Hochschule München
University of Applied Sciences

Fakultät Für Informatik und Mathematik
Stochastic Engineering in Business and Finance

Master-Thesis

Interpretation of linear models with SHAP

Interpretation linearer Modelle mit SHAP

Betreuer: Prof. Dr. Andreas Zielke

Eingereicht von:
Simon Symhoven, 49651418
Boschetsriederstraße 59A, D-81379 München
simon.symhoven@hm.edu

Eingereicht am:
München, den 12. Januar 2024

Abstract

Diese Masterarbeit beleuchtet die Interpretation linearer Modelle mit SHAP, welches seine theoretische Basis in den Shapley-Werten der kooperativen Spieltheorie findet. Nach einer historischen Einordnung und Begriffsdefinition werden die Shapley-Werte formal hergeleitet und deren axiomatische Grundlagen beleuchtet. Der Übergang von Shapley-Werten zu SHAP wird zeigen wie Beiträge einzelner Merkmale zur Modellvorhersage beitragen. Am Beispiel einer ausgewählten Modellklasse und unter Verwendung des `shap` Python-Pakets wird die praktische Anwendbarkeit von SHAP auf einen konkreten Datensatz demonstriert. Die Arbeit schließt mit einer Diskussion über die Grenzen von SHAP und bietet einen Ausblick auf dessen Einsatzmöglichkeiten für transparente und nachvollziehbare Modellentscheidungen in der Datenwissenschaft.

Inhaltsverzeichnis

1. Einleitung

In einer Zeit, in der datengetriebene Ansätze und automatisierte Modelle immer größere Relevanz erlangen, rückt die Notwendigkeit der Erklärbarkeit und Interpretierbarkeit von Modellen in den Vordergrund. Eines der vielversprechendsten Konzepte, das sich dieser Herausforderung annimmt, sind die sogenannten Shapley-Werte. Diese Masterarbeit erkundet die tiefgreifenden Konzepte der Shapley-Werte, ihre Anwendungen im Kontext von Machine Learning-Modellen und ihre praktische Umsetzung auf reale Datensätze.

Die Arbeit beginnt mit einer umfassenden Einführung in die Shapley-Werte und ihre historischen Wurzeln. Dabei wird insbesondere auf die kooperative Spieltheorie als Ursprung dieser Konzepte eingegangen. Anhand ausgewählter Literatur werden die theoretischen Grundlagen erörtert und der Forschungsstand auf diesem Gebiet aufgezeigt.

Im Anschluss daran wird die Brücke zur aktuellen Landschaft des maschinellen Lernens geschlagen. Es wird beleuchtet, wie die Shapley-Werte adaptiert werden können, um Einblicke in die Gewichtung von Merkmalen in komplexen Machine Learning-Modellen zu gewinnen. Dabei wird auf bestehende Methoden und Ansätze Bezug genommen und diskutiert, wie diese auf verschiedene Modelle angewendet werden können.

Ein zentraler Schwerpunkt der Arbeit liegt auf der praktischen Anwendung der Shapley-Werte. Ein realer Datensatz wird vorgestellt und die Methodik wird auf diesen angewendet, um die Wirksamkeit und Aussagekraft der Shapley-Werte in der Praxis zu evaluieren. Dies ermöglicht eine kritische Reflexion über die Stärken und Limitationen dieses Ansatzes im Kontext der Datenerklärung.

Abschließend werden die gewonnenen Erkenntnisse zusammengeführt und ein Ausblick auf zukünftige Entwicklungen und Forschungsrichtungen gegeben. Die Arbeit trägt somit dazu bei, das Verständnis für die Shapley-Werte als Instrument der Erklärbarkeit in komplexen Modellen zu vertiefen und ihre praktische Anwendbarkeit zu beleuchten.

2. Historischer Kontext und Begriffsdefinitionen

2.1. Der Ursprung der Shapley-Werte in der kooperativen Spieltheorie

Der Ursprung der Shapley-Werte liegt in der kooperativen Spieltheorie, einem fundamentalen Zweig der Spieltheorie. Dieser Bereich beschäftigt sich mit der Analyse von Situationen, in denen Akteure zusammenarbeiten, um gemeinsame Ziele zu erreichen. Zentrales Anliegen ist dabei die gerechte Verteilung der entstehenden Gewinne unter den Akteuren. Ein Schlüsselkonzept dieser Theorie ist die sogenannte „Charakteristische Funktion“, welche die Bewertung der Gewinnverteilung einer Koalition von Akteuren ermöglicht.

Die Shapley-Werte, entwickelt von Lloyd Shapley in den 1950er Jahren, bieten einen methodischen Ansatz, um den individuellen Beitrag eines jeden Akteurs zur kooperativen Zusammenarbeit gerecht zu bewerten. Dies geschieht durch die Durchschnittsbewertung der Beiträge über sämtliche mögliche Koalitionen hinweg. Diese Methode erweist sich als äußerst nützlich, um eine gerechte und rationale Verteilung von Gewinnen in vielfältigen Szenarien zu ermöglichen, sei es in wirtschaftlichen Verhandlungen oder der Aufteilung von Ressourcen.

Das Verständnis der kooperativen Spieltheorie und ihrer Anwendung in Form der Shapley-Werte ermöglicht es, dieses theoretische Konzept auf den Bereich des maschinellen Lernens zu übertragen. In dieser Arbeit wird der Übergang von abstrakten Spieltheorie-Konzepten zu konkreten Anwendungen in der Welt der datengetriebenen Modelle erforscht.

Zur Erreichung dieses Ziels werden in den kommenden Abschnitten nicht nur die formalen Definitionen und Eigenschaften der Shapley-Werte erläutert, sondern auch ihre Adaption und Anwendung auf Machine Learning-Modelle in Betracht gezogen. Die Anwendbarkeit wird durch die praktische Anwendung auf einen realen Datensatz verdeutlicht.

2.2. Shapley-Werte, SHAP, SHAP-Werte und shap

Zur Verdeutlichung und Abgrenzung der verschiedenen, jedoch verwandten Begrifflichkeiten, die im Kontext dieser Arbeit Verwendung finden, ist eine kurze Einordnung essenziell.

Beginnend mit den Shapley-Werten, entstammt dieser Begriff der kooperativen Spieltheorie und beschreibt eine Methode, um den fairen Beitrag eines Spielers zu der

Gesamtauszahlung eines kooperativen Spiels zu bestimmen.

SHAP (SHapley Additive exPlanations) ist ein Interpretationsframework, das die Shapley-Werte in den Bereich des maschinellen Lernens überträgt. Der Begriff wurde erstmals von Lundberg und Lee eingeführt [LL17, S. 1].

Die SHAP-Werte sind dann die konkreten quantitativen Beiträge der einzelnen Merkmale zu einer bestimmten Vorhersage, berechnet basierend auf dem SHAP-Framework.

Das Python-Paket `shap` schließlich ist eine Implementierung, die es praktikabel macht, SHAP-Werte in der Anwendung zu berechnen und zu visualisieren. Es stellt eine reiche Auswahl an Werkzeugen zur Verfügung, um diese Werte und ihre Auswirkungen zu interpretieren.

3. Theorie der Shapley-Werte

In diesem Kapitel werden die Shapley-Werte als Instrument zur gerechten Aufteilung von Gewinnen in kooperativen Spielen vorgestellt. Durch die Verwendung eines praktischen Beispiels – der Aufteilung eines Preisgeldes aus einem Designwettbewerb unter den Gewinnern – wird zunächst eine intuitive Einführung in das Konzept gegeben. Anschließend wird die formale Definition der Shapley-Werte erläutert, um die theoretischen Grundlagen für ihre Berechnung und Anwendung zu legen.

3.1. Wie lässt sich der Gewinn gerecht aufteilen?

Angenommen, drei Teilnehmer, Anna, Ben und Carla, haben als Team kooperiert und den ersten Platz bei einem Designwettbewerb belegt[1]. Dieser Erfolg führt zu einem Gesamtgewinn von 1000 €. Das Preisgeld für den zweiten Platz beträgt 750 € und 500 € für den dritten Platz. Die Herausforderung besteht nun darin, den Gewinn auf eine Weise zu verteilen, die den individuellen Beitrag jedes Teilnehmers zur Erzielung des ersten Platzes gerecht widerspiegelt.

Die Situation wird komplizierter, wenn man bedenkt, dass jeder Teilnehmer unterschiedlich zu dem Erfolg beigetragen hat und ihre individuellen Leistungen auch zu verschiedenen Ausgängen geführt hätten, wenn sie alleine oder in anderen Teilkonstellationen angetreten wären.

Um eine faire Aufteilung des Preisgeldes zu erreichen, betrachten wir die hypothetischen Gewinne, die Anna, Ben und Carla erzielt hätten, wenn sie in unterschiedlichen Konstellationen am Wettbewerb teilgenommen hätten. Tabelle 1 zeigt die gegegbene Gewinnverteilung der verschiedenen Koalitionen. Die Koalition $\emptyset$ entspricht dabei der leeren Koalition – der Nichtteilnahme an dem Wettbewerb.

Zur Berechnung der Shapley-Werte ist es erforderlich, den marginalen Beitrag jedes Spielers zu erfassen. Marginalbeiträge in der Spieltheorie, und speziell im Kontext der Shapley-Werte, sind die zusätzlichen Beiträge, die ein Spieler (Teilnehmer) zum Gesamtgewinn einer Koalition beiträgt, wenn er dieser beitritt. Die Berechnung des marginalen Beitrags eines Teilnehmers erfolgt, indem man den Wert der Koalition ohne diesen Teilnehmer vom Wert der Koalition mit dem Teilnehmer subtrahiert [Mol23, S. 18].

In diesem Beispiel mit Anna, Ben und Carla, die an einem Designwettbewerb teil-

[1] In Anlehnung an das Beispiel aus Kapitel 4.1 „Who's going to pay for that taxi?“ [Mol23, S.17-20].

Tabelle 1.: Potenzielle Gewinne für verschiedene Teilnehmerkonstellationen im Designwettbewerb.

Koalition	Gewinn	Bemerkung
∅	0 €	Keine Teilnahme
{Anna}	500 €	3. Platz als Einzelteilnehmerin
{Ben}	750 €	2. Platz als Einzelteilnehmer
{Carla}	0 €	Kein Gewinn als Einzelteilnehmerin
{Anna, Ben}	750 €	2. Platz als Team ohne Carla
{Anna, Carla}	750 €	2. Platz als Team ohne Ben
{Ben, Carla}	500 €	3. Platz als Team ohne Anna
{Anna, Ben, Carla}	1000 €	1. Platz als Gesamtteam

Quelle: Eigene Darstellung.

nehmen, ist der marginale Beitrag von Anna zur Koalition von {Ben} der zusätzliche Wert, den sie einbringt, wenn sie sich Ben anschließt, ausgehend von Bens individuellem Gewinn.

Tabelle 2.: Marginalbeiträge der einzelnen Teilnehmer zu den möglichen Koalitionen.

Teilnehmer	Zur Koalition	Gewinn vorher	Gewinn nachher	Marginalbeitrag
Anna	∅	0 €	500 €	500 €
Anna	{Ben}	750 €	750 €	0 €
Anna	{Carla}	0 €	750 €	750 €
Anna	{Ben, Carla}	500 €	1000 €	500 €
Ben	∅	0 €	750 €	750 €
Ben	{Anna}	500 €	750 €	250 €
Ben	{Carla}	0 €	500 €	500 €
Ben	{Anna, Carla}	750 €	1000 €	250 €
Carla	∅	0 €	0 €	0 €
Carla	{Anna}	500 €	750 €	250 €
Carla	{Ben}	750 €	500 €	-250 €
Carla	{Anna, Ben}	750 €	1000 €	250 €

Quelle: Eigene Darstellung.

Die Tabelle 2 illustriert den Gewinn jeder möglichen Koalition ohne den betrachteten Spieler und den neuen Gesamtgewinn, sobald dieser Spieler der Koalition beitritt. Der marginale Beitrag jedes Spielers wird dann als die Differenz zwischen diesen beiden Werten berechnet und gibt Aufschluss über den individuellen Wertbeitrag zum gemeinschaftlichen Erfolg.

Nachdem die marginalen Beiträge jedes Teilnehmers für die verschiedenen Koalitionen festgestellt wurden, ist der nächste Schritt, die Shapley-Werte zu bestimmen, welche eine faire Aufteilung des Gesamtgewinns erlauben. Hierzu wird jede mögliche Reihenfolge (Permutation) betrachtet, in der die Spieler der Koalition beitreten könnten. Jede dieser Permutationen liefert unterschiedliche marginale Beiträge für die Spieler, je nach der Reihenfolge ihres Beitritts [Sha53, S. 307ff].

Im Falle dieses Beispiels mit Anna, Ben und Carla bedeutet dies, dass alle möglichen Reihenfolgen berücksichtigt werden müssen, in denen sie zum ersten Platz beigetragen haben könnten. Die Shapley-Werte werden dann als Durchschnitt der marginalen

Beiträge über alle Permutationen berechnet. Dies gewährleistet, dass jeder Spieler einen Anteil des Preisgeldes erhält, der seinem durchschnittlichen Beitrag zum Erfolg entspricht.

Bei drei Teilnehmern exisitieren $3! = 3 \cdot 2 \cdot 1 = 6$ Permutationen:

1. Anna, Ben, Carla
2. Anna, Carla, Ben
3. Ben, Anna, Carla
4. Carla, Anna, Ben
5. Ben, Carla, Anna
6. Carla, Ben, Anna

Jede Permutation entspricht einer Koalitionsbildung. Anna wird in zwei Koalitionsbildungen (1. und 2.) einer leeren Koalition hinzugefügt. In weiteren zwei Koalitionsbildungen (5. und 6.) wird Anna der bestehenden Koalition aus Ben und Carla hinzugefügt. In den beiden übrigen Koalitionsbildungen wird Anna einmal der Koaliton bestehend aus Ben (3.) und einmal der Koalition bestehend aus Carla (4.) hinzugefügt.

Zusammen mit Tabelle 2 lässt sich nun der Shapley-Wert mit den gewichteten durchschnittlichen marginalen Beiträge für Anna berechnen:

$$\frac{1}{6}(\underbrace{2 \cdot 500€}_{\mathrm{A} \to \{\emptyset\}} + \underbrace{1 \cdot 0€}_{\mathrm{A} \to \{B\}} + \underbrace{1 \cdot 750€}_{\mathrm{A} \to \{C\}} + \underbrace{2 \cdot 500€}_{\mathrm{A} \to \{B,C\}}) \approx 458,34€ \tag{3.1}$$

Analog gilt das für Ben:

$$\frac{1}{6}(\underbrace{2 \cdot 750€}_{\mathrm{B} \to \{\emptyset\}} + \underbrace{1 \cdot 250€}_{\mathrm{B} \to \{A\}} + \underbrace{1 \cdot 500€}_{\mathrm{B} \to \{C\}} + \underbrace{2 \cdot 250€}_{\mathrm{B} \to \{A,C\}}) \approx 458,34€ \tag{3.2}$$

und Carla:

$$\frac{1}{6}(\underbrace{2 \cdot 0€}_{\mathrm{C} \to \{\emptyset\}} + \underbrace{1 \cdot 250€}_{\mathrm{C} \to \{A\}} + \underbrace{1 \cdot (-250)€}_{\mathrm{C} \to \{B\}} + \underbrace{2 \cdot 250€}_{\mathrm{C} \to \{A,B\}}) \approx 83,34€ \tag{3.3}$$

Auf Basis der gewichteten durchschnittlichen marginalen Beiträge lässt sich feststellen, dass Anna und Ben jeweils einen Shapley-Wert von ungefähr 458,34 € erhalten, während Carla einen Shapley-Wert von etwa 83,34 € zugewiesen bekommt. Diese Werte spiegeln den fairen Anteil jedes Teilnehmers an der Gesamtprämie wider, basierend auf ihrem individuellen Beitrag zum Erfolg des Teams. Mit dieser konkreten Anwendung der Shapley-Werte auf ein alltagsnahes Beispiel wird nun die zugrunde liegende Theorie und die formale Definition der Shapley-Werte, die diese Berechnungen ermöglichen, detaillierter betrachtet.

