Programmaggregation mit algebraischen Entscheidungsdiagrammen[1]

Frederik Gossen[2]

Abstract: Im Rahmen dieser Dissertation wurde das Potential von algebraischen Entscheidungsdiagrammen (ADDs) als Programmrepräsentation für deren Optimierung untersucht. Dabei wurden domänenspezifische Sprachen, maschinell erlernte Modelle und allgemeine Programmiersprachen betrachtet. Insbesondere die Anwendung auf Random Forests, eine Methode des klassischen maschinellen Lernens, war besonders erfolgreich. Hier konnten nicht nur Beschleunigungen von mehreren Größenordnungen erreicht werden, sondern auch gleich drei wichtige Erklärbarkeitsprobleme gelöst werden. Random Forests, die als nicht interpretierbare Black-Box-Modelle angesehen werden, können so semantisch aggregiert und verständlicher dargestellt werden. Das Resultat der Aggregation kann als semantisch äquivalentes White-Box-Modell angesehen werden. Die Lösung der Erklärbarkeitsprobleme ist beispielsweise in der Medizin oder im Bankensektor von enormer Bedeutung. Hier müssen automatisierte Entscheidungen immer erklärbar sein.

1 Einführung

Random Forests sind eine der beliebtesten Klassifikationsmethoden im klassischen maschinellen Lernen [Br01]. Sie finden Anwendung in den verschiedensten Bereichen, sei es im Bankensektor, im Gesundheitswesen [Ch21] oder im Web [Fa17], und sie helfen wichtige Entscheidungen zu automatisieren. Anstatt Kriterien für Entscheidungen manuell zu formalisieren, lernen Random Forests Muster automatisch und anhand von Beispieldaten. Auf diese Weise kann beispielsweise die Reaktion von Patienten auf eine Krebsbehandlung anhand ihrer DNA vorhergesagt werden [Ch21]. Obwohl Zusammenhänge wie dieser im Allgemeinen kaum oder gar nicht verstanden sind, können so gute Entscheidungen automatisch getroffen werden. Gleichzeitig birgt die Anwendung in diesen Bereichen ein Risiko, nämlich genau dann, wenn diese maschinell getroffenen Entscheidungen nicht mehr nachvollziehbar sind und man potentielle Fehler nicht erkennen kann. Je nach Anwendungsfall, zum Beispiel in der Medizin, kann dies aber enorm wichtig sein. Leider sind viele der erfolgreichsten Methoden im maschinellen Lernen in genau diesem Sinne nicht interpretierbar. Man nennt diese auch Black-Box-Modelle. Zu ihnen gehören auch die hier betrachteten Random Forests.

Durch holistische Aggregation von Random Forests ist es im Rahmen dieser Dissertation [Go21] gelungen deren Semantik zu aggregieren und in ein sehr viel verständlicheres und kompaktes Modell zu überführen. Dieses Resultat kann als Lösung des allgemeinen

[1] Englischer Titel der Dissertation: „Aggressive Aggregation - Domain-specific program optimization with Algebraic Decision Diagrams“

[2] Fakultät für Informatik, TU Dortmund, frederik.gossen@tu-dortmund.de

Erklärungsproblems für Random Forests gesehen werden. Die Aggregation erlaubt es sogar noch einen Schritt weiterzugehen und löst insgesamt gleich drei verschiedene Entscheidungsprobleme:

- Das allgemeine *Modell-Erklärungsproblem* [Gu19] fordert eine Erklärung des Klassifikationsmodells als Ganzes, hier des Random Forests. Das gelernte Modell soll so dargestellt werden, dass es für Experten verständlich ist und sie aus dem maschinell Gelernten Erkenntnisse ziehen können.
- Das *Klassen-Erklärungsproblem* [GS21] ist eine Reduktion des *Erklärungsproblems*. Es fordert eine Erklärung des Klassifikationsmodells bzgl. einer einzelnen Klasse, also zum Beispiel, ob ein Patient auf eine Krebsbehandlung besonders gut reagiert oder nicht.
- Das *Ergebnis-Erklärungsproblem* [Gu19] betrachtet eine konkrete Anwendung des gelernten Modells und fordert nur eine Erklärung für eine in einem bestimmten Fall getroffene Entscheidung.

