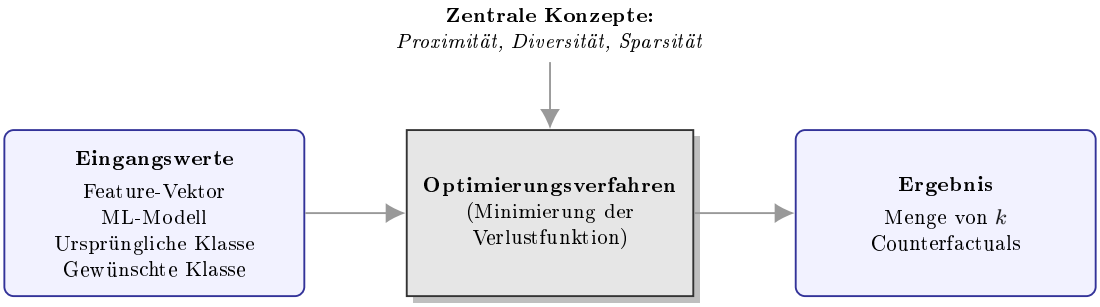

Abbildung 3.1: Methodischer Überblick der Counterfactual-Generierung: Basierend auf der ursprünglichen Eingabe und dem Modell sucht DiCE nach neuen Beispielen, welche die zentralen Konzepte bestmöglich erfüllen.

das *Subset Selection Problem* zu lösen. Das Problem beschreibt dabei die Auswahl von wenigen CFs aus einer unendlich großen Menge an möglichen Beispielen, welche zeitgleich gültig als auch divers sind. In Gleichung 3.1 beschreibt $dist(c_i, c_j)$ die Distanz zwischen zwei Counterfactuals. Somit führt eine kleine Ähnlichkeit $K_{i,j}$ der CFs zu einer großen Determinante $det(K)$ und Maximierung der Diversität.[MST20, KT+12]

$$dpp\_diversity = det(K), \text{ mit } K_{i,j} = \frac{1}{1 + dist(c_i, c_j)} \tag{3.1}$$

Diversität alleine ist nicht ausreichend, um einem Anwender eine Erklärung zu geben. Die generierten CFs sollten nicht nur unterschiedlich sein, sondern müssen möglichst nah an der ursprünglichen Eingabe sein. Diese **Proximität** ist für die Machbarkeit von zentraler Bedeutung, da Anwender den meisten Nutzen aus ähnlichen Counterfactuals erhalten. Die Proximität eines CFs ergibt sich aus der negative Distanz zwischen dem Counterfactual c_i und dem Eingabevektor x. Eine geringe Distanz resultiert in einer hohen Proximität. Die Berechnung der mittleren Proximität einer Menge von CFs ist in Gleichung 3.2 dargestellt.[MST20]

$$Proximity = -\frac{1}{k} \sum_{i=1}^{k} dist(c_i, x) \tag{3.2}$$

Eine weitere Eigenschaft für die Machbarkeit oder auch Umsetzbarkeit der kontrafaktischen Beispiele ist die **Sparsität**. Nach der Proximität ist auch ein Counterfactual nahe an einer Eingabe, wenn jeder Vektoreintrag minimal geändert wird. Dies ist zwar mathematisch korrekt, vernachlässigt aber den Umstand der Machbarkeit für einen Anwender. Ein Counterfactual ist einfacher umzusetzen, wenn sich möglichst wenige Eigenschaften ändern. Sparsität wird über die Anzahl an unterschiedlichen Eigenschaften zwischen Eingabe und Counterfactual gemessen.[MST20]

Weiterhin ist zu beachten, dass zwar ein Counterfactual ähnlich zu dem ursprünglichen Feature-Vektor sein kann, jedoch für den Anwender nicht umsetzbar

ist. Aus diesem Grund muss es die Möglichkeit geben, den Wertebereich von Eigenschaften einzuschränken und zum anderen die Änderung von Eigenschaften vollständig zu verhindern.[MST20]

3.2 Optimierung der Counterfactuals

DiCE hat das Ziel eine Menge an k Counterfactuals für einen Eingabevektor x zu finden. Um die optimale Menge $C(x)$ zu ermitteln, wird die **Verlustfunktion** in Gleichung 3.3 verwendet und nach den Argumenten gesucht, welche die Gesamtfunktion minimieren. Die Optimierung wird über ein Gradientenabstiegsverfahren realisiert, wobei die k Counterfactuals zufällig initialisiert werden.[MST20]

$$C(x) = \arg\min_{c_1,\dots,c_k} \underbrace{\frac{1}{k}\sum_{i=1}^{k} yloss(f(c_i), y)}_{\text{Term 1: Gültigkeit}} + \underbrace{\frac{\lambda_1}{k}\sum_{i=1}^{k} dist(c_i, x)}_{\text{Term 2: Proximität}} - \underbrace{\lambda_2 \cdot dpp\_diversity(c_1, ..., c_k)}_{\text{Term 3: Diversität}} \tag{3.3}$$