3.2. Formale Definition

Sei $\mathcal{N} = \{1, \ldots, n\}$ eine endliche Spielermenge mit $n := |\mathcal{N}|$ Elementen. Sei v die Koalitionsfunktion, die jeder Teilmenge von $\mathcal{N}$ eine reele Zahl zuweist und insbesondere der leeren Koalition den Wert 0 gibt.

$$\begin{array}{rcccc} v & : & \mathcal{P}(\mathcal{N}) & \longrightarrow & \mathbb{R} \\ & : & v(\emptyset) & \mapsto & 0 \end{array}$$

Eine nicht leere Teilmenge der Spieler $\mathcal{S} \subseteq \mathcal{N}$ heißt Koalition. $\mathcal{N}$ selbst bezichnet die große Koalition. Den Ausdruck $v(\mathcal{S})$ nennt man den Wert der Koalition $\mathcal{S}$. Der Shapley-Wert ordnet nun jedem Spieler aus $\mathcal{N}$ eine Auszahlung für das Spiel v zu.

Der marginale Beitag eines Spieler $i \in \mathcal{N}$, also der Wertbeitrag eines Spielers zu einer Koalition $\mathcal{S} \subseteq \mathcal{N}$, durch seinen Beitritt, ist

$$v(\mathcal{S} \cup \{i\}) - v(\mathcal{S}). \tag{3.4}$$

Sei $i =$ Anna und $\mathcal{S} = \{\text{Ben}\}$, dann ist $v(\{\text{Ben}\} \cup \{\text{Anna}\}) - v(\{\text{Ben}\})$ das zusätzliche Preisgeld, welches gewonnen wird, wenn Anna der Koalition mit Ben beitritt.

Der Shapley-Wert eines Spielers i errechnet sich als das gewichtete Mittel der marginalen Beiträge zu allen möglichen Koalitionen:

$$\varphi_i(\mathcal{N}, v) = \sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}} \underbrace{\frac{|\mathcal{S}|! \cdot (n - 1 - |\mathcal{S}|)!}{n!}}_{\text{Gewicht}} \underbrace{v(\mathcal{S} \cup \{i\}) - v(\mathcal{S})}_{\substack{\text{marginaler Beitrag von} \\ \text{Spieler } i \text{ zur Koalition } \mathcal{S}}} . \tag{3.5}$$

Die Summationsnotation $\sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}}$ erfasst die marginalen Beiträge, die der Spieler i zu allen Koalitionen leistet, die diesen noch nicht einschließen. Die Verwendung von $\mathcal{N} \setminus \{i\}$ stellt sicher, dass Spieler i nur für jene Koalitionen berücksichtigt wird, zu denen er noch beitragen kann. Im Falle von Anna etwa, beziehen sich die Berechnungen auf die Koalitionen bestehend aus der der leeren Koalition $\emptyset$, aus {Ben}, {Carla}, oder beiden zusammen {Ben, Carla} (vgl. Berechnung 3.1).

Die Formel $\frac{|\mathcal{S}|! \cdot (n-1-|\mathcal{S}|)!}{n!}$ in der Shapley-Wert-Berechnung reflektiert den Gewichtungsfaktor für die marginalen Beiträge eines Spielers. Hierbei gibt $|\mathcal{S}|!$ die Permutationen der Spieler innerhalb der Koalition $\mathcal{S}$ an, während $(n - 1 - |\mathcal{S}|)!$ die Anordnungen der außenstehenden Spieler repräsentiert, nachdem der betrachtete Spieler beigetreten ist. Der Bruchteil $\frac{1}{n!}$ normalisiert diesen Wert über alle möglichen Koalitionszusammensetzungen, wodurch die Wahrscheinlichkeit der Bildung einer spezifischen Koalition ausgedrückt wird.

Betrachten wir Anna als den Spieler i und die Koalition $\mathcal{S} =$ {Ben, Carla}. Die

Formel $\frac{|\mathcal{S}|!\cdot(n-1-|\mathcal{S}|)!}{n!}$ berechnet den Gewichtungsfaktor für Annas marginalen Beitrag zur Koalition $\mathcal{S}$. In diesem Fall ist $|\mathcal{S}| = 2$ und $n = 3$. Somit ergibt sich $|\mathcal{S}|! = 2!$ und $n - 1 - |\mathcal{S}| = 0!$, da nach dem Beitritt von Anna keine weiteren Spieler übrig sind. Der Normalisierungsfaktor ist $n! = 3! = 6$. Daraus folgt:

$$\frac{2! \cdot 0!}{3!} = \frac{2 \cdot 1}{6} = \frac{1}{3}. \tag{3.6}$$

Dies bedeutet, dass unter allen möglichen Permutationen der Spielerreihenfolge, Annas Beitritt zu der Koalition {Ben, Carla} genau ein Drittel der Zeit am Ende geschieht. Somit wird ihr marginaler Beitrag mit diesem Faktor gewichtet, um den Shapley-Wert zu berechnen (vgl. Berechnung 3.1) [Mol23, S. 21f].

3.3. Axiome

Nachdem die Berechnung des Shapley-Werts für das Beispiel konkretisiert wurde, ist es nun von Bedeutung, die zugrundeliegenden Axiome zu betrachten, welche die theoretische Rechtfertigung für die Methode liefern. Der Shapley-Wert wird nicht nur durch seine Berechnungsmethode, sondern auch durch eine Reihe von Axiomen charakterisiert, die seine Fairness und Kohärenz im Kontext kooperativer Spiele sicherstellen. Lloyd Shapley leitete den Shapley-Wert ursprünglich aus diesen Axiomen ab und bewies, dass dieser der einzige ist, der den Axiomen genügt[2]. Diese Axiome sind wesentliche Bestandteile, die die Einzigartigkeit und die wünschenswerten Eigenschaften des Shapley-Werts als Lösungskonzept definieren [Mol23, S. 22].

Effizienz Der Wert der großen Koalition wird an die Spieler verteilt:

$$\sum_{i \in \mathcal{N}} \varphi_i(\mathcal{N}, v) = v(\mathcal{N}). \tag{3.7}$$

Dies bedeutet, dass die Summe der Shapley-Werte aller Spieler dem Gesamtwert entspricht, den die Koalition aller Spieler zusammen erreichen kann. Der Gesamtwert, den die große Koalition $\mathcal{N}$, bestehend aus Anna, Ben und Carla, generiert, wird komplett unter den Spielern aufgeteilt [Mol23, S. 22]. Unter Vernachlässigung minimaler Rundungsdifferenzen entspricht die Summe der Shapley-Werte, berechnet in den Gleichungen 3.1, 3.2 und 3.3, dem kollektiven Ertrag der großen Koalition:

$$458,34€ + 458,34€ + 83,32€ \approx 1000€ \tag{3.8}$$

[2] Eine detaillierte Darstellung dieser Axiome und des Beweises ihrer Einzigartigkeit findet sich in Shapleys Originalarbeit, deren umfassende Behandlung jedoch den Rahmen dieser Arbeit überschreiten würde [Sha53, S. 307-318].

Symmetrie Zwei Spieler i und j, die die gleichen marginalen Beiträgen zu jeder Koalition haben erhalten das Gleiche:

$$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S} \cup \{j\}),\ \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{i, j\} \Rightarrow \varphi_i(\mathcal{N}, v) = \varphi_j(\mathcal{N}, v). \tag{3.9}$$

Obwohl Anna und Ben den gleichen Shapley-Wert erhalten, ist dies nicht auf das Symmetrieaxiom zurückzuführen, da ihre marginalen Beiträge zu den Koalitionen variieren. Zum Beispiel leistet Anna keinen Beitrag zur Koalition, wenn Ben bereits Teil davon ist, während Ben einen positiven Beitrag leistet, wenn Anna bereits zur Koalition gehört (vgl. Tabelle 2). Dies zeigt, dass die Gleichheit ihrer Shapley-Werte ein Ergebnis der spezifischen Zahlenkonstellation in diesem Szenario ist und nicht aus der symmetrischen Interaktion zwischen den beiden Spielern resultiert.

Null-Spieler-Eigenschaft (Dummy-Spieler-Eigenschaft) Ein Spieler i der zu jeder Koalition nichts beiträgt erhält den Wert Null:

$$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S}),\ \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{i\} \Rightarrow \varphi_i(\mathcal{N}, v) = 0. \tag{3.10}$$

Dies stell sicher, dass ein Spieler, der keinen Beitrag leistet, auch nicht belohnt wird.

Additivität Wenn das Spiel in zwei unabhängige Spiele mit Koalitionsfunktionen v und w zerlegt werden kann, dann ist die Auszahlung jedes Spielers im zusammengesetzten Spiel die Summe der Auszahlungen in den aufgeteilten Spielen:

$$\varphi_i(\mathcal{N}, v + w) = \varphi_i(\mathcal{N}, v) + \varphi_i(\mathcal{N}, w). \tag{3.11}$$

Wenn Anna, Ben und Carla neben dem ersten Wettbewerb an einem zweiten, unabhängigen Wettbewerb teilnehmen, besagt das Additivitätsaxiom, dass die Shapley-Werte jedes Spielers aus beiden Wettbewerben einfach die Summe ihrer individuellen Shapley-Werte aus jedem einzelnen Wettbewerb sind. Dies impliziert, dass die faire Aufteilung der Gewinne aus beiden Wettbewerben konsistent bleibt, indem die aus dem ersten Wettbewerb abgeleiteten Prinzipien auf den zweiten Wettbewerb übertragen und dann addiert werden [RWB+22, Mol22, S. 5573, S.22f].

4. Von Shapley-Werten zu SHAP: Brückenschlag zur Modellinterpretation

Im Rahmen der kooperativen Spieltheorie ermöglichen die Shapley-Werte eine faire Verteilung des kollektiv erwirtschafteten Nutzens auf die beteiligten Akteure. Diese Methodik findet eine analoge Anwendung in der Welt des maschinellen Lernens, um die Beiträge einzelner Merkmale zur Vorhersageleistung eines Modells zu bewerten. Hier wird die Terminologie der Shapley-Werte in den Kontext von Machine Learning Modellen übertragen, wobei jedes Merkmal als „Spieler“ betrachtet wird, dessen Beitrag zur „Auszahlung“ – der Vorhersage des Modells – evaluiert werden soll.

Tabelle 3.: Terminologie der originären Shapley-Werte im Kontext des maschinellen Lernens.

Terminologie Konzept	Terminologie Machine Learning	Ausdruck
Spieler	Merkmal Index	j
Anzahl aller Spieler	Anzahl aller Merkmale	p
Große Koalition	Menge aller Merkmale	$\mathcal{N} = \{1, \ldots, p\}$
Koalition	Menge von Merkmalen	$\mathcal{S} \subseteq \mathcal{N}$
Größe der Koalition	Anzahl der Merkmale in der Koalition $\mathcal{S}$	$\lvert\mathcal{S}\rvert$
Spieler, die nicht in der Koalition sind	Merkmale, die nicht in der Koalition enthalten sind	$C : C = \mathcal{N} \setminus \mathcal{S}$
Koalitionsfunktion	Vorhersage für Merkmalswerte in der Koalition $\mathcal{S}$ abzüglich der Vorhersage im Mittel	$v_{f,x^{(i)}}(\mathcal{S})$
Auszahlung	Vorhersage für eine Beobachtung $x^{(i)}$ abzüglich der Vorhersage im Mittel	$f(x^{(i)}) - \mathbb{E}(f(X))$
Shapley-Wert	Beitrag des Merkmals j zur Auszahlung des Modells für eine Beobachtung $x^{(i)}$	$\varphi_j^{(i)}(\mathcal{N}, f)$

Quelle: [Mol23, S. 26].

Die Koalitionsfunktion $v_{f,x^{(i)}}(\mathcal{S})$ für ein gegebenes Model f und eine Beobachtung $x^{(i)}$ ist definiert als:

$$v_{f,x^{(i)}}(\mathcal{S}) = \int_{\mathbb{R}} f(x_{\mathcal{S}}^{(i)} \cup X_C) d\mathbb{P}_{X_C} - \mathbb{E}(f(X)) \tag{4.1}$$

Diese Funktion berechnet den erwarteten Wert der Vorhersage des Modells f, wenn nur eine Teilmenge $\mathcal{S}$ der Merkmale genutzt wird, um die Vorhersage für die spezifische Beobachtung $x^{(i)} \in \mathbb{R}^p$ zu treffen. Das Integral $\int_{\mathbb{R}}$ repräsentiert die Berechnung

dieses erwarteten Wertes über alle möglichen Werte der Merkmale, die nicht in $\mathcal{S}$ enthalten sind (X_C), gewichtet durch deren Wahrscheinlichkeitsverteilung $\mathbb{P}_{X_C}$. Die Differenz zum Erwartungswert der Vorhersagen über alle Merkmale $\mathbb{E}(f(X))$ zeigt, wie viel die spezifische Menge an Merkmalen $\mathcal{S}$ zur Vorhersage beiträgt [Mol22, Mol23, S. 221, S. 27].

Der marginale Beitrag eines Merkmals j zu einer Koalition $\mathcal{S}$ ist dann:

$$\begin{aligned} v_{f,x^{(i)}}(\mathcal{S} \cup \{j\}) - v_{f,x^{(i)}}(\mathcal{S}) &= \int_{\mathbb{R}} f(x^{(i)}_{\mathcal{S}\cup\{j\}} \cup X_{C\setminus\{j\}}) d\mathbb{P}_{X_{C\setminus\{j\}}} - \mathbb{E}(f(X)) \\ &\quad - \left(\int_{\mathbb{R}} f(x^{(i)}_{\mathcal{S}} \cup X_C) d\mathbb{P}_{X_C} - \mathbb{E}(f(X)) \right) \\ &= \int_{\mathbb{R}} f(x^{(i)}_{\mathcal{S}\cup\{j\}} \cup X_{C\setminus\{j\}}) d\mathbb{P}_{X_{C\setminus\{j\}}} \\ &\quad - \int_{\mathbb{R}} f(x^{(i)}_{\mathcal{S}} \cup X_C) d\mathbb{P}_{X_C} \end{aligned} \tag{4.2}$$

Diese Gleichung beschreibt, wie sich der erwartete Wert der Vorhersage ändert, wenn das Merkmal j zu der Menge der Merkmale $\mathcal{S}$ hinzugefügt wird [Mol23, S. 29].

Der Beitrag $\varphi_j^{(i)}(\mathcal{N}, f)$ eines Merkmals j für eine Beobachtung $x^{(i)} \in \mathbb{R}^p$ für die Vorhersage $f(x^{(i)})$ ist gegeben als:

$$\begin{aligned} \varphi_j^{(i)}(\mathcal{N}, f) = &\sum_{\mathcal{S}\subseteq\mathcal{N}\setminus\{j\}} \frac{|\mathcal{S}|! \cdot (p - 1 - |\mathcal{S}|)!}{p!} \\ &\cdot \left(\int_{\mathbb{R}} f(x^{(i)}_{\mathcal{S}\cup\{j\}} \cup X_{C\setminus\{j\}}) d\mathbb{P}_{X_{C\setminus\{j\}}} - \int_{\mathbb{R}} f(x^{(i)}_{\mathcal{S}} \cup X_C) d\mathbb{P}_{X_C} \right) \end{aligned} \tag{4.3}$$

Diese Formel ist die zentrale Berechnung der SHAP-Werte im maschinellen Lernen. Sie summiert den gewichteten, marginalen Beitrag des Merkmals j über alle möglichen Kombinationen der anderen Merkmale. Die Gewichtung berücksichtigt die Anzahl der Merkmale in der Koalition $\mathcal{S}$ und die Anzahl der verbleibenden Merkmale, die noch hinzugefügt werden können. Dies ergibt den durchschnittlichen Beitrag des Merkmals j zur Vorhersage für die Beobachtung $x^{(i)}$ [Mol23, S. 29, 30].