Die ursprüngliche Zielsetzung dieser Arbeit war es, das Potential von algebraischen Entscheidungsdiagrammen (ADDs) [Ba97, Br86, Ak78] als Programmrepräsentation in Compilern zu untersuchen. Die Datenstruktur ist allgemein sehr gut verstanden und bekannt für ihre Optimalität in Größe und Tiefe [Ba97]. Im Kontext von Compilern entsprechen diese der Größe und der Laufzeit von Programmen, zwei klassische Ziele in der Programmoptimierung. Leider ist es nicht möglich allgemeine Programmiersprachen in diese Datenstruktur zu übersetzen. Beschränkt man allerdings die Domäne der betrachteten Programme auf solche, für die man Transformationen finden kann, so können die bekannten Algorithmen und Eigenschaften von ADDs ausgenutzt werden, um beeindruckende Laufzeitoptimierungen zu realisieren. Diese domänenspezifischen Programme können so teilweise um mehrere Größenordnungen beschleunigt werden. In Anwendungen mit enorm hohen Bandbreiten, kann dieser Effizienzgewinn enorm wertvoll sein [Fa17].

Im Rahmen dieser Dissertation wurde das Potential von ADDs als Programmrepräsentation in drei Domänen untersucht:

- Grafische und textuelle domänenspezifische Sprachen, die speziell für ADDs entworfen sind [St19, Go18] (siehe Abb. 1),
- Maschinell gelernte Modelle bzw. Programme am Beispiel von Random Forests [GS21, GMS20, GMS21] und
- Allgemeine Programmiersprachen am Beispiel der *while*-Sprache [Go19].

Im Folgenden werden wir genauer auf die Aggregation von Random Forests und die daraus resultierenden Vorteile eingehen. Wir werden sehen, wie selbst große Random Forests in ein einzelnes algebraisches Entscheidungsdiagramm überführt werden können und wie sukzessive Abstraktion den Random Forest gleichzeitig erklären und seine Auswertung enorm beschleunigen kann.

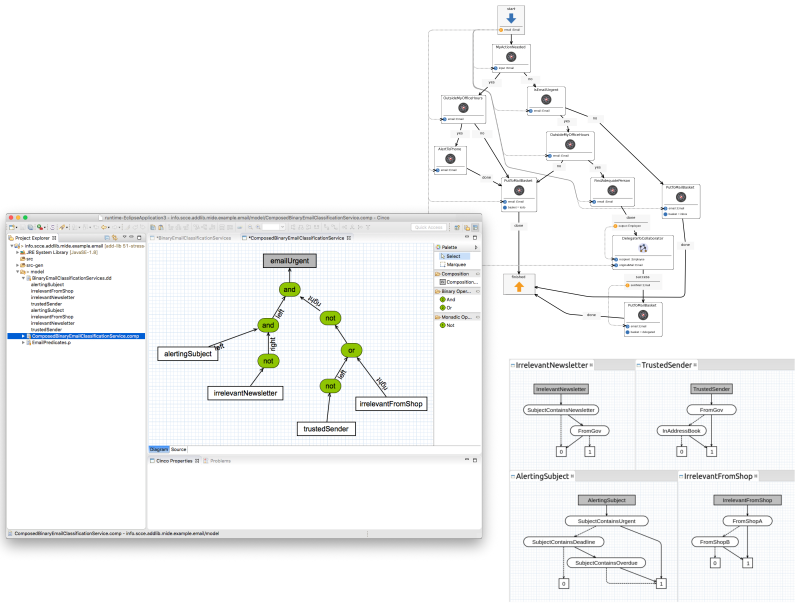

Abb. 1: Grafische ADD-basierte domänenspezifische Programmiersprachen und die darauf zugeschnittenen Entwicklungsumgebungen [St19].

