

Erklärung von Klassifikatoren des maschinellen Lernens
durch vielfältige kontrafaktische Erklärungen

Explaining Machine Learning Classifiers through Diverse Counter-
factual 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
2.2	Definition von DiCE	3
3	Methodisches Vorgehen	4
3.1	Zentrale Konzepte	4
3.2	Optimierung der Counterfactuals	5
4	Evaluation der Methode	7
4.1	Evaluationsmetriken	7
4.2	Bewertung der Erklärungsqualität	8
5	Anwendbarkeit und Bedeutung in der Praxis	9
5.1	Relevanz in der Praxis	9
5.2	Demonstration eines Beispiels	9
5.2.1	Vorbereitungsschritte	10
5.2.2	Generierung von Erklärungen	11
6	Zusammenfassung und Ausblick	14
	Literaturverzeichnis	15
	Eigenständigkeitserklärung	16

Einleitung und Problemstellung

Künstliche Intelligenz etabliert sich zu einem festen Bestandteil der Arbeitswelt und dem Alltag. Neben dem Einsatz als Chatbot oder Assistenten 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.

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

Ein **Modell des maschinellen Lernens** (ML-Modell) ist ein algorithmisches System, das mit Trainingsdaten zur Erkennung von Mustern durch überwachtes, unüberwachtes, halbüberwachtes oder bestärkendes Lernen trainiert wurde. Anschließend kann das Modell unbekannte Eingabedaten auf Basis des gelernten Wissens verarbeiten und klassifizieren.

Die Eingabe erfolgt in Form eines sogenannten **Feature-Vektors**, wobei die Dimension gleich der Anzahl an kategorischen und numerischen Merkmalen ist. Das Ergebnis erfolgt über die Angabe einer **Klasse**, zu der eine Eingabe durch das ML-Modell zugeordnet wurde. Die **ursprüngliche Eingabe** bezeichnet den unveränderten Feature Vector, welcher in einer **unerwünschten Klasse** resultiert. Eine **kontrafaktische Erklärung**, auch **Counterfactual** (CF) genannt, ist ein verändertes Gegenbeispiel der Eingabe, welches so gewählt wird, dass nicht die ursprünglich erhaltene, unerwünschte Klasse das Ergebnis ist, sondern eine andere Klasse. Diese erwünschte Klasse wird als **CF Klasse** bezeichnet.[MST20]

Bei ML-Modellen wird zwischen **Black-Box-** und **White-Box-Modellen** unterschieden. Black-Box-Modelle sind schwer erklärbar und für Domänenexperten unverständlich, sodass ein Ergebnis nicht nachvollziehbar für einen Menschen ist. Als White-Box-Modelle werden hingegen ML-Modelle bezeichnet, die erklärbare und nachvollziehbare Resultate liefern.[LG19]

Ein **globaler** Erklärungsansatz verfolgt das Ziel, das gesamte Entscheidungsverhalten eines ML-Modells zu verstehen. Im Gegensatz dazu verfolgen **lokale** Ansätze das Ziel ein Modell in einem eingeschränkten, weniger komplexen Lösungsraum zu erklären.[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]

Methodisches Vorgehen

In diesem Kapitel werden die zentralen Konzepte von DiCE betrachtet, welche zur Generierung der Counterfactuals benötigt werden. Darunter fallen Proximität, Diversität und Sparsität. Abschließend wird die Verlustfunktion als Möglichkeit der Optimierung von Counterfactuals untersucht.

3.1 Zentrale Konzepte

Diversität beschreibt, wie sich 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 schließen. Um Diversität zu berücksichtigen, wird in DiCE das Konzept der **Determinantal Point Processes** (DPP) verwendet, um 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]

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

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]

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

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]

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

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]

Evaluation der Methode

In diesem Kapitel werden quantitative Evaluationsmetriken für Gültigkeit, Proximität, Diversität und Sparsität definiert, um generierte Erklärungen bewerten zu können. Anschließend ...

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]

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

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]

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

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]

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

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 Counterfactuals c_i wird die Anzahl an veränderten Features im Vergleich zur ursprünglichen Eingabe x gezählt. Jedes der k Counterfactual 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]

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

[MST20]

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

[MST20]

$$Count - Diversity := \Delta = \frac{1}{C_k^2 d} \sum_{i=1}^{k-1} \sum_{j=i+1}^k \sum_{l=1}^d 1_{[c_i^l \neq c_j^l]} \quad (4.6)$$

4.2 Bewertung der Erklärungsqualität

- Wie aussagekräftig sind die quantitativen Metriken? - Wie sieht es mit qualitativen Metriken aus? - Wie gut im Vergleich zu anderen Erklärungsansätzen?

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 Beispiels 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 Bei-

spiel ist auf GitHub¹ 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 Repository² 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 Vorbereitungsschritte

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 Datensätzen
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.

```
1 # Konstruiere das Datenobjekt
2 d = dice_ml.Data(dataframe=train_dataset, continuous_features=['age',
3                                     '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).

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

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

7 # Konstruiere das Modellobjekt
8 m = dice_ml.Model(model_path=ML_modelpath, backend=backend,
9                   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.

```
1 # Konstruktion des DiCE-Objekts
2 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.

```

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

5 # Darstellung der Unterschiede zwischen Eingabevektor und Counterfactuals
6 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	Government	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 An-

gaben schränken die Counterfactuals 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.

```

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

8 # Darstellung der Unterschiede zwischen Eingabevektor und Counterfactuals
9 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 angebenen Einschränkungen aus Tabelle 5.2 nicht ohne Heirat in die höhere Einkommensklasse aufsteigen kann.

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.

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	-