Der erste Term beschreibt den mittleren Gültigkeitsfehler, wobei $yloss(.,.)$ die Distanz zwischen der Vorhersage $f(c_i)$ des ML-Modells für das Counterfactual c_i und dem gewünschten Ergebnis y misst. Die Minimierung dieses Terms führt dazu, dass die generierten Counterfactuals auch valide sind und in der gewünschten Klasse liegen. Für die Funktion $yloss$ wird eine Hinge-Loss-Funktion verwendet, da ein Counterfactual nur in der gewünschten Klasse liegen muss und nicht möglichst nah am Ziel y liegen muss. [MST20]

Die Minimierung des zweiten Terms maximiert die Proximität der Counterfactuals c_i, sodass diese möglichst ähnlich zum ursprünglichen Eingabevektor x sind. Über den Parameter λ_1 kann der Einfluss der Proximität auf das Gesamtergebnis variiert werden. Ein großes λ_1 erhöht die Wichtigkeit der Proximität erhöht, sodass die generierten Counterfactuals ähnlicher zu der Eingabe sind. Kategorische Features verwenden eine 0/1-Funktion, sodass in einem Counterfactual geänderte Features mit 1 und unveränderte Features mit 0 gekennzeichnet werden. Kontinuierliche Features verwenden eine l_1-Distanz, welche durch den Median der absoluten Abweichungen (MAD) dividiert wird. Dies skaliert die Distanz, um Änderungen fair zu gewichten. [MST20]

Um die Gesamtfunktion zu minimieren, muss der dritte Term maximiert werden. Die Gewichtung der Diversität wird mithilfe von λ_2 festgelegt, wobei eine Erhöhung des Parameters die Unterschiede zwischen den Counterfactuals erhöht. [MST20]

Die Verlustfunktion bestimmt eine optimale Menge an Counterfactuals bezüglich

Gültigkeit, Proximität und Diversität, vernachlässigt jedoch die Sparsität. Aus diesem Grund erfolgt eine Nachbearbeitung der CFs, um die Sparsität zu erhöhen. Dazu wird ein Greedy-Algorithmus verwendet, welcher jedes Counterfactuals einzeln betrachtet. Der Algorithmus versucht jedes kontinuierliche Feature des CFs c_i auf den Wert des ursprünglichen Eingabevektors x zurückzusetzen, ohne dass sich die Vorhersage des ML-Modells $f(c_i)$ ändert. Dabei wird das Zurücksetzen zuerst bei Features mit geringen Änderungen versucht.[MST20]

4

Evaluation der Methodik

In diesem Kapitel werden zunächst die Evaluationsmetriken für Gültigkeit, Proximität, Diversität und Sparsität definiert, um generierte Erklärungen bewerten zu können. Anschließend werden die Metriken bewertet und das Konzept der sogenannten Entscheidungsgrenze als das angestrebte Ziel betrachtet.

4.1 Evaluationsmetriken

Die Gültigkeitsmetrik in Gleichung 4.1 beschreibt den Anteil $\%ValidCFs$ an eindeutigen Counterfactuals aus der Menge C aller generierten Erklärungen, wobei k die Anzahl an geforderten CFs ist. Gültigkeit ist ein wichtiges Maß, da DiCE nicht auf die Einzigartigkeit der Erklärungen prüft.[MST20]

$$\%ValidCFs = \frac{|\{\text{unique instance in } C \text{s.t. } f(c) > 0.5\}|}{k} \tag{4.1}$$

Die Proximitätsmetrik bewertet, wie nah die Counterfactuals c_i an der ursprünglichen Eingabe x im Feature-Raum sind. Die kontinuierlichen und kategorialen Features werden getrennt evaluiert. In Gleichung 4.2 werden zunächst die Distanzen $dist\_cont$ der kontinuierlichen Features der einzelnen Counterfactuals im Bezug auf die ursprüngliche Eingabe berechnet. Anschließend wird der Mittelwert dieser Distanzen über alle k generierten Counterfactuals gebildet. Abschließend wird der Wert negativ dargestellt, um die Metrik von der Distanz in einen Proximitätswert umzuwandeln, sodass eine kleine Distanz in einer großen Nähe resultiert.[MST20]

$$ContinuousProximity := -\frac{1}{k}\sum_{i=1}^{k} dist\_cont(c_i, x) \tag{4.2}$$