2 Aggregation eines Random Forests

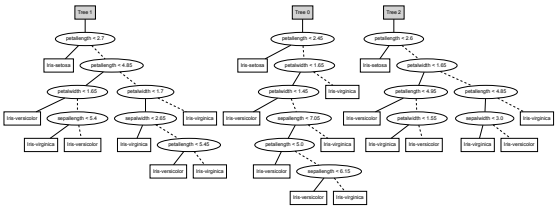

Abb. 2: Random Forest mit drei Bäumen, der auf dem Iris-Datensatz [Fi36] trainiert wurde. (41 Knoten)

Die erste Repräsentation ist der originale Random Forest wie er gelernt wurde [Br01]. Abbildung 2 zeigt einen solchen, der mit einer Standardmethode auf dem Iris-Datensatz [Fi36] trainiert wurde und im Folgenden als Beispiel dienen soll. Das Klassifikationsproblem des Datensatzes ist ein beliebtes Beispiel, um Methoden im maschinellen Lernen anschaulich darzustellen. Die Aufgabe ist es, anhand verschiedener Maße einer Irisblüte, die Korrekte von drei Spezies zu bestimmen. Für ein neues noch nicht klassifiziertes Beispiel sind also diese Maße gegeben und wir können jeden der drei Entscheidungsbäume von der Wurzel an auswerten, pro innerem Knoten den entsprechenden Nachfolger wählen und in den Blättern eine Vorhersage finden. Die Klassifikation des gesamten Random Forests

ist dann die am häufigsten gewählte Spezies, das Ergebnis des so genannten „majority votes“ [Br01]. [3]

Die Aggregation eines beliebig großen Random Forests zu einem einzelnen algebraischen Entscheidungsdiagramm (ADD) [Ba97] erfolgt in drei Schritten. Zunächst werden die Entscheidungsbäume des Random Forests elementweise in ADDs überführt. Von hier an kann die algebraische Natur und die Kompositionalität der ADD-basierten Repräsentation voll ausgenutzt werden um die vielen ADDs in einen einzelnen zu kondensieren. Der so resultierende ADD kann dann im dritten Schritt durch Abstraktion weiter optimiert werden, indem er zur Compile-Zeit maximal ausgerechnet wird und unerfüllbare Pfade eliminiert werden.

Es gibt zwei wesentliche Unterschiede zwischen den Entscheidungsbäumen des Random Forests (Abb. 2) und äquivalenten ADDs, die wir im ersten Schritt ableiten wollen. Während Prädikate zunächst in beliebiger und sogar verschiedener Reihenfolge in den unterschiedlichen Bäumen vorkommen können, fordern ADDs eine feste und einheitliche Reihenfolge ein. Darüber hinaus werden isomorphe Knoten in ADDs zu einem verschmolzen; jeder Knoten ist also eindeutig. Beide Invarianten lassen sich effizient umsetzen, sodass aus dem ursprünglichen Random Forest ein semantisch äquivalenter Forest aus ADDs wird.

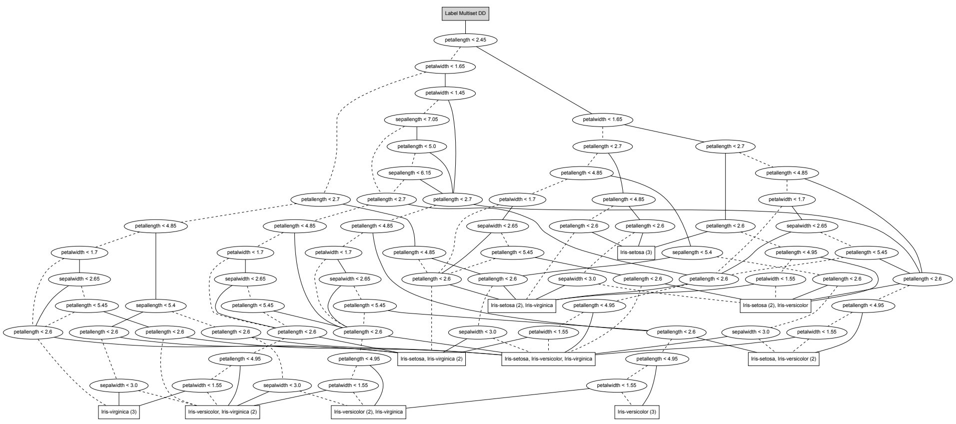

Abb. 3: Als Vektor-ADD aggregierter Random Forest. Dieser ADD ist semantisch äquivalent zu dem Random Forest in Abb. 2. (79 Knoten)