Die Integration in der SHAP-Formel ist ein zentraler Schritt, um den erwarteten Beitrag jedes Merkmals unter Berücksichtigung der gesamten Verteilung der Daten zu ermitteln. In diesem Ansatz werden die Merkmale als Zufallsvariablen behandelt, und die Integration erfolgt über die Wahrscheinlichkeitsverteilungen dieser Zufallsvariablen. Durch das Berechnen der erwarteten Vorhersagewerte mit und ohne des jeweiligen Merkmals, unter Einbeziehung der Verteilung aller anderen Merkmale, ermöglicht SHAP eine präzise und umfassende Einschätzung des Einflusses jedes einzelnen Merkmals. Dieser Prozess der Marginalisierung, bei dem man über die Wahr-

scheinlichkeitsverteilungen der Merkmale integriert, erlaubt es, den Beitrag eines jeden Merkmals zu isolieren und unabhängig von der spezifischen Zusammensetzung der anderen Merkmale zu bewerten. Dies führt zu einer fairen und ganzheitlichen Bewertung der Beiträge aller Merkmale zur Vorhersage des Modells [Mol23, S. 28].

4.1. Berechnung der SHAP-Werte unter Berücksichtigung der zugrundeliegenden Verteilung

Ein einfaches Beispiel soll helfen, die Anwendung von SHAP-Werten im Kontext des maschinellen Lernens zu illustrieren[1]. Betrachtet wird ein fiktiver Immobilien-Datensatz mit drei Merkmalen: Größe des Hauses in Quadratmetern (x_1), Anzahl der Zimmer (x_2) und Entfernung zum Stadtzentrum in Kilometern (x_3). Es gibt zwei Beobachtungen in diesem Datensatz:

Tabelle 4.: Merkmale von Beobachtungen in einem Immobilien-Datensatz.

	x_1: Größe (in m^2)	x_2: Anzahl Zimmer	x_3: Entfernung zum Zentrum (in km)
$x^{(1)}$	100	3	5
$x^{(2)}$	150	4	10

Quelle: Eigene Darstellung.

Angenommen das Modell $f(x^{(i)})$ prognostiziert den Preis eines Hauses in Euro als eine lineare Kombination der Merkmale:

$$f(x^{(i)}) = 5x_1^{(i)} + 20x_2^{(i)} - 2x_3^{(i)}. \tag{4.4}$$

Die Vorhersagen für die beiden Beobachtungen lauten dann:

$$\begin{aligned} f(x^{(1)}) &= 5x_1^{(1)} + 20x_2^{(1)} - 2x_3^{(1)} \\ &= 5 \cdot 100 + 20 \cdot 3 - 2 \cdot 5 \\ &= 550\,€ \end{aligned} \tag{4.5}$$

und

$$\begin{aligned} f(x^{(2)}) &= 5x_1^{(2)} + 20x_2^{(2)} - 2x_3^{(2)} \\ &= 5 \cdot 150 + 20 \cdot 4 - 2 \cdot 10 \\ &= 810\,€. \end{aligned} \tag{4.6}$$

[1] In Anlehnung an das Beispiel aus Kapitel 8.5.1 „General Idea“ [Mol22, S.215f].

Die erwartete Auszahlung des Modells $\mathbb{E}(f(X))$ wird berechnet als:

$$\begin{aligned}\mathbb{E}(f(X)) &= 5 \cdot \mathbb{E}(X_1) + 20 \cdot \mathbb{E}(X_2) - 2 \cdot \mathbb{E}(X_3) \\ &= 5 \cdot 125 + 20 \cdot 3,5 - 2 \cdot 7,5 \\ &= 680\,€,\end{aligned} \tag{4.7}$$

mit

$$\mathbb{E}(X_j) = \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)}. \tag{4.8}$$

Sei $\mathcal{N} = \{1, 2, 3\}$ die Menge aller Merkmale und die Beobachtung $x^{(1)} = [100, 3, 5]$. Der SHAP-Wert für jedes Merkmal $j \in \mathcal{N}$ wird unter Berücksichtigung der Verteilung der Daten und der Formel 4.3 berechnet:

$$\begin{aligned}\varphi_j^{(1)}(\mathcal{N}, f) = &\sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{j\}} \frac{|\mathcal{S}|! \cdot (p - 1 - |\mathcal{S}|)!}{p!} \\ &\cdot \left(\int_{\mathbb{R}} f(x^{(1)}_{\mathcal{S} \cup \{j\}} \cup X_{C \setminus \{j\}}) d\mathbb{P}_{X_{C \setminus \{j\}}} - \int_{\mathbb{R}} f(x^{(1)}_{\mathcal{S}} \cup X_C) d\mathbb{P}_{X_C} \right)\end{aligned} \tag{4.9}$$

wobei $p = |\mathcal{N}| = 3$ die Anzahl der Merkmale ist und X_C die Menge der Merkmale außerhalb der Koalition $\mathcal{S}$ repräsentiert. Die Integrale repräsentieren die erwartete Vorhersage des Modells über die Verteilung der nicht in der Koalition enthaltenen Merkmale.

In linearen Modellen, unter der Prämisse, dass die Merkmale unabhängig voneinander und gleichverteilt sind, ist es möglich, die Berechnung der SHAP-Werte zu vereinfachen. Anstelle der komplexen Integration über die Verteilungen aller Merkmale, kann der Fokus auf die Unterschiede in den Modellvorhersagen gelegt werden, die sich aus dem Hinzufügen oder Entfernen einzelner Merkmale ergeben. Hierbei wird anstelle der spezifischen Werte der nicht in der betrachteten Koalition enthaltenen Merkmale die Erwartungswerte herangezogen. Diese Vereinfachung ermöglicht es, den Einfluss jedes Merkmals auf die Modellvorhersage auf eine direktere und rechnerisch weniger aufwendige Weise zu erfassen. Diese Vereinfachung ist für lineare Modelle angemessen, da die Auswirkungen jedes Merkmals auf die Vorhersage des Modells additiv und unabhängig sind. Bei komplexeren, nichtlinearen Modellen ist eine detailliertere Berechnung erforderlich, die oft auf numerischen Methoden oder Annäherungen basiert, mehr dazu in Kapitel 5.

Der Beitrag durch das Hinzufügen des Merkmals x_1 zur bestehenden Koalition $\mathcal{S} = \{x_2\}$ wird nach Formel 4.9 berechnet als:

$$\varphi_1^{(1)}(\{x_2\}, f) = \frac{1! \cdot (3-1-1)!}{3!} \cdot \int f(x_1, x_2, X_3) d\mathbb{P}(X_3) - \int f(X_1, x_2, X_3) d\mathbb{P}(X_1, X_3) \tag{4.10}$$

Da X_1 und X_3 unabhängig und gleichmäßig verteilt sind, können X_2 und X_3 durch ihre Erwartungswerte (Gleichung 4.8) ersetzt werden:

$$\begin{aligned} \varphi_1^{(1)}(\{x_2\}, f) &= \frac{1}{6}\Big(f(x_1, x_2, \mathbb{E}(X_3)) - f(\mathbb{E}(X_1), x_2, \mathbb{E}(X_3))\Big) \\ &= \frac{1}{6}\Big(f(100, 3, 7.5) - f(125, 3, 7.5)\Big) \\ &= \frac{1}{6}\Big((5 \cdot 100 + 20 \cdot 3 - 2 \cdot 7, 5) - (5 \cdot 125 + 20 \cdot 3 - 2 \cdot 7, 5)\Big) \\ &= \frac{1}{6}(545 - 670) \\ &= \frac{1}{6}(-125) \end{aligned} \tag{4.11}$$

Die in Tabelle 5 dargestellten Kombinationen illustrieren die marginalen Beiträge und SHAP-Werte für jedes Merkmal in jeder möglichen Koalition von Merkmalen, bezogen auf die Beobachtung $x^{(1)}$.

Tabelle 5.: Marginalbeiträge der einzelnen Merkmale zu den möglichen Koalitionen für die Beobachtung $x^{(1)}$.

x_j	$\mathcal{S}$	$v_{f,x^{(1)}}(\mathcal{S})$	$v_{f,x^{(1)}}(\mathcal{S} \cup \{j\})$	$v_{f,x^{(1)}}(\mathcal{S} \cup \{j\}) - v_{f,x^{(1)}}(\mathcal{S})$	Gewicht	$\varphi_j^{(1)}(\mathcal{S}, f)$
x_1	$\emptyset$	680	555	-125	$\frac{1}{3}$	-41,67
x_1	$\{x_2\}$	670	545	-125	$\frac{1}{6}$	-20.83
x_1	$\{x_3\}$	685	560	-125	$\frac{1}{6}$	-20.83
x_1	$\{x_2, x_3\}$	675	550	-125	$\frac{1}{3}$	-41,67
x_2	$\emptyset$	680	670	-10	$\frac{1}{3}$	-3,33
x_2	$\{x_1\}$	555	545	-10	$\frac{1}{6}$	-1,67
x_2	$\{x_3\}$	685	675	-10	$\frac{1}{6}$	-1,67
x_2	$\{x_1, x_3\}$	560	550	-10	$\frac{1}{3}$	-3,33
x_3	$\emptyset$	680	685	5	$\frac{1}{3}$	1,67
x_3	$\{x_1\}$	555	560	5	$\frac{1}{6}$	0,83
x_3	$\{x_2\}$	670	675	5	$\frac{1}{6}$	0,83
x_3	$\{x_1, x_2\}$	545	550	5	$\frac{1}{3}$	1,67

Quelle: Eigene Darstellung.

Diese Analyse ist ebenso auf die Beobachtung $x^{(2)}$ anwendbar und erfordert eine analoge Vorgehensweise:

Die in den Tabellen 5 und 6 dargestellten SHAP-Werte für die Beobachtungen $x^{(1)}$ und $x^{(2)}$ zeigen eine interessante Symmetrie. Für jedes Merkmal entspricht der SHAP-Wert für $x^{(2)}$ genau dem negativen Wert für $x^{(1)}$. Die beobachtete Inversion der SHAP-Werte zwischen den Beobachtungen $x^{(1)}$ und $x^{(2)}$ lässt sich durch die zen-

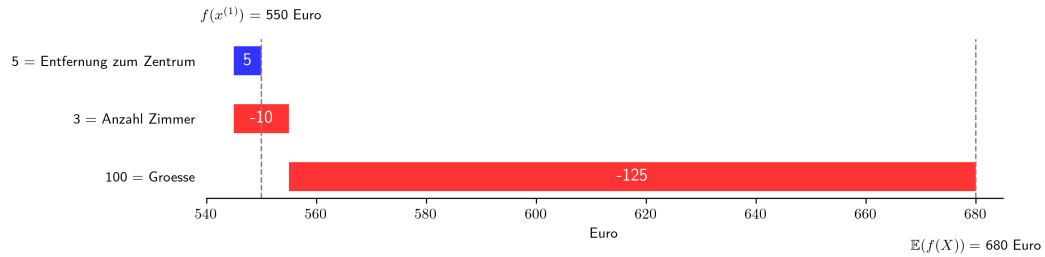
Tabelle 6.: Marginalbeiträge der einzelnen Merkmale zu den möglichen Koalitionen für die Beobachtung $x^{(2)}$.

x_j	$\mathcal{S}$	$v_{f,x^{(2)}}(\mathcal{S})$	$v_{f,x^{(2)}}(\mathcal{S} \cup \{j\})$	$v_{f,x^{(2)}}(\mathcal{S} \cup \{j\}) - v_{f,x^{(2)}}(\mathcal{S})$	Gewicht	$\varphi_j^{(2)}(\mathcal{S}, f)$
x_1	$\emptyset$	680	805	125	$\frac{1}{3}$	41,67
x_1	$\{x_2\}$	690	815	125	$\frac{1}{6}$	20.83
x_1	$\{x_3\}$	675	800	125	$\frac{1}{6}$	20.83
x_1	$\{x_2, x_3\}$	685	810	125	$\frac{1}{3}$	41,67
x_2	$\emptyset$	680	690	10	$\frac{1}{3}$	3,33
x_2	$\{x_1\}$	805	815	10	$\frac{1}{6}$	1,67
x_2	$\{x_3\}$	675	685	10	$\frac{1}{6}$	1,67
x_2	$\{x_1, x_3\}$	800	810	10	$\frac{1}{3}$	3,33
x_3	$\emptyset$	680	675	-5	$\frac{1}{3}$	-1,67
x_3	$\{x_1\}$	805	800	-5	$\frac{1}{6}$	-0,83
x_3	$\{x_2\}$	690	685	-5	$\frac{1}{6}$	-0,83
x_3	$\{x_1, x_2\}$	815	810	-5	$\frac{1}{3}$	-1,67

Quelle: Eigene Darstellung.

trale Rolle des Erwartungswerts des Modells erklären. Das Modell prognostiziert im Mittel einen Immobilienpreis von 680 €. In einem Szenario, in dem nur zwei Beobachtungen vorhanden sind, spiegeln die Beobachtungen $x^{(1)}$ und $x^{(2)}$ entgegengesetzte Abweichungen vom Erwartungswert wider. Die SHAP-Werte, die den Beitrag jedes Merkmals zur Abweichung der Vorhersage vom Mittelwert messen, zeigen daher für $x^{(1)}$ und $x^{(2)}$ genau entgegengesetzte Werte. Dieses Phänomen resultiert aus der Tatsache, dass die Merkmalsbeiträge in Bezug auf den Erwartungswert berechnet werden und unsere beispielhaften Beobachtungen symmetrisch um diesen Mittelwert verteilt sind. Folglich heben sich ihre Beiträge im Kontext unseres linearen Modells exakt auf, was die Konsistenz und Zuverlässigkeit der SHAP-Wert-Berechnung unterstreicht.

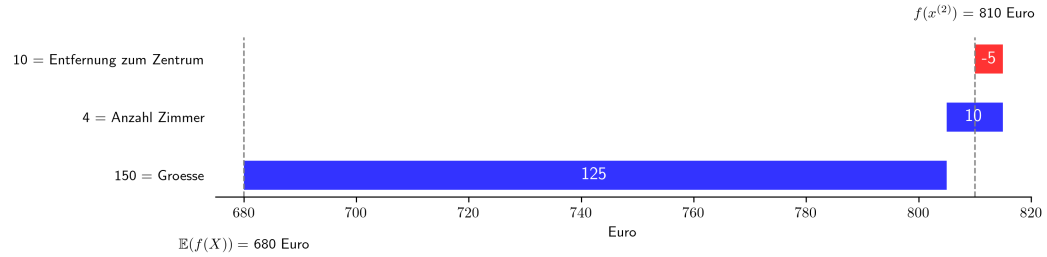
Abbildung 1.: Beitrag der Merkmale $x_{j\in\{1,2,3\}}$ zur Modellvorhersage $f(x^{(1)})$.



Quelle: Eigene Darstellung, A.2.

Das Modell $f(x^{(i)})$ prognostiziert im Mittel einen Immobilienpreis von 680 €. Im Vergleich zur Verteilung des jeweiligen Merkmals, reduziert die Größe der Wohnung ($x_1^{(1)}$) und die Anzahl der Zimmer ($x_2^{(1)}$) die Prognose des Preises für die Immobilie $x^{(1)}$ um insgesamt 135 €, während die Entfernung zum Stadtzentrum ($x_3^{(1)}$) den Preis der Wohnung um 5 € erhöht, wie in Abbildung 1 veranschaulicht. Abbildung 2 visualisiert die SHAP-Werte für die Immobilie $x^{(2)}$:

Abbildung 2.: Beitrag der Merkmale $x_{j\in\{1,2,3\}}$ zur Modellvorhersage $f(x^{(2)})$.



Quelle: Eigene Darstellung, A.2.