In Gleichung 4.3 wird zunächst die Distanz $dist\_cat$ der kategorialen Features der einzelnen Counterfactuals im Bezug auf die ursprüngliche Eingabe berechnet, welche zählt, wie viele kategoriale Features sich geändert haben. Anschließend wird der Mittelwert der Distanzen über alle k generierten Counterfactuals gebildet. Um den Anteil der kategorialen Features zu repräsentieren, welche nicht geändert

wurden, wird in einem letzten Schritt die Distanz von Eins subtrahiert. Eine hohe kategoriale Proximität resultiert somit in einem Wert nahe Eins, da nur wenige kategorialen Features geändert wurden.[MST20]

$$CategoricalProximity := 1 - \frac{1}{k}\sum_{i=1}^{k} dist\_cat(c_i, x) \tag{4.3}$$

Sparsität ist eine Metrik, um die Machbarkeit der generierten Erklärungen zu bewerten. Dieser Wert ist für kategoriale Features identisch mit der Proximität. Einfachheitshalber werden kontinuierliche und kategoriale Merkmale in der Sparsität zu einer einzigen Metrik in Gleichung 4.4 zusammengefasst. Für jedes Counterfactual c_i wird die Anzahl an veränderten Features im Vergleich zur ursprünglichen Eingabe x gezählt. Jedes der k Counterfactuals besitzt d Merkmale. Anschließend wird der Mittelwert über alle Counterfactuals und Features berechnet und von Eins subtrahiert. Ein hoher Sparsitätswert nahe Eins bedeutet, dass nur wenige Features in den Erklärungen verändert wurden und ist für eine Umsetzbarkeit der Erklärungen anzustreben.[MST20]

$$Sparsity := 1 - \frac{1}{kd}\sum_{i=1}^{k}\sum_{l=1}^{d} 1_{[c_i^l \neq x_i^l]} \tag{4.4}$$

Die Diversitätsmetrik wird analog zur Proximität definiert und in kontinuierliche und kategoriale Diversität aufgeteilt, wie in den Gleichungen 4.5 und 4.6 dargestellt ist. Der Unterschied liegt darin, dass nun die Distanz der Features zwischen zwei Counterfactuals c_i und c_j gemessen wird. Die Anzahl aller möglichen CF-Paare ist C_k^2. Im Anschluss an die Berechnung der kategorialen und kontinuierlichen Distanz wird jeweils der Mittelwert über die Distanzen gebildet. Ein hoher Wert bedeutet, dass eine große Diversität zwischen den einzelnen Erklärungen vorliegt. Dies resultiert in größeren Unterschieden zwischen den Counterfactuals, wodurch die Umsetzbarkeit für den Anwender wahrscheinlicher wird. [MST20]

$$ContinuousDiversity := \Delta = \frac{1}{C_k^2}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} dist\_cont(c_i, c_j) \tag{4.5}$$

$$CategoricalDiversity := \Delta = \frac{1}{C_k^2}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} dist\_cat(c_i, c_j) \tag{4.6}$$

Analog zu der Sparsität kann eine weitere Diversitätsmetrik (Gleichung 4.7) formuliert werden, welche den Durchschnitt aller unterschiedlichen Features beim paarweisen Vergleich der Counterfactuals angibt. [MST20]

$$CountDiversity := \Delta = \frac{1}{C_k^2 d}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\sum_{l=1}^{d} 1_{[c_i^l \neq c_j^l]} \tag{4.7}$$

4.2 Bewertung der Evaluationsqualität

Nachdem im vorherigen Abschnitt die Evaluationsmetriken definiert wurden, muss nun deren Güte überprüft werden. Die Funktion der Metriken ist die Bewertung einer Menge von generierten Erklärungen im Hinblick auf Gültigkeit, Proximität, Sparsität und Diversität.

4.2.1 Bewertung der Evaluationsmetriken

Die Gültigkeitsmetrik stellt sicher, dass die Counterfactuals tatsächlich einzigartige und passende Erklärungen für das vorliegende Problem sind. Eine Erklärung, die nicht korrekt in die gewünschte Zielklasse klassifiziert wird, ist wertlos. Dies gilt auch für mehrfache Generierungen derselben Erklärung, da keine neuen Erkenntnisse gewonnen werden können.[MST20]

Um die Nähe zwischen der Eingabe und den Counterfactuals zu bewerten, sind die Proximitäts- und Sparsitätsmetriken notwendig. Sie sind zentraler Bestandteil, um eine Aussage über die Umsetzbarkeit von Erklärungen tätigen zu können. Je näher eine Erklärung an der ursprünglichen Eingabe ist, desto höher ist die Umsetzbarkeit dieser Erklärung in der Praxis. Eine niedrige Proximität zwischen Erklärung und Ursprung führt zum Verlust des Zusammenhangs, wodurch die Erklärungen nutzlos werden. Dies verhindert jedoch nicht, dass alle Features gleichzeitig variiert werden. Aus diesem Grund ist Sparsität als separate Metrik relevant, da sie sicherstellt, dass nur eine minimale Anzahl an Merkmalen verändert wird. Dies garantiert, dass der Anwender nur möglichst wenige Anpassungen vornehmen muss.[MST20]