Um diese Menge von ADDs zu einem einzelnen zu aggregieren ist eine kompositionelle Darstellung notwendig. Die abstrakteste, noch kompositionale Darstellung, sind in diesem Fall Klassen-Vektoren, die eine ganzzahlige Dimension pro Klasse enthalten. Auf diese Weise lässt sich zählen, wie oft jede Klasse von einem der ursprünglichen Bäume gewählt wurde. Die Menge von ADDs lässt sich so einfach elementweise übersetzen, nämlich zu solchen ADDs, die anstatt einer einzelnen Klasse entsprechende Einheitsvektoren in ihren Blättern enthalten. Durch Summation dieser Klassen-Vektoren, lassen sich Entscheidungen der einzelnen ADDs auf natürliche Weise aggregieren. Die Datenstruktur erlaubt es

[3] Uneindeutigkeiten werden durch beliebige Priorität der Spezien gelöst.

aber auch, diese Aggregation auf der Metaebene der ADDs selbst durchzuführen. So werden die ADDs effektiv aufsummiert und der gesamte Random Forest wird in einem einzelnen, semantisch äquivalenten ADD dargestellt. Anstatt sämtliche Bäume des Random Forests auswerten zu müssen, reicht es nun, einen einzelnen ADD von seiner Wurzel an zu traversieren. Das Ergebnis ist ein Vektor, der die Wahlhäufigkeiten pro Klasse widerspiegelt. Die am häufigsten gewählte Klasse lässt sich daraus direkt ableiten. Abbildung 3 zeigt den aggregierten Vektor-ADD für den vorangegangenen Random Forest (Abb. 2).

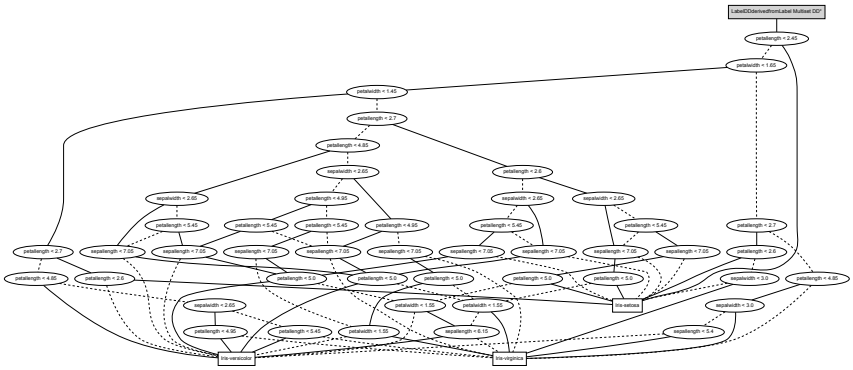

Abb. 4: Als Klassen-ADD aggregierter Random Forest. Dieser ADD wurde auf die Erfüllbarkeit seiner Pfade gefiltert und ist semantisch äquivalent zu dem Random Forest in Abb. 2 und dem ADD in Abb. 3. (50 Knoten)

Auch wenn die Klassen-Vektoren für die Aggregation unverzichtbar sind, weil sie die Kompositionalität der ADDs garantieren, so enthalten sie für die Auswertung des Modells unnötig viel Information. Tatsächlich sind wir nur an der am häufigsten gewählten Klasse interessiert, einer einfachen Abstraktion der Klassen-Vektoren. Wie zuvor, lässt sich auch diese Operation auf die Metaebene der ADDs heben und wir können den gesamten ADD so transformieren, dass seine Blätter direkt das gewünschte Ergebnis enthalten. Auf diese Weise ist es möglich den „majority vote“ schon zur Compile-Zeit auszurechnen. Der Effekt ist größer als er auf den ersten Blick scheint, denn die Abstraktion führt dazu, dass vorher verschiedene Blätter verschmolzen werden. Der ADD kollabiert so von seinen Blättern her und kann deutlich kleiner sein als zuvor.