4.2. Axiome

Die in Tabellen 5 und 6 präsentierten Ergebnisse bieten eine Grundlage, um die Konformität der SHAP-Werte mit den etablierten Axiomen der Shapley-Werte, wie sie im Kapitel 3.3 diskutiert wurden, zu beurteilen. Die Axiome der SHAP-Werte stellen eine adaptierte und kontextualisierte Anwendung dieser Prinzipien auf die Interpretation von Modellvorhersagen dar [AFSS19, S. 16f].

Effizienz Das Effizienzaxiom besagt, dass die Summe der SHAP-Werte aller Merkmals für eine gegebene Beobachtung $x^{(i)}$ gleich der Differenz zwischen der Modellvorhersage für diese Beobachtung $f(x^{(i)})$ und der durchschnittlichen Modellvorhersage $\mathbb{E}(f(X))$ sein muss:

$$\sum_{j=1}^{p} \varphi_j^{(i)}(\mathcal{N}, f) = f(x^{(i)}) - \mathbb{E}(f(X)), \tag{4.12}$$

[Mol22, S. 221]. Für die Beobachtung $x^{(1)}$ aus Kapitel 4.1 und den Berechnungen für $f(x^{(1)})$ (Gleichung 4.5), sowie $\mathbb{E}(f(X))$ (Gleichung 4.7) ergibt sich:

$$\begin{aligned} \sum_{j=1}^{3} \varphi_j^{(1)}(\mathcal{N}, f) &= -130 \\ f(x^{(1)}) - \mathbb{E}(f(X)) &= 550 - 680 = -130, \end{aligned} \tag{4.13}$$

womit das Effizienzaxiom erfüllt ist. Die Differenz der Vohersage einer konkreten Beobachtung zur durchschnittlichen Modellvorhersage wird auf alle Merkmale verteilt.

Symmetrie Das Symmetrieaxiom fordert, dass zwei Merkmale i und j, die in jeder Koalition denselben Beitrag leisten, auch denselben SHAP-Wert erhalten müssen.

In dem hier betrachteten Fall der Immobilienpreisprognose würde dies bedeuten, dass wenn zwei Merkmale, beispielsweise die Größe einer Wohnung und die Anzahl der Zimmer, immer den gleichen Einfluss auf den Preis hätten, unabhängig von der Kombination anderer Merkmale, ihre SHAP-Werte identisch sein müssen:

$$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S} \cup \{j\}),\ \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{i, j\} \Rightarrow \varphi_i(\mathcal{N}, v) = \varphi_j(\mathcal{N}, v), \tag{4.14}$$

[Mol22, S. 221]. Dies wird durch die Tabellen 5 und 6 nicht illustriert, da jedes Merkmal einen unterschiedlichen Beitrag liefert, was die Anwendung dieses Axioms in diesem speziellen Fall ausschließt.

Null-Spieler-Eigenschaft (Dummy-Spieler-Eigenschaft) Ein Merkmal i, das keinen Einfluss auf die Modellvorhersage hat, erhält gemäß der Null-Spieler-Eigenschaft einen SHAP-Wert von Null. Im Kontext des Beispiels würde ein Merkmal, das keine Veränderung in der Vorhersage bewirkt, unabhängig von den anderen Merkmalen, einen SHAP-Wert von Null erhalten:

$$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S}),\ \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{i\} \Rightarrow \varphi_i(\mathcal{N}, v) = 0, \tag{4.15}$$

[Mol22, S. 222]. In der fiktiven Datenlage der Tabellen 5 und 6 hat jedes Merkmal einen gewissen Einfluss, sodass die Null-Spieler-Eigenschaft hier nicht beobachtet werden kann.

Additivität Das Additivitätsaxiom ist ein zentrales Konzept, das die Konsistenz von Shapley-Werten über die Zusammensetzung von Spielen hinweg beschreibt. Es garantiert, dass für zwei separate Spiele oder Modelle v und w, die Summe der SHAP-Werte eines Merkmals über beide Spiele seinem SHAP-Wert im kombinierten Spiel entspricht:

$$\varphi_i(\mathcal{N}, v + w) = \varphi_i(\mathcal{N}, v) + \varphi_i(\mathcal{N}, w). \tag{4.16}$$

In Machine Learning-Modellen wie dem Random Forest, besonders bei Ensemble-Modellen, ist die Additivität relevant. Ein Random Forest besteht aus unabhängigen Entscheidungsbäumen, die zusammenarbeiten, um Vorhersagen zu treffen. Die SHAP-Werte der einzelnen Bäume können als additive Beiträge betrachtet werden, die den Gesamteinfluss eines Merkmals auf die Modellvorhersage zusammenfassen. [Mol23, S. 32].

5. Komplexitätsbewältigung bei der Berechnung von SHAP-Werten

Die direkte Berechnung von SHAP-Werten kann bei Modellen mit einer großen Anzahl an Merkmalen rechnerisch anspruchsvoll sein. Im Kontext linearer Modelle, wie dem in dieser Arbeit verwendeten Immobilienpreis-Beispiel, ist die vollständige Ermittlung aller möglichen Kombinationen von Merkmalen und deren Beiträgen jedoch nicht notwendig.

In einem linearen Modell sind die Beiträge der einzelnen Merkmale zur Vorhersage additiv und unabhängig voneinander. Dies ermöglicht eine direkte Ableitung der Beiträge jedes Merkmals aus den Modellkoeffizienten. Aufgrund dieser Eigenschaft liefert jedes Merkmal einen konstanten marginalen Beitrag, unabhängig von der Zusammensetzung der restlichen Koalition von Merkmalen. Theoretisch könnte daher für jedes Merkmal nur eine Koalition berechnet werden, um den SHAP-Wert zu bestimmen [Mol23, S. 38]. Dies bestätigen die Tabellen 5 und 6 und die daraus ersichtlichen marginalen Beiträge, die über die jeweiligen Merkmale unabhängig der ihnen beigetretenen Koalition konstant sind. So ist der marginale Beitrag für Merkmal $x_1^{(1)}$ zu allen möglichen Koalitionen stets -125.

Die Herausforderung bei der Berechnung von SHAP-Werten in realen Szenarien erwächst aus zwei zentralen Problembereichen: der Anzahl der Merkmale und der Notwendigkeit, über diese Merkmale zu integrieren.

Die Anzahl möglicher Koalitionen von Merkmalen wächst exponentiell mit 2^p an, was bei einer großen Anzahl von Merkmalen p schnell zu einer nicht handhabbaren Berechnungskomplexität führt. Hinzu kommt die Schwierigkeit, dass für eine exakte Berechnung der SHAP-Werte eine Integration über die Merkmalsverteilungen erforderlich ist, was jedoch voraussetzt, dass die Verteilung der Merkmale bekannt ist. In der Praxis verfügt man meist nur über eine Stichprobe der Daten, ohne genaue Kenntnis der zugrundeliegenden Verteilungen [Mol23, S. 33].

Um diesen Herausforderungen zu begegnen, wird in diesem Kapitel die Anwendung von Monte Carlo Integration zur Schätzung von SHAP-Werten diskutiert. Diese Methoden ermöglichen es, durch Zufallsstichproben aus den vorhandenen Daten eine Approximation der Verteilungen zu erstellen und dadurch die notwendigen Integrationen näherungsweise auszuführen [Mol23, S. 34]. Darüber hinaus werden Ansätze zur Stichprobenbildung von Koalitionen vorgestellt, die es erlauben, die rechnerische Last zu reduzieren, indem nicht alle Koalitionen betrachtet, sondern repräsentative Stichproben gezogen werden.

5.1. Approximation der marginalen Beiträge mittels Monte Carlo Integration

Statt das Integral über eine unbekannte Verteilung zu berechnen, wie in Gleichung 4.1, nähert die Monte Carlo Integration dieses Integral durch den Durchschnitt einer großen Anzahl zufällig ausgewählter Beobachtungen aus dem Eingaberaum an [Mol23, S. 34]. Die Monte Carlo Integration kann als ein erwartungstreuer Schätzer betrachtet werden, wenn die Anzahl der Zufallsstichproben n hinreichend groß ist. Dies bedeutet, dass mit zunehmendem n die Schätzung des Integrals immer genauer wird und gegen den wahren Wert des Integrals konvergiert. Dies basiert auf dem Gesetz der großen Zahlen, das besagt, dass der Durchschnitt einer großen Anzahl unabhängiger und identisch verteilter Zufallsvariablen gegen den Erwartungswert der Verteilung konvergiert [RC04, S. 83].

Die Zufallsvariable $X_C^{(k)}$, die Merkmale die nicht in der Koalition $\mathcal{S}$ enthalten sind, entfällt und wird durch konkrete Beobachtungen $x_C^{(k)}$ aus der Datenbasis ersetzt, was eine wünschenswerte Vereinfachung darstellt. Das Integral $\int$ wird dadurch zur Summe $\sum$ und die Verteilung $\mathbb{P}$ wird durch eine große Anzahl zufällig ausgewählter Beobachtungen ersetzt und das Ergebnis anschließend über alle ausgewählten Beobachtungen n gemittelt. $\hat{v}_{x^{(i)},f}(\mathcal{S})$ ist somit ein Schätzer für den Wert der Koalition $\mathcal{S}$ und definiert als:

$$\hat{v}_{x^{(i)},f}(\mathcal{S}) = \frac{1}{n}\sum_{k=1}^{n}\Big(f(x_{\mathcal{S}}^{(i)} \cup x_C^{(k)}) - f(x^{(k)})\Big) \tag{5.1}$$

Analog zu Gleichung 4.2 ist dann zusammen mit Gleichung 5.1 der marginale Beitrag $\hat{\Delta}_{\mathcal{S},j}$ von Merkmal j zur Koalition $\mathcal{S}$ gegeben als:

$$\begin{aligned}\hat{\Delta}_{\mathcal{S},j} &= \hat{v}_{x^{(i)},f}(\mathcal{S}\cup\{j\}) - \hat{v}_{x^{(i)},f}(\mathcal{S}) \\ &= \frac{1}{n}\sum_{k=1}^{n}\Big(f(x_{\mathcal{S}\cup\{j\}}^{(i)} \cup x_{C\setminus\{j\}}^{(k)}) - f(x_{\mathcal{S}}^{(i)} \cup x_C^{(k)})\Big)\end{aligned} \tag{5.2}$$

und die SHAP-Werte $\hat{\varphi}_j^{(i)}$ über alle möglichen Koalitionen analog zu Gleichung 4.3:

$$\hat{\varphi}_j^{(i)}(\mathcal{N}, f) = \sum_{\mathcal{S}\subseteq\mathcal{N}\setminus\{j\}} \frac{|\mathcal{S}|!\cdot(p-1-|\mathcal{S}|)!}{p!}\hat{\Delta}_{\mathcal{S},j}, \tag{5.3}$$

[Mol23, S.36].

5.2. Schätzung von Koalitionen

Obwohl die Monte Carlo Integration aus Gleichung 5.3 eine praktische Methode zur Approximation der benötigten marginalen Beiträge in der SHAP-Wertberechnung bietet, bleibt die Herausforderung, alle möglichen Koalitionen der Merkmals zu evaluieren. Die Anzahl der Koalitionen wächst exponentiell mit der Anzahl der Merkmals. Daher sind Ansätze gefragt, die eine effiziente Schätzung der SHAP-Werte erlauben, ohne jede mögliche Koalition explizit zu berücksichtigen.

Insbesondere für lineare Modelle ohne Interaktionsterme wurde in Kpaitel 5 bereits gezeigt, dass die Berechnung einer einzigen Koalition für jedes Merkmal ausreichend ist [Mol23, S. 38]. In einem solchen Modell wird angenommen, dass die Beziehung zwischen den unabhängigen Variablen und der abhängigen Variable additiv ist, das heißt, es gibt keine Terme im Modell, die das Produkt von zwei oder mehr unabhängigen Variablen enthalten, um mögliche Interaktionseffekte zwischen ihnen zu repräsentieren.

Neben dem Linearen SHAP Estimator in Kapitel 5.2.1 wird die Schätzung durch Permutationen in Kapitel 5.2.2 diskutirert. Anstelle der Iteration über alle Koallitionen, werden repräsentative Stichproben zur Berechnung herangezogen.

Beide Verfahren bieten Strategien zur Reduzierung der Berechnungskomplexität, indem sie Annahmen über die Eigenschaften des Modells und der Daten treffen. Weitere Ansätze werden in Kapitel 6.2 angesprochen.

5.2.1. Linearer SHAP Estimator für lineare Modelle

Für lineare Modelle bietet der lineare SHAP Estimator eine direkte und effiziente Methode zur Berechnung der SHAP-Werte. Diese Methode beruht auf der Annahme der Unabhängigkeit der Eingabemerkmale. Unter dieser Voraussetzung können die SHAP-Werte direkt aus den Gewichtungskoeffizienten des linearen Modells abgeleitet werden.

Ein lineares Modell wird typischerweise in der Form $f(x) = \sum_{j=1}^{p} \beta_j x_j + b$ ausgedrückt, wobei β_j der Gewichtungskoeffizient für das Merkmal j ist und b den Achsenabschnitt (Bias) darstellt. Der lineare SHAP Estimator nutzt diese Koeffizienten, um die Beiträge jedes Merkmals zur Modellvorhersage zu bestimmen. Der SHAP-Wert für ein Merkmal j wird dann definiert als:

$$\varphi_j^{(i)}(f, x) = \beta_j(x_j^{(i)} - \mathbb{E}[X_j]), \tag{5.4}$$

wobei $\mathbb{E}[X_j]$ der Erwartungswert des Merkmals j über den Datensatz ist. Der SHAP-Wert für den Achsenabschnitt (Bias) ist gleich dem Achsenabschnitt des Modells:

$\varphi_0(f, x) = b$ [LL17, S. 6].

Aufgrund der Unabhängigkeit der Eingabemerkmale, kann das arithmetische Mittel aus Kapitel 4.1 Gleichung 4.8 als erwartungstreuer Schätzer für den Erwartungswert verwendet werden:

$$\varphi_j^{(i)}(f, x) = \beta_j\Big(x_j^{(i)} - \frac{1}{n}\sum_{k=1}^{n}(x_j^{(k)})\Big), \tag{5.5}$$

Das Effizienzaxiom aus Gleichung 4.12 is erfüllt:

$$\begin{aligned}\sum_{j=1}^{p}\varphi_j^{(i)}(f, x) &= \sum_{j=1}^{p}\Big(\beta_j x_j^{(i)} - \mathbb{E}[\beta_j X_j]\Big) \\ &= \beta_0 + \sum_{j=1}^{p}\beta_j x_j^{(i)} - (\beta_0 + \sum_{j=1}^{p}\mathbb{E}[\beta_j X_j]) \\ &= f(x) - \mathbb{E}[f(X)],\end{aligned} \tag{5.6}$$

[Mol23, S. 48].

Diese Berechnung der SHAP-Werte basiert auf der Idee, dass die Änderung des Modelloutputs, die durch das Abweichen eines Merkmals von seinem Erwartungswert entsteht, direkt durch den entsprechenden Gewichtungskoeffizienten des linearen Modells beschrieben werden kann. Dieser Ansatz ermöglicht eine schnelle und genaue Berechnung der SHAP-Werte für lineare Modelle und ist besonders nützlich, um die Auswirkungen einzelner Merkmals in einem solchen Modell zu interpretieren.

5.2.2. Schätzung durch Permutationen

Der Einsatz von Permutationen zur Schätzung von SHAP-Werten bietet eine praktikable Alternative zur vollständigen Auswertung aller Merkmalskoalitionen. Dieses Verfahren ist insbesondere in Szenarien mit einer hohen Anzahl von Merkmalen von Vorteil, da es die Berechnungslast reduziert, ohne die Genauigkeit der Schätzung wesentlich zu beeinträchtigen. Im Folgenden wird die Methode anhand eines Beispiels illustriert und die Anwendung im Kontext des in Kapitel 4.1 eingeführten linearen Modells beschrieben.