Die bisherigen Metriken stellen sicher, dass die Counterfactuals zwar valide, einzigartig und nahe an der Eingabe sind, vernachlässigen jedoch die Beziehung zwischen den Erklärungen selbst. Die Maximierung der Unterschiede zwischen den Counterfactuals wird durch die Diversität erreicht. Dies erhöht die Wahrscheinlichkeit, dass die CF-Menge eine für den Anwender umsetzbare Erklärung beinhaltet. Damit ist Diversität für den Anwender essentiell, um die Entscheidungsgrenze des ML-Modells besser verstehen zu können. Es ist jedoch zu beachten, dass eine hohe Diversität auf Kosten der Proximität geht, da sie konkurrierende Ziele verfolgen.[MST20]

4.2.2 Näherung der Entscheidungsgrenze

Die Optimierung der besprochenen Metriken ist wünschenswert, doch sie sind nicht das endgültige Ziel. Das Ziel von DiCE ist es, mithilfe von Counterfactuals einem Anwender die lokale Entscheidungsgrenze und Logik von ML-Modellen aufzuzeigen. Dies soll dem Anwender ermöglichen, die Entscheidungen, die ein Modell für ähnliche Eingaben treffen wird, vorherzusagen. Eine visuelle Darstellung einer Entscheidungsgrenze ist beispiehaft in Abbildung 4.1 dargestellt.[MST20]

Eine Metrik für eine Messung der Entscheidung, wird über die Verwendung eines weiteren ML-Modells realisiert. Dieses wird auf den generierten Counterfactuals

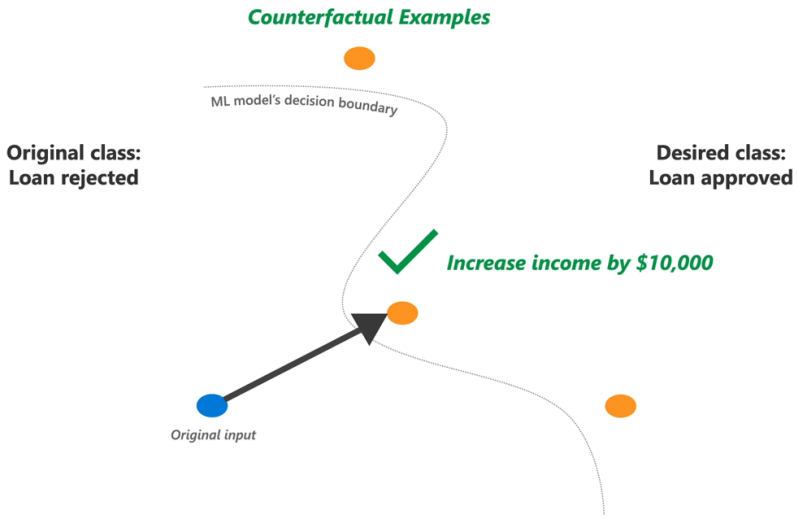

Abbildung 4.1: Visualisierung einer lokalen Entscheidungsgrenze, um die gewünschte Klasse zu erhalten. [`https://interpret.ml/DiCE/`]

und der ursprünglichen Eingabe trainiert und simuliert einen Anwender. Die Güte der Erklärungen kann daran gemessen werden, wie gut das zweite ML-Modell das ursprüngliche Modell imitieren kann.[MST20]

4.3 Vergleich mit anderen Ansätzen

Die Wirksamkeit von DiCE wurde von den Autoren anhand von vier realen Datensätzen (Adult-Income, LendingClub, German-Credit und COMPAS) evaluiert und mit Baselines sowie LIME verglichen. Dazu werden zunächst die alternativen Ansätze vorgestellt und anschließend die zentralen Ergebnisse zusammengefasst.

4.3.1 Baselines und LIME

Vier Baselines werden zum Vergleich für die Generierung von Counterfactuals eingesetzt. **SingleCF** generiert ein einzelnes auf Proximität und Gültigkeit optimiertes Counterfactual [Wat19]. Die Methode **MixedIntegerCF** gegeneriert diverse Erklärungen durch Mixed Integer Programming [Rus19]. Eine Erweiterung der SingleCF ist die Methode **RandoimInitCF** zur Generierung von k Counterfactuals mit zufälligen Startpunkten und einer nicht-konvexen Verlustfunktion. Die letzte Methode ist **NoDiversityCF**, wobei es die Verlustfunktion analog wie in DiCE zur gleichzeitigen Optimierung der Counterfactuals verwendet, wobei jedoch der λ_2-Parameter auf Null gesetzt wird, sodass der Diversitätsterm ignoriert wird.[MST20]

Weiterhin wird DiCE mit LIME (Local Interpretable Model-agnostic Explanations) verglichen. Dieses Verfahren verfolgt einen merkmalsbasierten Erklärungsansatz, also dass die Wichtigkeit einzelner Features für eine Entscheidung bestimmt.