Ein weiterer Aspekt für die Optimierung sind unerfüllbare Pfade. Weil ADDs symbolischer Natur sind und ihre Optimalitätsgarantien auf der Unabhängigkeit von Prädikaten beruhen, finden sich in dem aggregierten ADD unerfüllbare Pfade. So kann es passieren, dass sich auf einem Pfad widersprüchliche Bedingungen finden, wie zum Beispiel $petalwidth < 3$ und $\neg(petalwidth < 5)$. Diese Unerfüllbarkeiten können durch Intervallpropagation aus dem ADD eliminiert werden. Es ist so möglich den ADD weiter zu vereinfachen und seine Größe und Tiefe weiter zu reduzieren.

Abbildung 4 zeigt den aggregierten und semantisch äquivalenten ADD für den ursprünglichen Random Forest (Abb. 2). Diese Repräsentation aggregiert die Semantik auf eine Weise, die es erlaubt, den Random Forest (i) extrem schnell auszuwerten und (ii) seine Semantik besser zu verstehen.

3 Auswirkungen auf die Laufzeit

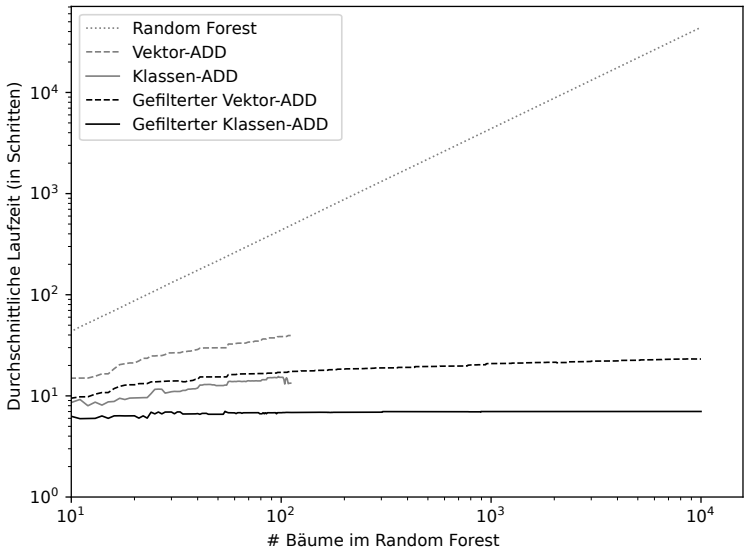

Abb. 5: Laufzeiten der verschiedenen Repräsentationen eines Random Forests mit bis zu 10.000 Bäumen (gemessen in Schritten).

Das ursprüngliche Ziel dieser Dissertation war es die Laufzeit von Random Forests zu reduzieren. Der aggregierte ADD erreicht dieses Ziel und kann die Laufzeit in einigen Fällen um mehrere Größenordnungen reduzieren.

Abbildung 5 zeigt die Auswirkungen der Repräsentation auf die Laufzeit für das vorangegangene Beispiel des Iris-Datensatzes. Anstatt eines Random Forests mit nur 3 Bäumen, betrachten wir hier Forests mit 1 bis 10.000 Bäumen. Die Laufzeit wird in Schritten durch die Datenstruktur gemessen, die für die Beispiele des Datensatzes durchschnittlich erforderlich sind.

Die durchschnittliche Laufzeit des originalen Random Forests wächst linear mit der Anzahl an Bäumen, weil jeder separat ausgewertet werden muss. Im Gegensatz dazu ist die Auswertung der aggregierten ADDs um bis zu drei Größenordnungen schneller. Für, auf unerfüllbare Pfade gefilterte, Klassen-ADDs scheint die Laufzeit sogar zu konvergieren.

Das Filtern auf Unerfüllbarkeit hat hier zweierlei Auswirkung: es hält die Datenstruktur klein und reduziert gleichzeitig deren Tiefe. Das führt einerseits zu schnelleren Laufzeiten, ermöglicht es andererseits aber überhaupt erst die Aggregation für größere Random Forests zu berechnen. Das ist auch der Grund dafür, dass die Laufzeit der ungefilterten ADDs nur bis zu einer Größe von 100 Bäumen untersucht wurde.