In einem ersten Schritt wird eine zufällige k-te Permutation der Merkmale $o(k)$ gewählt. Beispielsweise könnte $o(k) = (x_2, x_3, x_1)$ eine solche Permutation darstellen. Wird das Merkmal j, für das der SHAP-Wert berechnet werden soll, als das dritte Merkmal $j = 3$ angenommen, ergibt sich der marginale Beitrag $\hat{\Delta}_{o(k),j}$ analog

zu Gleichung 5.2 aus der Differenz der Koalitionen mit und ohne dem betrachteten Merkmal:

$$\hat{\Delta}_{o(k),j} = \hat{v}(\{x_2, x_3\}) - \hat{v}(\{x_2\}) \tag{5.7}$$

Dieser Prozess wird für eine Anzahl von m Permutationen durchgeführt. Die Wahl von m, die kleiner als die Gesamtzahl der Merkmale sein kann, hängt von der gewünschten Approximationsgenauigkeit ab. Eine größere Anzahl von Permutationen m führt zu einer präziseren Annäherung an den tatsächlichen SHAP-Wert.

Die geschätzten marginalen Beiträge $\hat{\Delta}_{o(k),j}$ werden dann über die m Permutationen analog zu Gleichung 5.3 gemittelt:

$$\hat{\varphi}_j^{(i)}(\mathcal{N}, f) = \frac{1}{m} \sum_{k=1}^{m} \hat{\Delta}_{o(k),j}, \tag{5.8}$$

Durch diese Methodik wird der Berechnungsaufwand bei der Ermittlung von SHAP-Werten signifikant verringert, was besonders bei Modellen mit einer großen Anzahl von Merkmalen von Bedeutung ist. Der hier vorgestellte Ansatz ermöglicht es, mit einer begrenzten Anzahl von Permutationen eine aussagekräftige Schätzung der SHAP-Werte zu erhalten [Mol23, S. 39].

Zusätzlich zur Mittelung der geschätzten marginalen Beiträge bietet die Methode der Permutationen die Möglichkeit, die Effekte von Vorwärts- und Rückwärtspropagation zu untersuchen, indem die Reihenfolge der Merkmale sowohl in ihrer ursprünglichen als auch in umgekehrter Abfolge betrachtet wird. Für eine detaillierte Darstellung dieser Technik und ihrer Auswirkungen auf die SHAP-Wertberechnung sei auf die weiterführende Literatur verwiesen [Mol23, S. 39f].

6. Praktische Anwendung von SHAP auf lineare Modelle

In diesem Kapitel wird der Einsatz des SHAP-Frameworks zur Interpretation linearer Modelle im Kontext des maschinellen Lernens untersucht. Lineare Modelle, gekennzeichnet durch ihre Transparenz und einfache Struktur, bilden oft die Basis für das Verständnis komplexerer Algorithmen. Dennoch bleibt die Herausforderung bestehen, die Beiträge individueller Merkmale zur Modellvorhersage zu quantifizieren und zu interpretieren.

Die Anwendung von SHAP-Werten ermöglicht es, diesen Herausforderungen zu begegnen und Einblicke in die Modellvorhersagen zu gewähren, die über traditionelle Methoden hinausgehen. Dieses Kapitel führt in die Grundlagen des shap-Pakets ein, demonstriert dessen Anwendung auf einen spezifischen Datensatz und diskutiert die Berechnung sowie Interpretation der resultierenden SHAP-Werte.

6.1. Lineare Modelle als analytische Grundlage

In linearen Regressionsmodellen wird die Zielgröße als eine gewichtete Kombination der Eingangsmerkmale bestimmt. Die einfache lineare Struktur dieser Modelle erleichtert das Verständnis der Beziehungen zwischen den Eingangsdaten und den Vorhersagen.

Lineare Modelle sind ein grundlegendes Werkzeug in der statistischen Modellierung und dienen dazu, das Verhältnis zwischen einer abhängigen Variablen, die üblicherweise mit $y^{(i)}$ bezeichnet wird, und einem oder mehreren Prädiktoren, den unabhängigen Variablen x_i, zu erfassen. Diese Beziehungen werden mittels linearer Gleichungen dargestellt, die für jede einzelne Beobachtung i im Datensatz folgendermaßen formuliert werden können:

$$y^{(i)} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)} + \epsilon^{(i)}, \tag{6.1}$$

wobei das Ergebnis, das von einem linearen Modell für eine gegebene Beobachtung vorhergesagt wird, sich als Summe der mit Gewichten β_j versehenen Merkmale p ergibt.

Hierbei stellt $y^{(i)}$ den beobachteten Wert der abhängigen Variablen für die Beobachtungseinheit i dar. Der Term β_0 ist der Achsenabschnitt oder y-Achsenabschnitt des Modells, welcher den erwarteten Wert von y darstellt, wenn alle unabhängigen

Variablen x null sind. Die Summe $\sum_{j=1}^{p} \beta_j x_j^{(i)}$ berechnet sich aus den Produkten der Koeffizienten β_j und den Werten der unabhängigen Variablen $x_j^{(i)}$ für jede Beobachtungseinheit i und jeden Prädiktor j, wobei die Koeffizienten β_j den geschätzten Einfluss der entsprechenden unabhängigen Variablen auf die abhängige Variable beschreiben.

Der Fehlerterm $\epsilon^{(i)}$ steht für die Residuen, also die Differenzen zwischen den beobachteten und durch das Modell geschätzten Werten von $y^{(i)}$. Es wird angenommen, dass diese Fehler normalverteilt sind, was bedeutet, dass Abweichungen in beiden Richtungen um den Mittelwert (hier Null) mit abnehmender Wahrscheinlichkeit für größere Fehler auftreten [Mol22, S. 37].

In einem linearen Modell stellt der Achsenabschnitt die Basislinie dar, an der die Auswirkungen aller anderen Merkmale gemessen werden. Dieser Wert gibt an, was das Modell für die Zielvariable vorhersagen würde, wenn alle anderen Merkmale nicht vorhanden wären – der Ausgangspunkt der Vorhersage für einen Datensatz, in dem alle anderen Variablen auf null gesetzt sind. Es ist wichtig zu erwähnen, dass der Achsenabschnitt für sich genommen nicht immer eine praktische Bedeutung hat, da es selten vorkommt, dass alle Variablen tatsächlich den Wert null annehmen. Die wahre Aussagekraft des Achsenabschnitts tritt zutage, wenn die Daten so standardisiert wurden, dass ihre Mittelwerte bei null und die Standardabweichung bei eins liegen. Unter diesen Umständen repräsentiert der Achsenabschnitt die erwartete Zielvariable für einen hypothetischen Fall, in dem alle Merkmale ihren Durchschnittswert aufweisen.

Bei der Betrachtung einzelner Merkmale innerhalb des Modells sagt das Gewicht β_j eines Merkmals, um wie viel sich die Zielvariable $y^{(i)}$ ändert, wenn das Merkmal $x_j^{(i)}$ um eine Einheit erhöht wird – und zwar unter der Annahme, dass alle anderen Merkmale unverändert bleiben. Dies ermöglicht es, den isolierten Effekt eines jeden Merkmals auf die Vorhersage zu verstehen [Mol22, S. 39].

Die optimalen Gewichte, oder Koeffizienten, eines linearen Regressionsmodells werden üblicherweise durch ein Verfahren bestimmt, das als Methode der kleinsten Quadrate (engl. *Ordinary Least Squares*, OLS) bekannt ist. Diese Methode sucht die Koeffizienten $\beta_0, \ldots, \beta_p$, welche die Summe der quadrierten Differenzen zwischen den beobachteten Werten der Zielvariablen $y^{(i)}$ und den von dem Modell vorhergesagten Werten minimieren:

$$\hat{\beta} = \arg\min_{\beta_0,\ldots,\beta_p} \sum_{i=1}^{n} \left(y^{(i)} - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)} \right) \right)^2 . \tag{6.2}$$

Das Ergebnis der Minimierung, $\hat{\beta}$ stellt den Vektor der geschätzten Koeffizienten

dar [Mol22, S. 37]. In der vorliegenden Arbeit wird das Python-Paket scikit-learn[1] verwendet, um die lineare Regression durchzuführen und die Koeffizienten $\hat{\beta}$ zu bestimmen.

6.2. Einführung in das shap Python-Paket

Das Python-Paket shap[2] ist eine Open-Source-Bibliothek, die es Nutzern ermöglicht, die Auswirkungen von Merkmalen auf Vorhersagen von maschinellen Lernmodellen zu interpretieren und zu visualisieren. Entwickelt wurde die Bibliothek ursprünglich von Scott Lundberg und weiteren Mitwirkenden im Rahmen der Forschungsarbeit an der University of Washington [LL17]. Das Paket basiert auf dem Konzept der Shapley-Werte aus der kooperativen Spieltheorie und überträgt diese auf den Kontext des maschinellen Lernens, um als Tool für die Interpretierbarkeit und Erklärbarkeit von Modellvorhersagen zu dienen.

Die Kernfunktion des shap-Pakets ist die Berechnung von SHAP-Werten, welche die Auswirkung der Einzelmerkmale auf die Modellvorhersage quantifizieren. Jeder SHAP-Wert ist ein Maß dafür, wie viel jedes Merkmal zur Vorhersage beigetragen hat, im Vergleich zu einer durchschnittlichen Vorhersage über den gesamten Datensatz. Diese Werte sind besonders wertvoll, weil sie ein Maß für die Bedeutung jedes Merkmals liefern, das sowohl lokal (für einzelne Vorhersagen) als auch global (über das gesamte Modell) interpretiert werden kann.

Mit shap können Benutzer die Vorhersagen einer Vielzahl von Modellen interpretieren, von linearen Modellen bis hin zu komplexen Konstrukten wie tiefe neuronale Netzwerke. Die Bibliothek bietet eine vielseitige Auswahl an Visualisierungsoptionen, darunter Beeswarm-Plots, Dependence-Plots und Bar-Plots, die es ermöglichen, die SHAP-Werte intuitiv zu verstehen. Eine Übersicht aller Visualisierungsoptionen ist in der Dokumentation des Pakets zu finden[3]. Diese Visualisierungen erleichtern es, Muster und Beiträge einzelner Merkmale zu erkennen, was nicht nur wertvolle Einblicke in die Leistung des Modells bietet, sondern auch zu faireren und transparenteren Modellentscheidungen führen kann.

Im Kapitel 5.2.1 wurde bereits der LinearExplainer aus dem shap-Paket vorgestellt, ein Beispiel für die verschiedenen Estimators, die das Paket in Form von Explainern bereitstellt. Das Paket bietet eine Vielzahl von Explainern, die auf unterschiedliche Modelltypen zugeschnitten sind. Einer der bemerkenswerten Aspekte von shap ist der auto Modus des Estimators, der automatisch den am besten geeigneten Explainer für das gegebene Modell auswählt. Diese Funktion ist besonders nützlich, da sie die Komplexität der Auswahl des richtigen Explainers reduziert und den Anwendungsprozess vereinfacht. Speziell für Modelle mit ca. 15 Merkmalen[4] wählt der auto

[1] https://scikit-learn.org
[2] https://shap.readthedocs.io
[3] https://shap.readthedocs.io/en/latest/api.html#plots
[4] https://github.com/shap/shap/blob/master/shap/explainers/_exact.py

Modus den exakten Explainer, der präzise SHAP-Werte auf Grundlage aller Daten und Koalitionen berechnet, was für Modelle mit einer geringeren Anzahl von Merkmalen effizient und praktikabel ist [Mol23, S. 40f]. Eine Übersicht der zur Verfügung stehenden shap-Explainern ist in der Dokumentation des Pakets zu finden[5].

6.3. Einführung in den Datensatz

In der vorliegenden Arbeit wird der SGEMM GPU Kernel Performance Datensatz verwendet, welcher die Laufzeitmessung eines Matrix-Matrix-Produkts $A * B = C$ dokumentiert, wobei alle Matrizen die Größe 2048 x 2048 haben. Der Datensatz wurde erstellt, um verschiedene Konfigurationen eines parameterisierbaren SGEMM (Single-precision General Matrix Multiply) GPU-Kernels zu testen. Mit insgesamt 261400 möglichen Parameterkombinationen, von denen jede viermal ausgeführt wurde, bietet der Datensatz eine umfassende Grundlage zur Untersuchung der Laufzeitperformance. Die Daten wurden von Enrique G. Paredes und Rafael Ballester-Ripoll vom Visualization and MultiMedia Lab der Universität Zürich gesammelt [BRPP17, NC15].

Der Datensatz umfasst folgende Variablen [PBR18]:

- **MWG, NWG** (Matrix Widths for Global tile size): Dimensionen der 2D-Aufteilung einer Matrix auf Workgroup-Ebene. Mögliche Werte sind {16, 32, 64, 128}, was die Anzahl der Datenpunkte in jeder Dimension angibt.
- **KWG** (Kernel Width for Global tile size): Innere Dimension der 2D-Aufteilung auf Workgroup-Ebene. Mögliche Werte sind {16, 32}, ebenfalls als Anzahl der Datenpunkte.
- **MDIMC, NDIMC** (Local Workgroup Size): Dimensionen für die lokale Workgroup-Größe. Werte in {8, 16, 32}, repräsentieren die Anzahl der Threads pro Dimension innerhalb einer Workgroup.
- **MDIMA, NDIMB** (Local Memory Shape): Dimensionen, die die Form des lokalen Speichers bestimmen. Werte sind {8, 16, 32} und beziehen sich auf die Größe des Speicherblocks pro Dimension.
- **KWI** (Kernel Workgroup unrolling): Ein Faktor für das Kernel Loop Unrolling. Mögliche Werte sind {2, 8}, die den Grad der Schleifenvereinfachung angeben.
- **VWM, VWN** (Vector Widths for loading/storing): Vektorbreiten für das Laden und Speichern von Matrizen. Werte in {1, 2, 4, 8} repräsentieren die Anzahl der Datenpunkte, die simultan pro Operation gehandhabt werden.
- **STRM, STRN** (Enable stride): Binäre Einstellungen, die festlegen, ob ein Stride für den Speicherzugriff innerhalb eines Threads aktiviert ist. Werte {0,

[5] `https://shap.readthedocs.io/en/latest/api.html#explainers`

1} repräsentieren ausgeschaltet bzw. eingeschaltet.

- **SA, SB** (Manual Caching): Binäre Variablen zur Aktivierung der manuellen Zwischenspeicherung von 2D Workgroup-Kacheln. Werte {0, 1} repräsentieren ausgeschaltet bzw. eingeschaltet.
- **Run1, Run2, Run3, Run4** (Performance Times): Vier unabhängige Laufzeiten (in Millisekunden) für jede Parameterkombination. Sie zeigen die Performanz des GPU-Kernels unter verschiedenen Einstellungen und Bedingungen. Die Zeiten variieren zwischen 13.25 ms und 3397.08 ms.

6.4. Explorative Datenanalyse & Datenaufbereitung

Der Datensatz wurde in Python mithilfe der Bibliothek pandas als Dataframe eingelesen. Die Zielvariable Runtime für die Regression wurde als Durchschnitt der vier unabhängigen Laufzeiten (Run1, Run2, Run3 und Run4) berechnet, um eine repräsentative Maßzahl für die Gesamtperformance zu erhalten.

Der vollständige Quellcode für das Einlesen der Daten sowie alle weiteren Analyseschritte ist im Anhang A.3 dieser Arbeit zu finden.

Tabelle 7 zeigt die ersten fünf Beobachtungen des Datensatzes:

Tabelle 7.: Auszug aus dem SGEMM GPU Kernel Performance Datensatz

Index	MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	STRM	STRN	SA	SB	∅ Runtime (ms)
0	16	16	16	8	8	8	8	2	1	1	0	0	0	0	116.3700
1	16	16	16	8	8	8	8	2	1	1	0	0	0	1	78.7050
2	16	16	16	8	8	8	8	2	1	1	0	0	1	0	80.5650
3	16	16	16	8	8	8	8	2	1	1	0	0	1	1	86.6375
4	16	16	16	8	8	8	8	2	1	1	0	1	0	0	118.6625

Quelle: Eigene Darstellung auf Basis der Datengrundlage [PBR18].