LIME verwendet dafür ein vereinfachtes Ersatzmodell, sodass die Erklärung nicht wahrheitsgetreu gegenüber dem Originalmodell sein muss.[RSG16]

4.3.2 Ergebnisse

Über alle Datensätze hinweg zeigt DiCE durchgehend eine sehr hohe Gültigkeit von fast 100% bei der Generierung von Counterfactuals. MixedIntegerCF weist über alle Datensätze eine mit der Anzahl an Erklärungen sinkende Gültigkeit auf. NoDiverseCF und RandomInitCF weisen eine hohe Gültigkeit ähnlich wie DiCE auf, wobei beim Datensatz COMPAS die Gültigkeit bei steigender Anzahl an geforderten CFs stark abfällt. Dies liegt an der fehlenden Diversität der Verfahren, sodass diese redundante Counterfactuals generieren. Bei der Diversität ist DiCE signifikant besser als die Vergleichsmethoden, unabhängig ob kategorische oder kontinuierliche Features betrachtet werden. Dies hat Auswirkungen auf die Ergebnisse der Proximität, wo DiCE etwas geringere Werte als die Baselines erzielt.[MST20]

Neben der quantitativen Evaluation wurde auch die Näherung der Entscheidungsgrenze untersucht. Hierfür wurde ein einfacher 1-Nearest-Neighbor (1-NN) Klassifikator herangezogen und auf Erklärungen (DiCE) oder Stichproben (LIME) trainiert. DiCE erzielte bei fast durchgehend höhere F1-Scores beider Vorhersage der CF-Klassen als LIME. Dies lässt darauf schließen, dass DiCE einem Anwender die Entscheidungsgrenze besser vermitteln kann. Anzumerken ist, dass die Baseline-Verfahren NoDiverseCF und RandomInitCF ähnlich und teilweise etwas höhere F1-Scores aufweisen. Die Autoren merken an, dass Studien eine höhere Aussagekraft besitzen würden, bei denen die generierten Counterfactuals von Menschen auf ihre Erklärungsqualität evaluiert werden.[MST20]

5

Anwendbarkeit und Bedeutung in der Praxis

In diesem Kapitel wird DiCE als ein Werkzeug betrachtet, um bestehende Probleme in der Praxis zu lösen. Die Anwendung von DiCE wird anhand eines konkreten Beispiel demonstriert.

5.1 Relevanz in der Praxis

Die Anzahl der eingesetzten Black-Box-Modelle steigt in der Welt immer weiter an und trifft Entscheidungen, die das Leben vieler Menschen beeinflussen können. Beispielsweise kann ein solches Modell entscheiden, ob ein Kunde einen Kredit bekommt oder abgelehnt wird. Das Problem hierbei ist, dass der Kunde nun nicht weiß, wie er handeln kann, um den Kredit in Zukunft zu erhalten. Diese Handlungsfähigkeit wird durch die Generierung von Counterfactuals gegeben. Sie liefern dem Anwender mehrere "Was-wäre-wenn"-Szenarien. Aufgrund der Diversität der Erklärungen stehen dem Anwender verschiedene Vorschläge zum Anpassen zur Verfügung und erhöhen so die Umsetzbarkeit von mindestens einer Möglichkeit. Der Einsatz von DiCE ist nicht nur auf den Endanwender beschränkt, sondern kann auch zum Aufdecken von Bias in Modellen verwendet werden. Dies ermöglicht einen breiten Einsatz des Erklärungsansatzes. In der praktischen Anwendung muss berücksichtigt werden, dass viele Features nicht geändert werden können. Dazu zählen das Alter, die ethnische Zugehörigkeit oder auch die Senkung des Bildungsabschlusses. Dieses Problem wird in DiCE durch die manuelle Beschränkung durch den Anwender berücksichtigt, sodass nur plausible Counterfactuals generiert werden. Hierbei liefert DiCE zwei Varianten der Einschränkung. Entweder dürfen Merkmale überhaupt nicht bei der Generierung der Erklärungen verändert werden oder der Wertebereich kann eingeschränkt werden. Dies erlaubt eine große Anpassung auf viele Einsatzsituationen. Es ist zu beachten, dass dies die Gültigkeit und Diversität der Counterfactuals reduzieren kann.[MST20]

5.2 Demonstration eines Beispiels

Um die zuvor diskutierte praktische Relevanz zu veranschaulichen, wird die Anwendung von DiCE an einem konkreten Fallbeispiel demonstriert. Das folgende

Beispiel ist auf GitHub[1] von den Autoren zur Verfügung gestellt wurde. Das Beispiel verwendet `Python` und das Framework `Tensorflow 2.x`. Alternativ unterstützt DiCE auch die Machine-Learning-Frameworks `PyTorch`, `TensorFlow 1.x` und die Bibliothek `sklearn`, auf die in diesem Anwendungsbeispiel nicht näher eingegangen wird.

