Optimierung von Klassifikator-Ensembles mit AutoML

Julius Voggesberger[1]

[1]Institut für Parallele und Verteilte Systeme, Universität Stuttgart, Universitätsstr. 38, Stuttgart, Deutschland

Abstract

Data for classification problems often exhibit complex characteristics that lead to inaccurate predictions of classifiers trained on the data. For example, training a classifier on a small amount of data can lead to overfitting. To solve such problems, multiple classifiers can be combined into an ensemble. For this purpose, multiple classifiers have to be trained that are as accurate and divers as possible. In this case, diversity stands for classifiers that make correct predictions on different data instances. Furhtermore, a suitable method for fusing the individual classifier predictions has to be selected In this paper, we present an approach using AutoML to automatically create and optimize ensembles. The approach is evaluated on two real-world datasets with complex data characteristics. The results of the evaluation show an improvement of the predictive accuracy by the automatically created ensembles.

Zusammenfassung

Daten für Klassifikationsprobleme weisen oft komplexe Charakteristika auf, die zu ungenauen Vorhersagen der mit den Daten trainierten Klassifikatoren führen. Beispielsweise kann eine geringe Menge an Daten zu einer Überanpassung der Klassifikatoren führen. Um derartige Probleme zu lösen, können mehrere Klassifikatoren zu einem Ensemble kombiniert werden. Hierfür müssen mehrere Klassifikatoren trainiert werden, die möglichst genau, aber auch divers sind. Diversität bedeutet in diesem Fall, dass die Klassifikatoren auf unterschiedlichen Dateninstanzen korrekte Vorhersagen treffen. Weiterhin muss eine geeignete Methode für die Fusion der einzelnen Klassifikatorvorhersagen ausgewählt werden. In dieser Arbeit stellen wir einen AutoML-Ansatz vor, mit dem die Erstellung und Optimierung eines Ensembles automatisiert möglich ist. Der Ansatz wird anhand zweier Echtweltdatensätze mit komplexen Datencharakteristika evaluiert. Die Ergebnisse der Evaluation zeigen hierbei eine Verbesserung der Vorhersagegenauigkeit durch die automatisch erstellten Ensembles.

Keywords

Maschinelles Lernen, Klassifikator-Ensembles, Klassifikatordiversität, Entscheidungsfusion

1. Einführung

Um Erkenntnisse aus Daten zu gewinnen sowie Vorhersagen und Entscheidungen zu treffen, werden oft Klassifikationsalgorithmen verwendet. Diese finden in vielen unterschiedlichen Bereichen, wie z. B. dem Medizinwesen, dem Internet der Dinge oder in der Bilderkennung Anwendung [1]. Jedoch kann ein Klassifikationsalgorithmus nicht in jedem Fall einen Klassifikator mit einer hohen Vorhersagegenauigkeit erstellen. So kann ein Klassifikator durch das Training auf kleinen Datensätzen überangepasst werden oder komplexe Datencharakteristika nicht vollständig erfassen [2, 3, 4, 5, 6].

Um diese Probleme zu lösen, können mehrere Klassifikatoren in einem Ensemble kombiniert werden, sodass eine höhere Vorhersagegenauigkeit und Generalisierbarkeit erreicht werden kann [3]. Dabei treffen die Klassifikatoren diverse Vorhersagen, welche in einem Fusionsschritt zu einer gemeinsamen Vorhersage kombiniert werden. Diversität bedeutet hierbei, dass die Klassifikatoren fehlerhafte und korrekte Vorhersagen auf unterschiedlichen Dateninstanzen treffen. Bei der anschließenden Entscheidungsfusion können so primär die korrekten Vorhersagen übernommen werden. Damit ein Ensemble mit hoher Vorhersagegenauigkeit erzeugt werden kann, müssen drei Anforderungen erfüllt werden.

A1: Genaue Klassifikatoren: Eine Steigerung der Vorhersagegenauigkeit durch die Entscheidungsfusion setzt voraus, dass die einzelnen Klassifikatoren bereits eine hohe Vorhersagegenauigkeit erreichen [2, 7]. Folglich ist es nötig, dass geeignete Klassifikationsalgorithmen ausgewählt werden, die anhand ihrer Hyper-Parameter optimiert sind.

A2: Diverse Klassifikatoren: Die zweite Anforderung ist die Diversität der Klassifikatoren hinsichtlich ihrer korrekten und fehlerhaften Vorhersagen für unterschiedliche Dateninstanzen [7, 2, 8]. Eine hohe Diversität ermöglicht die Fusion der korrekten Vorhersagen, wodurch eine Steigerung der Vorhersagegenauigkeit und Generalisierbarkeit möglich ist. Diversität kann hierbei implizit oder explizit erzeugt werden [8]. Implizite Methoden manipulieren den Trainingsprozess der Klassifikatoren durch die Verwendung von Zufall, wohingegen explizite Methoden die Diversität direkt durch z. B. Diversitätsmetriken optimieren.

A3: Genaue Entscheidungsfusion: Damit während der Fusion die korrekten Vorhersagen der Klassifikatoren bevorzugt werden, muss ein geeigneter Entscheidungsfusionsalgorithmus ausgewählt werden [3]. Jedoch existie-

34th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), June 7-9, 2023, Hirsau, Germany

julius.voggesberger@ipvs.uni-stuttgart.de (J. Voggesberger)

0000-0003-4808-1922 (J. Voggesberger)

Tabelle 1
Vergleich von Ansätzen aus der Literatur. ✓ = vollständig erfüllt, (✓) = teilweise erfüllt, ✗ = nicht erfüllt.