In Tabelle 8 sind deskriptive Statistiken der verschiedenen Variablen des Datensatzes dargestellt.

Die Zielvariable Runtime weist eine erhebliche Variabilität und Spannweite auf. Um eine homogenere Verteilung für die Regression zu erreichen und den Einfluss von Ausreißern zu verringern, wurde sie einer logarithmischen Transformation unterzogen, wie auch von den Datensatzautoren empfohlen [PBR18]. Grafik 3 illustriert die Verteilungen der Runtime mittels Boxplots und Histogrammen, vor und nach der log-Transformation, und verdeutlicht den Effekt der Normalisierung:

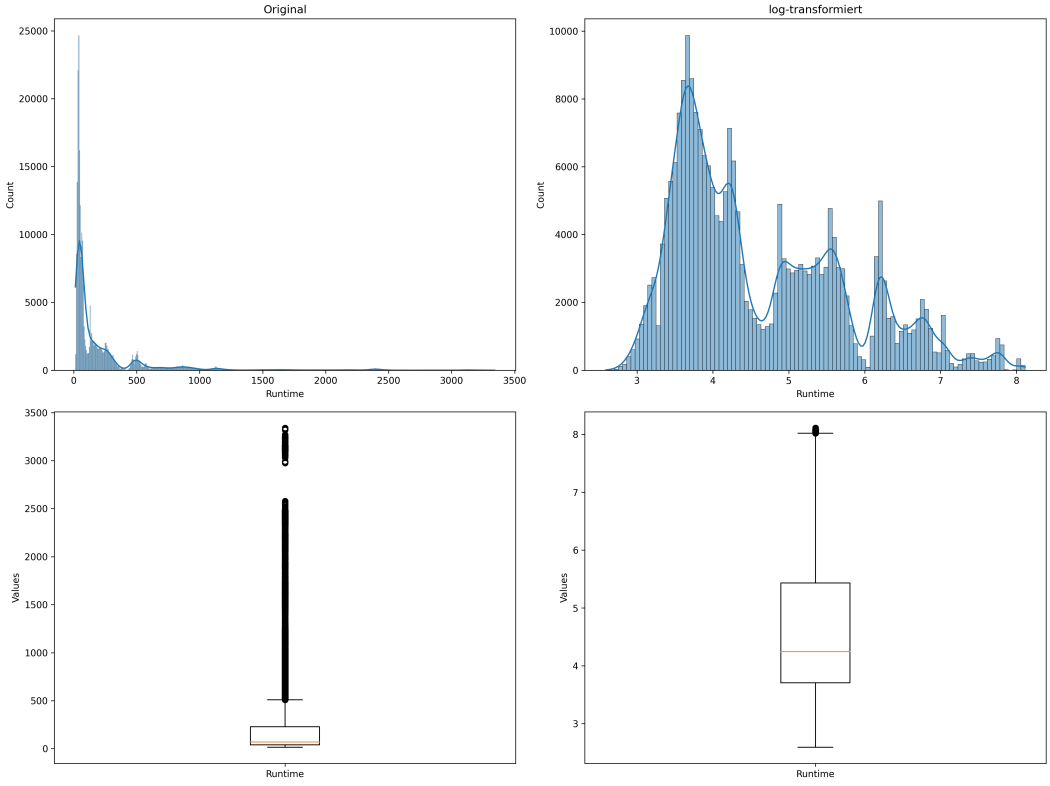
Die Korrelationsmatrix, die in Abbildung 4 dargestellt ist, umfasst die Zielvariable

Tabelle 8.: Auszug aus dem SGEMM GPU Kernel Performance Datensatz

Variable	mean	std	min	25%	50%	75%	max
MWG	80.415364	42.469220	16.0000	32.0000	64.00	128.0000	128.0000
NWG	80.415364	42.469220	16.0000	32.0000	64.00	128.0000	128.0000
KWG	25.513113	7.855619	16.0000	16.0000	32.00	32.0000	32.0000
MDIMC	13.935894	7.873662	8.0000	8.0000	8.00	16.0000	32.0000
NDIMC	13.935894	7.873662	8.0000	8.0000	8.00	16.0000	32.0000
MDIMA	17.371126	9.389418	8.0000	8.0000	16.00	32.0000	32.0000
NDIMB	17.371126	9.389418	8.0000	8.0000	16.00	32.0000	32.0000
KWI	5.000000	3.000006	2.0000	2.0000	5.00	8.0000	8.0000
VWM	2.448609	1.953759	1.0000	1.0000	2.00	4.0000	8.0000
VWN	2.448609	1.953759	1.0000	1.0000	2.00	4.0000	8.0000
STRM	0.500000	0.500001	0.0000	0.0000	0.50	1.0000	1.0000
STRN	0.500000	0.500001	0.0000	0.0000	0.50	1.0000	1.0000
SA	0.500000	0.500001	0.0000	0.0000	0.50	1.0000	1.0000
SB	0.500000	0.500001	0.0000	0.0000	0.50	1.0000	1.0000
Runtime	217.571953	368.750161	13.3175	40.6675	69.79	228.3875	3341.5075

Quelle: Eigene Darstellung, A.3.

Abbildung 3.: Verteilungen der Runtime vor und nach log-Transformation



Quelle: Eigene Darstellung auf Basis der Datengrundlage [PBR18], A.3.

Runtime, wodurch direkte Einblicke in die Beziehungen zwischen den Merkmalen und der Zielvariablen möglich sind. Die Koeffizienten variieren von -0.25 bis 0.46, was darauf hindeutet, dass einige Variablen eine moderate Korrelation mit der Runtime aufweisen. Positive Werte, wie der höchste Koeffizient von 0.46, implizieren, dass eine Erhöhung der entsprechenden Merkmalsausprägungen tendenziell mit längeren Ausführungszeiten verbunden ist. Negative Werte, wie der niedrigste Koeffizi-

ent von -0.25, deuten hingegen auf eine umgekehrte Beziehung hin, bei der höhere Merkmalsausprägungen mit kürzeren Laufzeiten korrelieren. Diese Korrelationen liefern wichtige Informationen für die Modellierung, da sie aufzeigen, welche Parameter möglicherweise einen stärkeren Einfluss auf die Laufzeit haben.

Abbildung 4.: Korrelationsmatrix der Merkmale im Datensatz.

Quelle: Eigene Darstellung, A.3.

6.5. Modellierung der linearen Regression

Um die Beziehung zwischen den unabhängigen Variablen und der Zielvariablen Runtime zu untersuchen, wurde ein lineares Regressionsmodell aufgestellt. Zur Bewertung der Vorhersageleistung des Modells und zur Vermeidung von Overfitting wurde der Datensatz in zwei Teile aufgeteilt: 80% der Daten dienten als Trainingsset zur Anpassung des Modells, während die restlichen 20% als Testset verwendet wurden, um die Modellleistung anhand neuer, unbekannter Daten zu evaluieren. Diese Aufteilung erfolgte zufällig, aber reproduzierbar, durch Festlegen eines Seed-Werts für den Zufallszahlengenerator, der eine konsistente Teilung des Datensatzes ermöglicht.

Das Trainingsset wurde dazu verwendet, die Koeffizienten der linearen Regression zu schätzen, die den Einfluss jeder unabhängigen Variablen auf die Zielvariable quantifizieren. Anschließend wurde das Modell mit dem Testset geprüft, um seine Vorhersagegenauigkeit zu bewerten. Die Leistung des Modells wurde anhand von Metriken

wie dem mittleren quadratischen Fehler (Mean Squared Error, MSE) gemessen, die ein Maß für die Abweichung der Modellvorhersagen von den tatsächlichen Werten darstellen.

Codeauschnitt 6.1 und 6.2 zeigen das Trainieren und Testen der zugrundeliegenden Daten eines linearen Regressionsmodells:

Listing 6.1: Initialisierung eines linearen Regressionsmodells, A.3.

```
def model(X: pd.DataFrame, y: pd.Series) -> (LinearRegression, pd.DataFrame):
    """
    Fits a Linear Regression model to the given data.

    Args:
        X (pd.DataFrame): The feature matrix.
        y (pd.Series): The target variable.

    Returns:
        LinearRegression: The fitted Linear Regression model.
        DataFrame: The coefficients as DataFrame.
    """
    model = LinearRegression()
    model.fit(X, y)

    cdf = pd.DataFrame(model.coef_.round(5), X.columns, columns=['Coefficients'])
    cdf.loc['Intercept'] = model.intercept_.round(5)

    return model, cdf
```

Listing 6.2: Training und Testen eines linearen Regressionsmodells, A.3.

```
X = df.drop(['Runtime'], axis=1)
y = df['Runtime']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
    random_state=2743)

linreg, coef = model(X=X_train, y=y_train)

y_pred = linreg.predict(X_test)
```

6.6. Berechnung von SHAP-Werten

Um SHAP-Werte zu berechnen, wird zunächst ein SHAP-Explainer-Objekt erstellt. In diesem Fall wird der Explainer von SHAP mit dem trainierten linearen Regressionsmodell und dem Trainingsdatensatz initialisiert. Anschließend werden die SHAP-Werte für die Testdaten berechnet, um die Beiträge der einzelnen Merkmale zu analysieren. Der Typ des Explainers wird durch die Art des übergebenen Modells bestimmt. Da in diesem Beispiel ein lineares Modell verwendet wird, wird automatisch ein geeigneter Explainer für lineare Modelle ausgewählt.

Das folgende Codeausschnitt 6.3 zeigt die Initialisierung des SHAP-Explainers und die Berechnung der SHAP-Werte:

Listing 6.3: Berechnung von SHAP-Werten für das lineare Regressionsmodell, A.3.

```
explainer = shap.LinearExplainer(linreg, X_train)
shap_values = explainer(X_test)
```

Das Explainer-Objekt enthält neben den SHAP-Werten (.values), die die Einflüsse der einzelnen Merkmale der Testmenge auf die Modellvorhersage quantifizieren, auch die Basiswerte (.base_values), die die durchschnittliche Vorhersage des Modells darstellen, und die ursprünglichen Merkmalsausprägungen (.data), die für die Berechnung dieser Werte verwendet wurden [Mol23, S. 51].

Dies bildet die Grundlage für den nächsten entscheidenden Schritt: die Visualisierung und tiefere Analyse dieser Werte. Die SHAP-Bibliothek bietet eine Reihe von leistungsstarken Visualisierungswerkzeugen, die es ermöglichen, die Auswirkungen der einzelnen Merkmale auf die Modellvorhersagen intuitiv und verständlich darzustellen.

Im folgenden Kapitel 7 werden diese Visualisierungen im Detail vorgestellt. Anhand von Beeswarm-Plots, Dependence-Plots und Bar-Plots werden die Ergebnisse der SHAP-Analyse dargestellt, die ein umfassendes Bild der Einflüsse und Wichtigkeiten der verschiedenen Merkmale im Kontext des linearen Regressionsmodells bieten.

Die Grafiken wurden mithilfe der shap-Bibliothek wie folgt erzeugt:

Listing 6.4: Erzeugen der SHAP Plots, A.3.

```
    """
    Creates and saves SHAP beeswarm, bar and waterfall plots.

    Args:
        shap_values (shap.Explanation): SHAP values.
        idx (int): Index for the SHAP waterfall plot.
    """
    plt.figure(figsize=(12,10))
    shap.plots.beeswarm(shap_values)
    plt.tight_layout()
    plt.savefig('images/shap_beeswarm_plot_gpu.png', dpi=300)

    plt.figure(figsize=(12,10))
    shap.plots.bar(shap_values)
    plt.tight_layout()
    plt.savefig('images/shap_bar_plot_gpu.png', dpi=300)

    plt.figure(figsize=(12,10))
    shap.plots.waterfall(shap_values[idx])
    plt.tight_layout()
    plt.savefig('images/shap_waterfall_plot_gpu.png', dpi=300)
```

7. Ergebnisse

Das folgende Kapitel präsentiert eine eingehende Analyse der Ergebnisse, die aus dem linearen Regressionsmodell gewonnen wurden, welches entwickelt wurde, um die Runtime basierend auf verschiedenen Parametern aus dem SGEMM GPU Kernel Performance Datensatz zu prognostizieren. Neben der detaillierten Darstellung der Modellkoeffizienten und der zugehörigen Metriken umfasst dieses Kapitel auch eine Vorstellung der SHAP-Werte zur Interpretation der Modellvorhersagen und zur Einsicht in die Wichtigkeit der einzelnen Merkmale.

7.1. Lineares Regressionmodell

Die Koeffizienten des Modells, die in Tabelle 9 aufgeführt sind, zeigen, wie stark sich eine Einheitänderung jedes unabhängigen Merkmals auf die Runtime auswirkt. Positive Koeffizienten deuten auf eine Erhöhung der Runtime bei Zunahme der Variablen hin, während negative Koeffizienten eine Verringerung anzeigen. Der Intercept-Wert repräsentiert die geschätzte Runtime, wenn alle unabhängigen Variablen den Wert Null annehmen. Daraus ergbit sich zusammen mit Gleichung 6.1 die Regressionsgerade für das Modell.

Tabelle 9.: Koeffizienten des linearen Regressionsmodells

Merkmal (β_j)	Koeffizient
Intercept (β_0)	4.23287
MWG	0.01336
NWG	0.01057
KWG	0.01239
MDIMC	-0.05684
NDIMC	-0.05493
MDIMA	0.00020
NDIMB	-0.00003
KWI	-0.00422
VWM	-0.00917
VWN	-0.02371
STRM	-0.13102
STRN	-0.01729
SA	-0.18991
SB	-0.04747

Quelle: Eigene Darstellung, A.3.

Die Modellmetriken, dargestellt in Tabelle 10, geben Auskunft über die Vorhersage-

genauigkeit und die Anpassungsgüte des Modells. Der MAE und RMSE liefern dabei Informationen über die durchschnittliche Größe der Fehler in den Vorhersagen, und die R^2-Werte zeigen, wie gut das Modell die Varianz der Zielvariable erklärt.

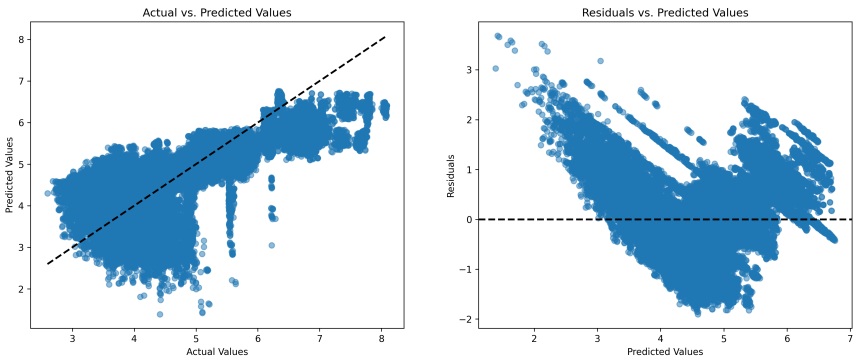
Tabelle 10.: Modellmetriken des linearen Regressionsmodells

Metrik	Wert
Mean Absolute Error (MAE)	0.6000
Mean Squared Error (MSE)	0.5600
Root Mean Squared Error (RMSE)	0.7500
Training Score (R^2)	0.5622
Test Score (R^2)	0.5571

Quelle: Eigene Darstellung, A.3.

Darüber hinaus wurden die tatsächlichen gegen die vorhergesagten Werte der Runtime in Abbildung 5 visualisiert, die eine allgemeine Einschätzung der Modellgenauigkeit ermöglicht. Ein weiterer wichtiger Aspekt sind die Residuen des Modells. Die Residuen, also die Differenzen zwischen den tatsächlichen und vorhergesagten Werten, sollten idealerweise zufällig um Null verteilt sein und keine Muster aufweisen, die auf eine Verletzung der Modellannahmen hindeuten könnten.