Für die Demonstration wird der Datensatz *adult* aus dem UCI Machine Learning Repository[2] verwendet. Der Datensatz enthält 8 Merkmale über Personen und die Einteilung, ob sie ein niedriges oder hohes Einkommen besitzen. Die ersten fünf Einträge des Datensatzes sind in Tabelle 5.1 dargestellt.

Tabelle 5.1: Die ersten 5 Einträge aus dem adult Datensatz mit den Features: Alter, Beschäftigungsart, Ausbildung, Familienstand, Beruf, Ethnie, Geschlecht und Wochenarbeitszeit sowie der Klasse Einkommen, aufgeteilt in "0"niedriges Einkommen ($\leq$ 50K) und "1"hohes Einkommen ($>$ 50K).

age	workclass	education	marital_status	occupation	race	gender	h/week	income
28	Private	Bachelors	Single	White-Collar	White	Female	60	0
30	Self-Employed	Assoc	Married	Professional	White	Male	65	1
32	Private	Some-college	Married	White-Collar	White	Male	50	0
20	Private	Some-college	Single	Service	White	Female	35	0
41	Self-Employed	Some-college	Married	White-Collar	White	Male	50	0

5.2.1 Initialisierungsschritte

In Listing 5.1 wird zunächst der Datensatz geladen und in Trainings- sowie Testdaten aufgeteilt. In diesem Beispiel werden 80% des Datensatzes für das Training verwendet und 20% für die Tests. Die Variablen `x_train` und `x_test` enthalten die Features der Trainings- bzw. Testdaten, während `y_train` und `y_test` die zugehörigen Zielklassen (Einkommen) enthalten. In diesem Beispiel wird kein ML-Modell trainiert, sondern ein vor-trainiertes Modell aus der DiCE-Bibliothek der Autoren verwendet.

```
1  # Lade den Datensatz 'adult'
2  dataset = helpers.load_adult_income_dataset()

4  # Teile den Datensatz in Training und Test auf
5  target = dataset["income"]
6  train_dataset, test_dataset, y_train, y_test = train_test_split(dataset,
7                       target, test_size=0.2, random_state=0, stratify=target)

9  # Entferne die Klassifikation aus den Datensaetzen
10 x_train = train_dataset.drop('income', axis=1)
11 x_test = test_dataset.drop('income', axis=1)
```

Listing 5.1: Laden und Aufteilung des Datensatzes in Trainings- und Testdaten.

[1] https://github.com/interpretml/DiCE/blob/main/docs/source/notebooks/DiCE_getting_started.ipynb, Commit 1651751.
[2] https://archive.ics.uci.edu/dataset/2/adult

Als nächstes wird in Listing 5.2 das sogenannte Datenobjekt für DiCE konstruiert. Hierzu müssen die kontinuierlichen Features extra deklariert werden, da diese anders behandelt werden als die kategorialen Features. Weiterhin erfolgt die Angabe der Trainingsdaten und der Zielklasse, welche das Modell lernen soll, vorhersagen zu können.

```
# Konstruiere das Datenobjekt
d = dice_ml.Data(dataframe=train_dataset, continuous_features=['age',
                                  'hours_per_week'], outcome_name='income')
```

Listing 5.2: Konstruktion des Datenobjekts für DiCE.

Es folgt die Konstruktion des Modellobjekts in Listing 5.3. Zunächst wird das Framework `TensorFlow 2.x` durch die Variable `backend` festgelegt. Diese Variable und der Speicherpfad des zugehörigen vor-trainierten ML-Modells der Autoren werden an ein Modellobjekt übergeben. Das dritte Argument des Modellobjekts bestimmt, welche Daten-Transformer-Funktion für die Datenvorverarbeitung verwendet werden soll. Die Funktion *ohe-min-max* transformiert alle kategorialen Features in binäre Vektoren (One-Hot-Encoding) und skaliert alle Wertebereiche der Features auf 0 bis 1 (Min-Max-Normalisierung).

```
# Verwende Tensorflow 2.x
backend = 'TF2'

# Verwendung eines bereits vor-trainierten ML-Modells der Autoren
ML_modelpath = helpers.get_adult_income_modelpath(backend=backend)

# Konstruiere das Modellobjekt
m = dice_ml.Model(model_path=ML_modelpath, backend=backend,
                  func="ohe-min-max")
```

Listing 5.3: Konstruktion des Modellobjekts für DiCE.

Als letzter Vorbereitungsschritt erfolgt die Konstruktion des DiCE-Objekts in Listing 5.4, auf dem die Counterfactuals generiert werden können. Hierzu werden das Datenobjekt und das Modellobjekt übergeben. Der Parameter `method` legt fest, welches Vorgehen, hier das Gradientenabstiegsverfahren, zur Optimierung der Verlustfunktion verwendet werden soll.

```
# Konstruktion des DiCE-Objekts
exp = dice_ml.Dice(d, m, method="gradient")
```

Listing 5.4: Konstruktion des DiCE-Objekts.