Die Ergebnisse des Iris-Datensatzes lassen sich auch auf anderen Datensätzen des UCI Machine Learning Repository [DG17] reproduzieren. Tabelle 1 zeigt durchschnittliche Laufzeiten für Random Forests mit 1.000 Bäumen auf anderen Datensätzen.

Datensatz	Originaler Random Forest	Aggregierter ADD
Balance Scale	8.014,12	7,73 (-99,90%)
Breast Cancer	13.020,03	17,11 (-99,87%)
Lenses	4.431,42	3,67 (-99,92%)
Iris	4.395,77	6,97 (-99,84%)
Tic-Tac-Toe	10.733,68	14,22 (-99,87%)
Vote	6.921,56	8,33 (-99,88%)

Tab. 1: Durchschnittliche Laufzeit für die Klassifikation (gemessen in Schritten). Betrachtet wird hier ein Random Forest mit 1.000 Bäumen, der jeweils auf einem Datensatz aus dem UCI Machine Learning Repository [DG17] gelernt wurde.

4 Lösungen der Erklärungsprobleme

Die Aggregation mittels algebraischer Entscheidungsdiagramme ist semantik-erhaltend und löst neben Laufzeitverbesserungen ein Problem einer weiteren Disziplin: *Erklärbarkeit* [Gu19]. Während Random Forests als Black-Box-Modelle nicht interpretierbar sind, kondensiert die Aggregation mit ADDs deren Semantik und stellt diese kompakt dar. Ähnlich wie Entscheidungsbäume können ADDs als White-Box-Modelle gesehen werden; sie sind also interpretierbar. Wenn man die sukzessive Abstraktion der ADDs fortsetzt, kann man sogar noch bessere Erklärungen für selektierte Aspekte des Modells finden.

Mit der ADD-basierten Aggregation ist es möglich gleich drei *Erklärbarkeitsprobleme* für Random Forests zu lösen [Gu19]:

- das allgemeine *Modell-Erklärungsproblem*,
- das *Klassen-Erklärungsproblem* und
- das *Ergebnis-Erklärungsproblem*.

Die Lösung des allgemeinen *Modell-Erklärungsproblems* wurde bereits in Abschnitt 2 diskutiert. Die Aggregation des Random Forests in einem einzelnen, redundanzfreien ADD ist ebenso einfach zu verstehen wie ein einzelner Entscheidungsbaum. Aus diesem Grund kann der präziseste und kompakteste ADD als Lösung des allgemeinen *Modell-Erklärungsproblems* für Random Forests gesehen werden. Abbildung 4 zeigt diese Modell-Erklärung für das Beispiel des Iris-Datensatzes.

Wenn man die sukzessive Abstraktion der ADDs fortführt, kann man auf ähnliche Weise die Lösung für ein spezielleres Erklärungsproblem finden: das *Klassen-Erklärungsproblem*. Hier wird die Erklärung nur für eine der betrachteten Klassen gefordert, also zum Beispiel für die Klasse *Iris-setosa*. Anstatt in den Blättern des ADD alle Klassen aufzulisten, unterscheidet man hier nur zwischen der Klasse *Iris-setosa* und allen anderen. Das Ergebnis ist ein zweiblättriger ADD bzw. ein BDD. Abbildung 6 zeigt diesen am Beispiel des Iris-Datensatzes. Die Semantik des ursprünglichen Random Forests kann so noch präziser für eine gewählte Klasse beschrieben werden.

Abb. 6: Als Klassen-BDD aggregierter Random Forest für die Klasse *Iris-setosa*. Bzgl. dieser Klasse ist der BDD semantisch äquivalent zu dem Random Forest in Abb. 2. (15 Knoten)

Das *Klassen-Erklärungsproblem* ist auch deshalb interessant, weil es den Klassifikationsprozess umdreht. Anstatt nach der korrekten Klasse für eine bestimmte Eingabe zu suchen, sucht man hier nach allen möglichen Eingaben für eine bestimmte Klasse. Diese Perspektivänderung kann zum Beispiel im Marketing hilfreich sein, weil man zwischen Kunden- und Produkt-Perspektiven wechseln kann [GMS20].