Abbildung 5.: Residuenanalyse: Beziehung zwischen Vorhersagen und Abweichungen.



Quelle: Eigene Darstellung, A.3.

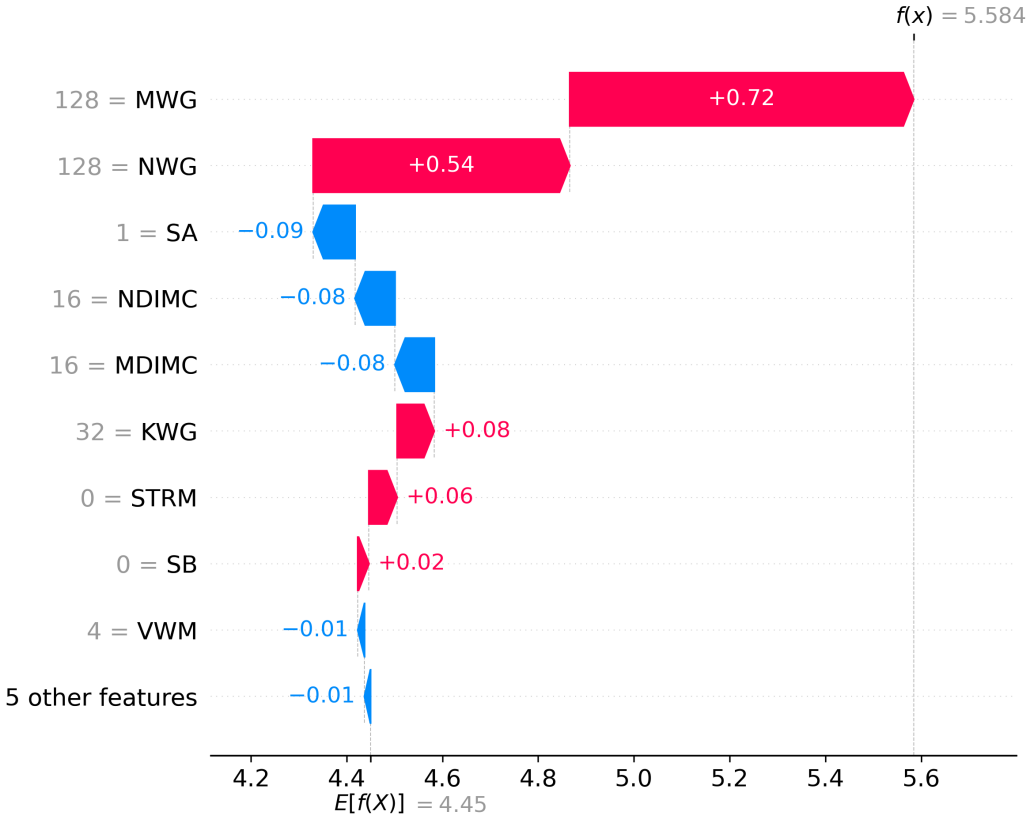
7.2. Interpretation

Die Interpretation der Ergebnisse der Modellanalyse bietet wertvolle Einsichten in die Daten und das Verhalten des linearen Regressionsmodells. Durch die Verwendung von SHAP-Werten wird es möglich, die Beiträge der einzelnen Merkmale zur Vorhersageleistung des Modells nicht nur auf globaler Ebene, sondern auch auf lokaler, individueller Ebene zu verstehen. Diese tiefgehende Analyse ermöglicht es, das Modell auf seine Fairness, Genauigkeit und Transparenz zu überprüfen.

7.2.1. Lokale Interpretation

Die lokale Interpretation konzentriert sich auf das Verständnis der Vorhersagen für eine einzelne Beobachtung aus dem Datensatz. Hierzu wird der SHAP Waterfall Plot eingesetzt, der eine visuelle Darstellung des Beitrags eines jeden Merkmals zu einer spezifischen Vorhersage liefert.

Abbildung 6.: SHAP Waterfall Plot



Quelle: Eigene Darstellung, A.3.

In Abbildung 6 ist ein Waterfall Plot der ersten Beobachtung $x^{(0)}$ dargestellt, der die Zerlegung einer einzelnen Modellvorhersage zeigt. Der Plot beginnt mit dem Basiswert $\mathbb{E}[f(X)] = 4.45$, der durchschnittlichen Vorhersage des Modells.

Von diesem Wert ausgehend, illustrieren die Balken, wie jede Merkmalausprägung – angezeigt durch die grauen Zahlen entlang der y-Achse – die Vorhersage $f(x_j^{(0)})$ beeinflusst. So steigert beispielsweise MWG mit einem Wert von 128 die Vorhersage deutlich um $+0.72$, wohingegen SA mit einem Wert von 1 die Vorhersage um -0.09 verringert.

Rote Balken repräsentieren Merkmale, die die Vorhersage erhöhen, während blaue Balken solche darstellen, die sie senken. Die Größe jedes Balkens zeigt das Ausmaß des jeweiligen Beitrags, und die abschließende Vorhersage $f(x) = 5.584$ wird am Ende der Kette dieser Effekte erreicht.

Kleine positive und negative Beiträge von Merkmalen wie KWG (32), STRM (0), SB (0) und VWM (4) zeigen, wie feingranulare Anpassungen der Merkmalsausprägungen die Vorhersage leicht erhöhen oder senken können.

Für die Beobachtung $x^{(0)}$ führt die kumulative Abweichung der Merkmal-Effekte vom Basiswert $\mathbb{E}[f(X)] = 4.45$ zu einem tatsächlichen Modelloutput von $f(x) = 5.584$, was eine Differenz von $+1.134$ zwischen der durchschnittlichen Vorhersage und der spezifischen Vorhersage für diese Beobachtung offenlegt. Diese Differenz entspricht der Summe aller SHAP-Werte für diese konkrete Beobachtung [Mol23, S. 52f].

Da die Zielgröße einer logarithmischen Transformation unterzogen wurde, muss diese für die Interpreation wieder rückgängig gemacht werden. Dies bedeutet, dass der tatsächliche erwartete Wert der Laufzeit der Exponentialfunktion des prognostizierten Wertes entspricht, also $e^{4.45} \approx 85.63$ ms. Dieser Rücktransformationsprozess ist notwendig, um die Modellprognosen in der ursprünglichen Skala der Zielvariablen zu interpretieren. Dies gilt darüberhinaus sowhl für die einzelnen SHAP-Werte, als auch für die konkrete Vorhersage $f(x) = 5.584$. Die prognostizierte Laufzeit für die Beobachtung $x^{(0)}$ beträgt folglich $e^{5.584} \approx 266.13$ ms.

Dies ermöglicht eine detaillierte Analyse, wie das Modell zu einer bestimmten Vorhersage kommt, und hilft dabei, die Beiträge und Interaktionen zwischen verschiedenen Merkmalen zu verstehen.

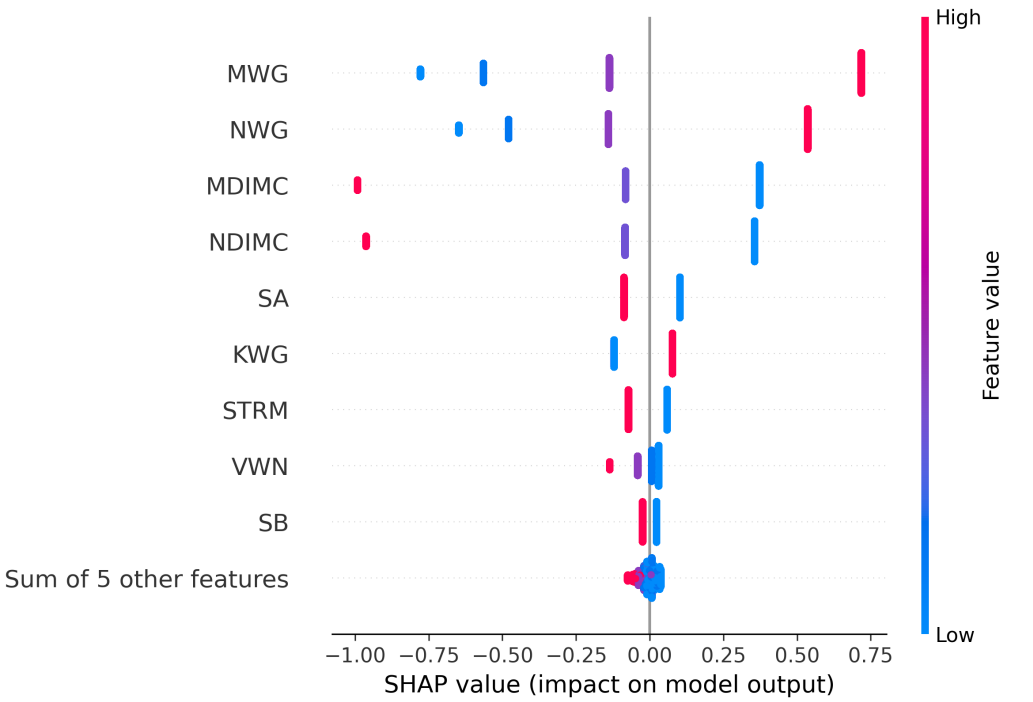
Die lokale Interpretation mittels SHAP-Werten ermöglicht zwar eine präzise Erklärung der Modellvorhersagen für individuelle Beobachtungen, jedoch stellt sich bei einer solchen Betrachtung das Problem der fehlenden Generalisierbarkeit. Lokale Analysen können dazu führen, dass spezifische Merkmal-Kontributionen überinterpretiert werden, ohne die übergeordneten Muster und Einflüsse zu berücksichtigen, die das Modellverhalten im gesamten Datensatz charakterisieren. Eine globale Interpretation ist daher erforderlich, um die Konsistenz und Zuverlässigkeit des Modells über verschiedene Beobachtungen hinweg zu erfassen.

7.2.2. Globale Interpretation

Der folgende Abschnitt widmet sich dieser globalen Sichtweise und untersucht, wie die Merkmal-Beiträge sich im Kontext des gesamten Datensatzes darstellen lassen.

Der SHAP Beeswarm Plot in Abbildung 7 bietet eine globale Sicht auf die Modellvorhersagen, indem er die Verteilung der SHAP-Werte für jedes Merkmals über alle Beobachtungen hinweg darstellt. Jeder Punkt repräsentiert eine Beobachtung aus dem Datensatz. Die Farbe der Punkte zeigt die Merkmalsausprägungen an: hohe Werte in Rot und niedrige Werte in Blau. Die Position auf der x-Achse gibt den Einfluss des Merkmals auf die Modellvorhersage an. Positive SHAP-Werte (rechts von der Nulllinie) zeigen eine Erhöhung der Vorhersage an, während negative Werte (links von der Nulllinie) eine Verringerung bedeuten.

Abbildung 7.: SHAP Beeswarm Plot



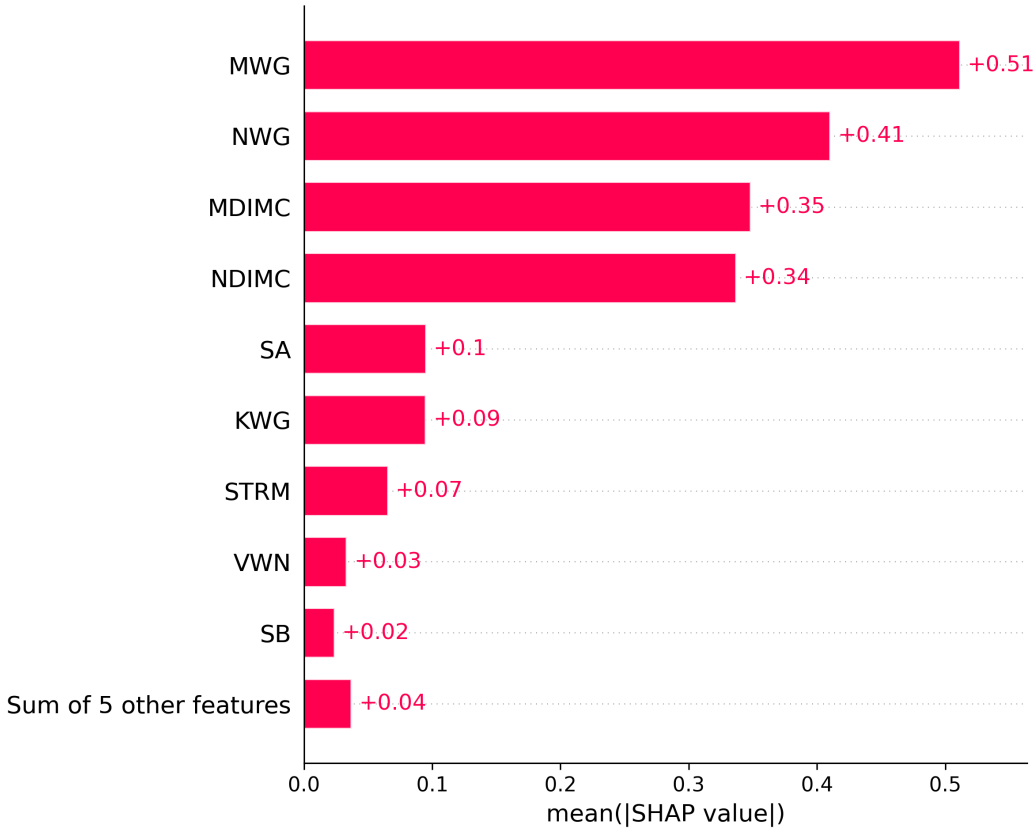
Quelle: Eigene Darstellung, A.3.

Das Merkmal MWG mit den spezifischen Ausprägungen 16, 32, 64 und 128 zeigt eine Variabilität in seinem Einfluss auf die Modellvorhersage. Höhere Werte von MWG, insbesondere 128, sind mit einer Zunahme der Vorhersage (positive SHAP-Werte) assoziiert, was durch die rechtsseitigen Punkte in der Grafik dargestellt wird. iedrigere Werte wie 16 führen hingegen zu einer geringeren Vorhersage, erkennbar an den linksseitigen Punkten. Diese Streuung der Punkte zeigt, dass die Auswirkung von MWG auf die Vorhersage stark von seiner quantitativen Ausprägung abhängt.

Diese Darstellung ermöglicht es, die Merkmale zu identifizieren, die den größten Einfluss auf das Modell haben und wie dieser Einfluss über unterschiedliche Beobachtungen variiert.

Der SHAP Bar Plot in Abbildung 8 illustriert die durchschnittliche Auswirkung jedes Merkmals auf das Modell, gemessen an der absoluten Größe der SHAP-Werte über alle Beobachtungen hinweg. Die Balken zeigen die durchschnittlichen Beiträge der Merkmale zur Vorhersage: Je länger der Balken, desto größer ist der Einfluss des jeweiligen Merkmale. Hier ist das Merkmal MWG mit dem höchsten durchschnittlichen SHAP-Wert (+0.51) das einflussreichste Merkmal, was auf eine starke positive Beziehung zur Zielvariablen hinweist. Die weiteren Merkmale folgen in absteigender Reihenfolge ihrer Bedeutung, wobei auch die Summe der Beiträge der fünf weiteren Merkmale am unteren Rand der Grafik dargestellt wird.

Abbildung 8.: SHAP Bar Plot



Quelle: Eigene Darstellung, A.3.

8. Fazit

Das Fazit der vorliegenden Arbeit reflektiert die erzielten Erkenntnisse aus der Anwendung des linearen Regressionsmodells zur Vorhersage der Laufzeiten von GPU-Kernen. Durch die Integration von SHAP-Werten konnte eine tiefergehende Interpretation der Modellvorhersagen erreicht werden, die über die traditionelle statistische Analyse hinausgeht. Es wurde demonstriert, dass die Koeffizienten des linearen Modells einen direkten Einblick in die Beziehung zwischen den unabhängigen Variablen und der Zielvariable bieten, während die SHAP-Werte es ermöglichen, diese Beziehungen auf einer granularen Ebene zu interpretieren.