5.2.2 Generierung von Erklärungen

Die Initialisierungsphase ist abgeschlossen. Die Erklärungen können nun mit dem konstruierten DiCE-Objekt generiert werden, wie in Listing 5.5 dargestellt wurde. Die Generierung erfolgt über die `generate_ counterfactuals`-Methode unter Angabe eines Eingabevektors, der Anzahl an gewünschten Counterfactuals sowie, ob die Klassifikation der CFs gleich oder ungleich zur Eingabe sein soll.

```
# Generiere Counterfactuals
dice_exp = exp.generate_counterfactuals(x_test[1:2], total_CFs=4,
                                        desired_class="opposite")

# Darstellung der Unterschiede zwischen Eingabevektor und Counterfactuals
dice_exp.visualize_as_dataframe(show_only_changes=True)
```

Listing 5.5: Generierung von vier Counterfactuals und Visualisierung der Änderungen im Vergleich zum Eingabevektor als Tabelle.

Es wird als Eingabe ein Feature-Vektor aus den Testdaten verwendet, der in Tabelle 5.2 dargestellt ist. Die durch den Vektor repräsentierte Person befindet sich in der niedrigen Einkommensklasse. DiCE soll nun erklären, welche Aktionen die Person durchführen könnte, um in die höhere Einkommensklasse aufzusteigen.

Tabelle 5.2: Der Eingabevektor (Zeile 0) aus dem Testset, für den Counterfactuals generiert werden sollen. Das Modell sagt ein hohes Einkommen (0.671) voraus.

age	workclass	education	marital_status	occupation	race	gender	h/week	income
26	Goverment	Some-college	Single	Service	Other	Female	10	0

In Tabelle 5.3 sind die vier generierten Erklärungen für die Person aus Tabelle 5.2 aufgelistet. Der Parameter `show_only_changes` der `visualize_as_dataframe`-Methode ermöglicht, dass nur Änderungen im Vergleich zur Eingabe visualisiert werden. Bei näherer Betrachtung der vorgeschlagenen Änderungen fällt auf, dass diese teilweise nicht umsetzbar sind. Das erste Counterfactual erreicht die gewünschte Einkommensklasse, durch Erhöhung des Bildungsabschlusses, der Änderung des Familienstandes sowie eine wöchentliche Arbeitszeit von 90 Stunden. Diese Merkmale sind theoretisch änderbar, doch die dritte Erklärung verlangt neben einer Heirat ein Alter von 88. Die Änderung des Alters hilft einer Person zwar nicht weiter, um in eine höhere Einkommensklasse zu gelangen, jedoch kann diese Information für Entwickler und Forscher zum Aufzeigen von Bias helfen. Das vierte Counterfactual zeigt ein ähnliches Verhalten durch die Änderung des Geschlechts der betroffenen Person.

Tabelle 5.3: Das diverse Set an generierten Counterfactuals (Ergebnis 0.0). Es werden nur die Features angezeigt, die sich vom Eingabevektor unterscheiden.

age	workclass	education	marital_status	occupation	race	gender	h/week	income
-	-	Masters	Married	-	-	-	90	1
-	-	Doctorate	Married	-	-	-	58	1
88	-	-	Married	-	-	-	-	1
-	-	Bachelors	Married	-	-	Male	72	1

Die Generierung wurde in Listing 5.6 ergänzt die `generate_counterfactuals`-Methode um die Parameter `features_to_vary` und `permitted_range`. Diese Angaben schränken die Counterfacutals ein, um die Machbarkeit zu erhöhen. Hierzu

werden explizit alle Features angegeben, welche in einer Erklärung durch DiCE geändert werden dürfen. Weiterhin ist es möglich Wertebereiche, wie die wöchentliche Arbeitszeit, einzuschränken, sodass die Erklärungen für einen Menschen individuell angepasst werden können, um den maximalen Nutzen aus den Erklärungen zu ziehen. Die neuen Counterfactuals sind in Tabelle 5.4 dargestellt und beinhalten nun praktikablere Vorschläge.

```
# Generiere Counterfactuals mit Einschraenkungen
dice_exp = exp.generate_counterfactuals(x_test[8:9], total_CFs=4,
    desired_class="opposite", features_to_vary=['workclass',
    'hours_per_week', 'education', 'occupation', 'marital_status'],
    permitted_range={'hours_per_week': [10, 50]})


# Darstellung der Unterschiede zwischen Eingabevektor und Counterfactuals
dice_exp.visualize_as_dataframe(show_only_changes=True)
```

Listing 5.6: Generierung von vier Counterfactuals und Visualisierung der Änderungen im Vergleich zum Eingabevektor als Tabelle.