Das dritte Problem ist das *Ergebnis-Erklärungsproblem*. Es ist vor allem für die verantwortliche Nutzung von maschinellem Lernen wichtig und fordert die Erklärung in einem ganz konkreten Anwendungsfall [Gu19]. So können konkrete, automatisch getroffene Entscheidungen begründet werden, was insbesondere in der Medizin von enormer Bedeutung ist. Im Fall des Iris-Beispiels heißt das, dass wir eine Erklärung für die Klassifikation mit ganz konkreten Maßen fordern.

Die Lösung des vorangegangenen *Klassen-Erklärungsproblems* (Abb. 6) erlaubt es uns, auch das *Ergebnis-Erklärungsproblem* in zwei einfachen Schritten zu lösen:

Pfad-basierte Erklärung. Für die konkreten Maße einer Eingabe können wir den entsprechenden Klassen-BDD von seiner Wurzel an traversieren. Wenn wir die (negierten) Prädikate entlang dieses Pfades aufsammeln, erhalten wir eine präzise Erklärung. Deren Konjunktion ist eine hinreichende Bedingung für die getroffene Entscheidung. Beispielsweise erhalten wir für die Werte

$$petallength = 2.6,\ petalwidth = 1.5,\ petallength = 2.65,\ sepallength = 6.9$$

folgende hinreichende Bedingung:

$$petallength \geq 2.45 \wedge\ petalwidth \geq 1.45\ \wedge\ petalwidth < 1.65$$
$$\wedge\ petallength \geq 2.6\ \wedge\ petallength < 2.7 \wedge\ sepallength < 7.05.$$

Konjunktionen vereinfachen. Das Aufsammeln der Prädikate entlang eines Pfades kann erneut Redundanzen für das konkrete Beispiel zum Vorschein bringen. Das ist selbst dann der Fall, wenn der ADD bereits auf die Erfüllbarkeit seiner Pfade gefiltert wurde. Diese können nun genutzt werden, um die Konjunktion weiter zu vereinfachen, indem man nur die stärksten Prädikate aufsammelt. In dem vorangegangenen Beispiel ist $petallength \geq 2.45$ redundant, weil es von dem stärkeren Prädikat $petallength \geq 2.6$ impliziert wird. Auf diese Weise erhält man eine minimale Lösung für das *Ergebnis-Erklärungsproblem*.

5 Schlussfolgerungen

Durch die semantik-erhaltende Aggregation von Random Forest zu einem einzelnen algebraischen Entscheidungsdiagramm, ist es möglich deren Semantik zu kondensieren. Die semantik-erhaltende Transformation lässt sich algebraisch elegant definieren [GS21] und erhält die Kompositionalität zunächst vollständig. Durch eine Reihe nicht-kompositionaler Abstraktionen ist es dann möglich bestimmte Aspekte des Modells hervorzuheben oder seine Größe und Tiefe (bzw. Laufzeit) zu optimieren.

Auf diese Weise werden gleich drei Erklärungsprobleme für diese Methode des maschinellen Lernens gelöst: das *Modell-Erklärungsproblem*, das *Klassen-Erklärungsproblem* und das *Ergebnis-Erklärungsproblem*. Alle drei Lösungen sind von enormer Bedeutung in den verschiedensten Anwendungsgebieten im maschinellen Lernen, beispielsweise in der Medizin [Ch21] oder im Bankensektor.

Gleichzeitig erlaubt die radikale Aggregation es, den Random Forest deutlich schneller auswerten zu können, was in Anwendungen mit hohen Bandbreiten von enormer Bedeutung ist [Fa17]. Anstatt jeden Entscheidungsbaum des Random Forests einzeln auswerten zu müssen, reicht es, einen einzelnen aggregierten ADD zu traversieren. So ist es in mehreren Anwendungsfällen möglich, die gleiche Semantik um mehrere Größenordnungen schneller auswerten zu können.