Die Visualisierungen, insbesondere der Waterfall- und Beeswarm-Plots, haben eine klare und intuitive Darstellung der Merkmals-Beiträge ermöglicht, die für die Verständlichkeit des Modellverhaltens von entscheidender Bedeutung ist. Die Arbeit hat auch aufgezeigt, dass die lokale Interpretation, obwohl sie präzise und aufschlussreich für einzelne Vorhersagen ist, durch die globale Perspektive ergänzt werden muss, um allgemeingültige Schlussfolgerungen über das Modellverhalten zu ziehen.

Insgesamt hat die Untersuchung gezeigt, dass die Kombination aus linearen Regressionsmodellen und SHAP-basierter Interpretation ein leistungsstarkes Werkzeug darstellt, um sowohl die Vorhersagegenauigkeit als auch die Transparenz und Nachvollziehbarkeit von maschinellen Lernmodellen zu verbessern. Die gewonnenen Einsichten können genutzt werden, um das Modell weiter zu verfeinern, und bieten eine Grundlage für fundierte Entscheidungen bei der Optimierung von GPU-Kerneln.

Gleichzeitig werden jedoch auch die Grenzen und Herausforderungen dieser Methode deutlich. Kritisch betrachtet, können Shapley-Werte in bestimmten Kontexten zu einer Überinterpretation führen, insbesondere wenn die Beziehungen zwischen den Merkmalen komplex und nicht-linear sind. Die Berechnung von Shapley-Werten kann zudem rechnerisch intensiv sein, vor allem bei Modellen mit einer großen Anzahl von Merkmalen. Die Entwicklung effizienterer Algorithmen und die Nutzung fortschrittlicher Rechenressourcen, wie GPU-basiertes Computing, könnten die Anwendbarkeit von SHAP-Werten in der Praxis erheblich erweitern. Dies würde es ermöglichen, auch bei umfangreicheren und komplexeren Modellen eine detaillierte Erklärbarkeit zu gewährleisten.

Die Anwendung von Shapley-Werten im maschinellen Lernen bietet vielfältige Möglichkeiten, die weit über die in dieser Arbeit behandelten Aspekte hinausgehen. Zukünftige Forschungen und Entwicklungen könnten sich in mehrere Richtungen erstrecken, um die Grenzen und Potenziale von SHAP-Werten weiter auszuloten und zu

erweitern.

Eine wichtige Erweiterung der SHAP-Wert-Analyse ist die Einbeziehung von Interaktionen zwischen Merkmalen. Viele reale Datensätze weisen komplexe Beziehungen und Abhängigkeiten zwischen ihren Merkmalen auf. Diese Interaktionen explizit zu modellieren und in die Berechnung der SHAP-Werte zu integrieren, würde zu einer noch genaueren und realistischeren Interpretation der Modellvorhersagen führen.

Literaturverzeichnis

[AFSS19] Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano. Handbook of the shapley value. 2019. URL: https://api.semanticscholar.org/CorpusID:213768429.

[BRPP17] Rafael Ballester-Ripoll, Enrique G. Paredes, and Renato Pajarola. Sobol tensor trains for global sensitivity analysis, 2017. arXiv:1712.00233.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[Mol22] Christoph Molnar. *Interpretable machine learning: A guide for making Black Box models explainable*. Chistoph Molnar c/o Mucbook Clubhouse, Heidi Seibold, 2 edition, 2022.

[Mol23] Christoph Molnar. *Interpreting machine learning models with SAP A guide with python examples and theory on Shapley Values*. Chistoph Molnar c/o MUCBOOK, 1 edition, 2023.

[NC15] Cedric Nugteren and Valeriu Codreanu. Cltune: A generic auto-tuner for opencl kernels. In *2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*. IEEE, September 2015. URL: http://dx.doi.org/10.1109/MCSoC.2015.10, doi:10.1109/mcsoc.2015.10.

[PBR18] Enrique Paredes and Rafael Ballester-Ripoll. SGEMM GPU kernel performance. UCI Machine Learning Repository, 2018. doi:10.24432/C5MK70.

[RC04] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004. URL: https://mcube.lab.nycu.edu.tw/~cfung/docs/books/robert2004monte_carlo_statistical_methods.pdf.

[RWB+22] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. The shapley value in ma-

chine learning. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5572–5579. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Survey Track. doi:10.24963/ijcai.2022/778.

[Sha53] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, Princeton, 1953. URL: https://doi.org/10.1515/9781400881970-018, doi:doi:10.1515/9781400881970-018.

Abbildungsverzeichnis

Tabellenverzeichnis

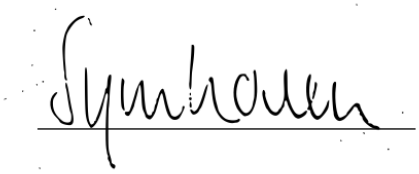
Quellcodeverzeichnis

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen übernommen wurden, sind als solche kenntlich gemacht. Alle Internetquellen sind der Arbeit beigefügt.

Des Weiteren versichere ich, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und dass die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

München, 12. Januar 2024



Simon Symhoven

A. Quellcode

A.1. requirements.txt

```
pandas
matplotlib
scikit-learn
numpy
shap
seaborn
statsmodels
mlxtend
```

A.2. charts.py

```python
import matplotlib.pyplot as plt
from typing import List

plt.rcParams.update({
    'text.usetex': True,
    'text.latex.preamble': r'\usepackage{amsfonts}'
})

def plot_waterfall(expected_value_model: float, observation_values: List[float],
                   observation_labels: List[str], contributions: List[float],
                   name: str, index: int):
    """
    Creates a waterfall chart plot that visualizes the expected model value,
    observation values, their contributions, and labels.

    Parameters:
        - expected_value_model (float): The expected model value.
        - observation_values (List[float]): A list of observation values.
        - observation_labels (List[str]): A list of labels associated with
                                          the observation values.
        - contributions (List[float]): A list of contributions representing
                                       changes from observation values to the
                                       model value.
        - name (str): The name of the plot and the file for saving.
        - index (int): The index for the observation.
    """

    labels = [f'{value} = {label}' for (value, label)
              in zip(observation_values, observation_labels)]
    waterfall = [expected_value_model]
    for contribution in contributions:
        waterfall.append(waterfall[-1] + contribution)

    _, ax = plt.subplots(figsize=(10, 3))

    for idx, (contribution, start) in enumerate(
        zip(contributions, waterfall[:-1])
    ):
```

```
        ax.barh(idx, contribution, left=start, color='blue'
                if contribution >= 0 else 'red', height=0.5, alpha=0.8)
        text_x = start + contribution / 2
        ax.text(text_x, idx, contribution, va='center', ha='center', fontsize=12,
    color='white')

    l1 = r'$\mathbb{E}(f(X)) =$' +f' {expected_value_model} Euro'
    l2 = r'$f(x^{{({index})}})$ = {waterfall_value} Euro'.format(index=index,
    waterfall_value=waterfall[-1])
    ax.text(expected_value_model, -1.2, l1 , va='center', ha='center')
    ax.text(waterfall[-1], 2.8, l2, va='center', ha='center')

    ax.set_yticks(range(len(labels)))
    ax.set_yticklabels(labels)
    ax.set_xlabel('Euro')

    ax.axvline(x=expected_value_model, color='grey', linestyle='--', linewidth=1)
    ax.axvline(x=waterfall[-1], color='grey', linestyle='--', linewidth=1)

    ax.set_xlim(min(waterfall)-5, max(waterfall)+5)

    for spine in ['top', 'right', 'left']:
        ax.spines[spine].set_visible(False)

    plt.tight_layout()
    plt.tick_params(left = False)
    plt.savefig(f'images/{name}', dpi=300)

"""
Create charts for obersavtion x^{(1)} and x^{(2)}
"""
# General Model Values
observation_labels = ['Groesse', 'Anzahl Zimmer', 'Entfernung zum Zentrum']
expected_value_model = 680

# Observation values and contributions for x^{(1)}
index = 1
observation_values_x1 = [100, 3, 5]
contributions_x1 = [-125, -10, 5]

plot_waterfall(expected_value_model=expected_value_model,
               observation_values=observation_values_x1,
               observation_labels=observation_labels,
               contributions=contributions_x1,
               name="model-output-x1.png", index=index)

# Observation values and contributions for x^{(2)}
index = 2
observation_values_x2 = [150, 4, 10]
contributions_x2 = [125, 10, -5]

plot_waterfall(expected_value_model=expected_value_model,
               observation_values=observation_values_x2,
               observation_labels=observation_labels,
               contributions=contributions_x2,
               name="model-output-x2.png", index=index)
```

A.3. linreg.py

```
import requests
```

```
import zipfile
from io import BytesIO
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import shap
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

matplotlib.use('Agg')

def load_data() -> pd.DataFrame:
    """
    Loads and returns the dataset from the given URL as a Pandas DataFrame.

    Returns:
        pd.DataFrame: The loaded dataset.
    """
    url = "https://archive.ics.uci.edu/static/public/440/sgemm+gpu+kernel+
    performance.zip"

    r = requests.get(url)

    if r.ok:
        with zipfile.ZipFile(BytesIO(r.content)) as thezip:
            with thezip.open("sgemm_product.csv") as thefile:
                return pd.read_csv(thefile, sep=",")
    else:
        raise Exception("Something went wrong.")

def model(X: pd.DataFrame, y: pd.Series) -> (LinearRegression, pd.DataFrame):
    """
    Fits a Linear Regression model to the given data.

    Args:
        X (pd.DataFrame): The feature matrix.
        y (pd.Series): The target variable.

    Returns:
        LinearRegression: The fitted Linear Regression model.
        DataFrame: The coefficients as DataFrame.
    """
    model = LinearRegression()
    model.fit(X, y)

    cdf = pd.DataFrame(model.coef_.round(5), X.columns, columns=['Coefficients'])
    cdf.loc['Intercept'] = model.intercept_.round(5)

    return model, cdf

def plot_residuals(y_test: pd.Series, y_pred: pd.Series) -> None:
    """
    Creates a residual plots.

    Args:
        y_test (pd.Series): The actual target values.
        y_pred (pd.Series): The predicted target values.
    """
```

```
62      residuals = y_test - y_pred
63
64      _, axs = plt.subplots(1, 2, figsize=(16, 6))
65
66      axs[0].scatter(y_test, y_pred, alpha=0.5)
67      axs[0].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--k',
         linewidth=2)
68      axs[0].set_title('Actual vs. Predicted Values')
69      axs[0].set_xlabel('Actual Values')
70      axs[0].set_ylabel('Predicted Values')
71
72      axs[1].scatter(y_pred, residuals, alpha=0.5)
73      axs[1].axhline(y=0, color='k', linestyle='--', linewidth=2)
74      axs[1].set_title('Residuals vs. Predicted Values')
75      axs[1].set_xlabel('Predicted Values')
76      axs[1].set_ylabel('Residuals')
77
78      plt.savefig('images/residuals_gpu.png', dpi=300)
79
80  def plot_corr(df: pd.DataFrame) -> None:
81      """
82      Creates and saves a correlation heatmap plot.
83
84      Args:
85          df (pd.DataFrame): The feature matrix.
86      """
87      plt.figure(figsize=(12,10))
88      sns.heatmap(df.corr(), annot=True, cmap="YlGnBu", fmt=".2f")
89      plt.savefig('images/corr_gpu.png', dpi=300)
90
91  def plot_shap(shap_values: shap.Explanation, idx: int) -> None:
92      """
93      Creates and saves SHAP beeswarm, bar and waterfall plots.
94
95      Args:
96          shap_values (shap.Explanation): SHAP values.
97          idx (int): Index for the SHAP waterfall plot.
98      """
99      plt.figure(figsize=(12,10))
100     shap.plots.beeswarm(shap_values)
101     plt.tight_layout()
102     plt.savefig('images/shap_beeswarm_plot_gpu.png', dpi=300)
103
104     plt.figure(figsize=(12,10))
105     shap.plots.bar(shap_values)
106     plt.tight_layout()
107     plt.savefig('images/shap_bar_plot_gpu.png', dpi=300)
108
109     plt.figure(figsize=(12,10))
110     shap.plots.waterfall(shap_values[idx])
111     plt.tight_layout()
112     plt.savefig('images/shap_waterfall_plot_gpu.png', dpi=300)
113
114 def plot_dist(df: pd.DataFrame):
115     """
116     Creates and saves distribution and histrogramm plot over all
117     features of df.
118
119     Args:
120         df (pd.DataFrame): The feature matrix.
121     """
```

```
122     fig, axes = plt.subplots(5, 3, figsize=(15, 20))
123     axes = axes.flatten()
124 
125     for i, var in enumerate(df.keys()):
126         sns.histplot(df[var], ax=axes[i], kde=True)
127         axes[i].set_title(var)
128 
129     plt.tight_layout()
130     plt.savefig('images/dist_gpu.png', dpi=300)
131     plt.show()
132 
133 
134 def plot_target_dist(df: pd.DataFrame, original_runtime: pd.Series):
135     """
136     Creates and saves distribution and histrogramm plot for targets variable
137     before log transformation and after.
138 
139     Args:
140         df (pd.DataFrame): The feature matrix.
141     """
142     fig, axs = plt.subplots(2, 2, figsize=(16, 12))
143 
144     sns.histplot(original_runtime, ax=axs[0, 0], kde=True)
145     axs[0, 0].set_ylabel("Count")
146     axs[0, 0].set_title("Original")
147 
148     axs[1, 0].boxplot(original_runtime)
149     axs[1, 0].set_ylabel("Values")
150     axs[1, 0].set_xticks([1], ['Runtime'])
151 
152     sns.histplot(df["Runtime"], ax=axs[0, 1], kde=True)
153     axs[0, 1].set_ylabel("Count")
154     axs[0, 1].set_title("log-transformiert")
155 
156     axs[1, 1].boxplot(df["Runtime"])
157     axs[1, 1].set_ylabel("Values")
158     axs[1, 1].set_xticks([1], ['Runtime'])
159 
160     plt.tight_layout()
161     plt.savefig('images/combined_runtime_plots.png', dpi=300)
162     plt.show()
163 
164 
165 df = load_data()
166 
167 
168 # Preprocessing
169 mean_values = df.iloc[:, 14:18].mean(axis=1)
170 df.insert(14, "Runtime", mean_values)
171 df.drop(['Run1 (ms)', 'Run2 (ms)', 'Run3 (ms)', 'Run4 (ms)'], axis='columns',
        inplace=True)
172 
173 print(df.head())
174 print(df.isnull().sum())
175 print(df.describe().T)
176 
177 original_runtime = df["Runtime"]
178 df["Runtime"] = np.log(df["Runtime"])
179 
180 plot_target_dist(df=df, original_runtime=original_runtime)
181 plot_dist(df=df)
```

```
182
183 X = df.drop(['Runtime'], axis=1)
184 y = df['Runtime']
185
186 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
        random_state=2743)
187
188 linreg, coef = model(X=X_train, y=y_train)
189
190 y_pred = linreg.predict(X_test)
191 explainer = shap.LinearExplainer(linreg, X_train)
192 shap_values = explainer(X_test)
193
194 mae = mean_absolute_error(y_test, y_pred)
195 mse = mean_squared_error(y_test, y_pred)
196 train_score = linreg.score(X_train, y_train)
197 test_score = linreg.score(X_test, y_test)
198
199 print(f"Mean Absolute Error (MAE): {mae:.2f}")
200 print(f"Mean Squared Error (MSE): {mse:.2f}")
201 print(f"Root Mean Squared Error (RMSE): {mse ** 0.5:.2f}")
202 print(f"Training Score (R^2): {train_score:.4f}")
203 print(f"Test Score (R^2): {test_score:.4f}")
204
205 print(coef)
206 plot_corr(X=df)
207 plot_residuals(y_test=y_test, y_pred=y_pred)
208 plot_shap(shap_values=shap_values, idx=0)
```