Tabelle 5.4: Das diverse Set an generierten Counterfactuals (Ergebnis 0.0). Es werden nur die Features angezeigt, die sich vom Eingabevektor unterscheiden.

age	**workclass**	**education**	**marital_status**	**occupation**	**race**	**gender**	**h/week**	**income**
-	-	Doctorate	Married	Professional	-	-	37	1
-	-	Prof-school	Married	-	-	-	37	1
-	-	Prof-school	Married	-	-	-	39	1
-	-	Doctorate	Married	-	-	-	45	1

Es ist bei den Einschränkungen zu beachten, dass zu viele Limitierungen eine Generierung von Erklärungen verhindern kann. Wird der Familienstand in dem obigen Beispiel als unveränderlich betrachtet, so findet DiCE keine Counterfactuals mehr. Dieses Phänomen kann auf einen Bias im Datensatz *adult* hindeuten, da die weibliche Person unter den angebenden Einschränkungen aus Tabelle 5.2 nicht ohne Heirat in die höhere Einkommensklasse aufsteigen kann.

6

Zusammenfassung und Ausblick

- zusammenfassung - Herausforderungen - wo liegen die grenzen von dice? - was ist problematisch und was benötigt weitere forschung

Literaturverzeichnis

BATGL+19. Barredo Arrieta, Alejandro, Siham Tabik, Salvador García López, Daniel Molina Cabrera, Francisco Herrera Triguero, Natalia Ana Díaz Rodríguez et al.: *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI.* 2019.

KT+12. Kulesza, Alex, Ben Taskar et al.: *Determinantal point processes for machine learning.* Foundations and Trends® in Machine Learning, 5(2–3):123–286, 2012.

LG19. Loyola-González, Octavio: *Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View.* IEEE Access, 7:154096–154113, 2019.

MST20. Mothilal, Ramaravind K, Amit Sharma und Chenhao Tan: *Explaining machine learning classifiers through diverse counterfactual explanations.* In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, Seiten 607–617, 2020.

RSG16. Ribeiro, Marco Tulio, Sameer Singh und Carlos Guestrin: *"Why Should I Trust You?".* In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 1135–1144, New York, NY, USA, 2016. ACM.

Rus19. Russell, Chris: *Efficient Search for Diverse Coherent Explanations.* In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Seiten 20–28, New York, NY, USA, 2019. ACM.

Wat19. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, New York, NY, USA, 2019. ACM.

A

Eigenständigkeitserklärung

☐ Die vorliegende Arbeit wurde als Einzelarbeit angefertigt.

☐ Die vorliegende Arbeit wurde als Gruppenarbeit angefertigt. Mein Anteil an der Gruppenarbeit ist im untenstehenden Abschnitt *Verantwortliche* dokumentiert:

☐ Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche kenntlich gemacht. Darüber hinaus erkläre ich, dass ich die vorliegende Arbeit in dieser oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht habe.

☐ Es ist keine Nutzung von KI-basierten text- oder inhaltgenerierenden Hilfsmitteln erfolgt.

☐ Die Nutzung von KI-basierten text- oder inhaltgenerierenden Hilfsmitteln wurde von der/dem Prüfenden ausdrücklich gestattet. Die von der/dem Prüfenden mit Ausgabe der Arbeit vorgegebenen Anforderungen zur Dokumentation und Kennzeichnung habe ich erhalten und eingehalten. Sofern gefordert, habe ich in der untenstehenden Tabelle *Nutzung von KI-Tools* die verwendeten KI-basierten text- oder inhaltgenerierenden Hilfsmittel aufgeführt und die Stellen in der Arbeit genannt. Die Richtigkeit übernommener KI-Aussagen und Inhalte habe ich nach bestem Wissen und Gewissen überprüft.

Datum

Unterschrift der Kandidatin/des Kandidaten

Nutzung von KI-Tools

KI-Tool	**Genutzt für**	**Warum?**	**Wann?**	**Mit welcher Eingabefrage bzw. -aufforderung?**	**An welcher Stelle der Arbeit übernommen?**
ChatGPT, DeepSeek, Gemini	Korrektur bzgl. Rechtschreibung, Grammatik und Formulierungen	Verbesserung der Textqualität	Während der gesamten Arbeit	Textabschnitte mit der Aufforderung zur Kontrolle	-
Gemini	Erstellung der Abbildungen 2.1 und 2.2	Schnelle und einfache Modifikation durch Latex-Code	Für die Abbildungen 2.1 und 2.2	Erstelle ein Bild in Latex mit dem Inhalt: input [200€, Männlich, Arbeitslos] -> Black-Box -> "Kein Kreditäls Beispiel zur Erläuterung der Datenverarbeitung eines Black-Box-Modells.	Abbildungen 2.1 und 2.2