Literaturverzeichnis

[Ak78] Akers, S. B.: Binary Decision Diagrams. IEEE Trans. Comput., 27(6):509–516, 1978.

[Ba97] Bahar, R.I.; Frohm, E.A.; Gaona, C.M.; Hachtel, G.D.; Macii, E.; Pardo, A.; Somenzi, F.: Algebric Decision Diagrams and Their Applications. Formal Methods in System Design, 10, 1997.

[Br86] Bryant, Randal E.: Graph-Based Algorithms for Boolean Function Manipulation. IEEE Trans. Comput., 35(8):677–691, 1986.

[Br01] Breiman, Leo: Random Forests. Machine Learning, 45(1), 2001.

[Ch21] Chowell, D., Yoo SK. Valero C. et al.: Improved prediction of immune checkpoint blockade efficacy across multiple cancer types. Nature Biotechnology, 2021.

[DG17] Dua, Dheeru; Graff, Casey: , UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, 2017. Accessed: 2020-02-15.

[Fa17] Facebook: , Evaluating boosted decision trees for billions of users. `https://code.fb.com/ml-applications/evaluating-boosted-decision-trees-for-billions-of-users`, 2017. Accessed: 2019-06-11.

[Fi36] Fisher, R. A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, 7(7):179–188, 1936.

[GMS20] Gossen, F.; Margaria, T.; Steffen, B.: Towards Explainability in Machine Learning: The Formal Methods Way. IT Professional, 22(04):8–12, 2020.

[GMS21] Gossen, Frederik; Margaria, Tiziana; Steffen, Bernhard: Formal Methods Boost Experimental Performance for Explainable AI. IT Professional, 23(6):8–12, 2021.

[Go18] Gossen, Frederik; Margaria, Tiziana; Murtovi, Alnis; Naujokat, Stefan; Steffen, Bernhard: DSLs for Decision Services: A Tutorial Introduction to Language-Driven Engineering. In (Margaria, Tiziana; Steffen, Bernhard, Hrsg.): Leveraging Applications of Formal Methods, Verification and Validation. Modeling. Springer International Publishing, S. 546–564, 2018.

[Go19] Gossen, Frederik; Jasper, Marc; Murtovi, Alnis; Steffen, Bernhard: Aggressive Aggregation: a New Paradigm for Program Optimization. CoRR, abs/1912.11281, 2019.

[Go21] Gossen, Frederik: Aggressive Aggregation - (Domain-specific) Program Optimisation with Algebraic Decision Diagrams. LS 05 Programmiersysteme, 2021.

[GS21] Gossen, F.; Steffen, B.: Algebraic Aggregation of Random Forests: Towards Explainability and Rapid Evaluation. International Journal on Software Tools for Technology Transfer, 2021.

[Gu19] Guidotti, Riccardo; Monreale, Anna; Ruggieri, Salvatore; Turini, Franco; Giannotti, Fosca; Pedreschi, Dino: A Survey of Methods for Explaining Black Box Models. ACM Comput. Surv., 51(5):93:1–93:42, 2019.

[St19] Steffen, Bernhard; Gossen, Frederik; Naujokat, Stefan; Margaria, Tiziana: Language-Driven Engineering: From General-Purpose to Purpose-Specific Languages. In (Steffen, Bernhard; Woeginger, Gerhard, Hrsg.): Computing and Software Science: State of the Art and Perspectives. Springer International Publishing, S. 311–344, 2019.

Frederik Gossen wurde am 5. Februar 1992 in Herdecke, Deutschland, geboren. Seit September 2015 promovierte er in der Informatik in Kooperation mit Lero, dem irischen Zentrum für Software Engineering Research, an den Universitäten *University of Limerick*, Irland, und *Technische Universität Dortmund*, Deutschland. In seiner Forschung untersuchte er das Potential von algebraischen Entscheidungsdiagrammen für die Programmoptimierung in verschiedenen Domänen, insbesondere im maschinellen Lernen. Seit April 2020 arbeitet er bei Google an Compilern und Programmoptimierung für maschinelles Lernen, zunächst in München, Deutschland, dann in New York, USA.