	Methoden	A1: Genaue Klassifikatoren	A2: Diverse Klassifikatoren	A3: Genaue Fusion
Ensemble Algorithmen	Bagging [11], Boosting [12]	(✓)	(✓)	✗
	Ungleichverteilte Daten [13], MEG [14]	(✓)	(✓)	(✓)
	PUSION [9]	✗	✗	(✓)
Automated Machine Learning	H2O [15]	✓	(✓)	(✓)
	Auto-Sklearn [16], ESMBO [17], BOE [18], AutoGluon-Tabular [19]	✓	(✓)	✗

ren viele unterschiedliche Fusionsalgorithmen, die verschiedene Strategien zur Fusion der Vorhersagen verwenden und bei denen zusätzlich Hyper-Parameter optimiert werden müssen [9, 10].

In dieser Arbeit wird ein Überblick über unterschiedlichen Lösungsansätze für die Erstellung eines Ensembles, das diese Anforderungen erfüllt, gegeben. Die Ansätze beziehen sich neben den Klassifikations- und Entscheidungsfusionsalgorithmen zu einem großen Teil auch auf die Betrachtung der Daten und ihrer Charakteristiken sowie auf Aspekte der Datenvorbereitung, die ebenso einen wesentlichen Einfluss auf die Diversität und Genauigkeit eines Ensembles haben können. Zusätzlich werden erste Ergebnisse anhand eines implementierten Ansatzes vorgestellt. Für die Automatisierung der Auswahl von Klassifikations- und Entscheidungsfusionsalgorithmen und der Optimierung ihrer Hyper-Parameter wird Automated Machine Learning (AutoML) verwendet. Abschließend werden die Ergebnisse der Evaluation des Ansatzes anhand zweier Datensätze diskutiert.

Die restliche Arbeit ist wie folgt gegliedert: Anhand verwandter Arbeiten wird in Abschnitt 2 die Forschungslücke aufgezeigt und anschließend in Abschnitt 3 Ansätze für deren Lösung präsentiert. Ein erstes auf diesen Ansätzen basierendes Konzept wird in Abschnitt 4 vorgestellt, sowie anhand zweier Datensätze evaluiert. Abschließend werden in Abschnitt 5 die Arbeit zusammengefasst sowie mögliche Ansatzpunkte für zukünftige Arbeiten genannt.

2. Verwandte Arbeiten

In diesem Abschnitt werden bestehende Ansätze betrachtet, die die Erstellung von Ensembles behandeln. Hierbei wird zusätzlich der Fokus auf AutoML Ansätze gelegt, da dieser Ansatz im weiteren Verlauf dieser Arbeit verfolgt wird. Eine Übersicht über die vorgestellte Literatur anhand der Anforderungen A1-A3 ist in Tabelle 1 gegeben.

2.1. Ensemble-Algorithmen

Es existiert bereits eine Vielfalt an Algorithmen für die Erstellung von Ensembles. Die bekanntesten Verfahren sind Boosting [12] und Bagging [11]. Beispiele für Algorithmen dieser Verfahren sind AdaBoost [20] und Random Forests [21]. AdaBoost erzeugt Diversität explizit, indem in einem iterativen Verfahren Klassifikatoren mit Fokus auf die Fehler der vorhergehenden Klassifikatoren trainiert werden. Eine implizite Erzeugung der Diversität erfolgt jedoch nicht. Ebenso werden die Hyperparameter der Klassifikatoren und die Entscheidungsfusion nicht optimiert. Random Forests teilen diese Defizite, jedoch erzeugt der Algorithmus die Diversität implizit statt explizit. Hierfür werden die Klassifikatoren auf unterschiedlichen, zufällig ausgewählten Teilmengen der Trainingsdaten trainiert.

Neuere Veröffentlichung fokussieren sich hingegen auf die Verwendung von multikriterieller Optimierung. So wird z. B. in "Multi-objective Ensemble Generation" (MEG) [14] ein Ensemble sowohl anhand der Diversität als auch der Vorhersagegenauigkeit optimiert. Die betrachtete Menge an Klassifikatoren und Entscheidungsfusionsalgorithmen ist hierbei jedoch gering und Diversität wird nur explizit erzeugt. Ein weiterer Algorithmus betrachtet Ensembles für ungleich verteilte Daten [13]. Hierfür werden die Gewichte eines Mehrheitsvotums Ansatzes sowohl anhand von Precision als auch Recall optimiert. Jedoch werden die Klassifikatoren selbst nicht optimiert, sondern nur aus einer Menge an 15 Algorithmen eine Untermenge für das Ensemble ausgewählt.

Einen anderen Ansatz bietet das PUSION Framework [9]. Dieses nimmt die diverse Klassifikatormenge als gegeben an und optimiert ausschließlich den zu verwendeten Entscheidungsfusionsalgorithmus. Jedoch wird hierbei nicht die Hyper-Parameteroptimierung der Entscheidungsfusionsalgorithmen mitbetrachtet.

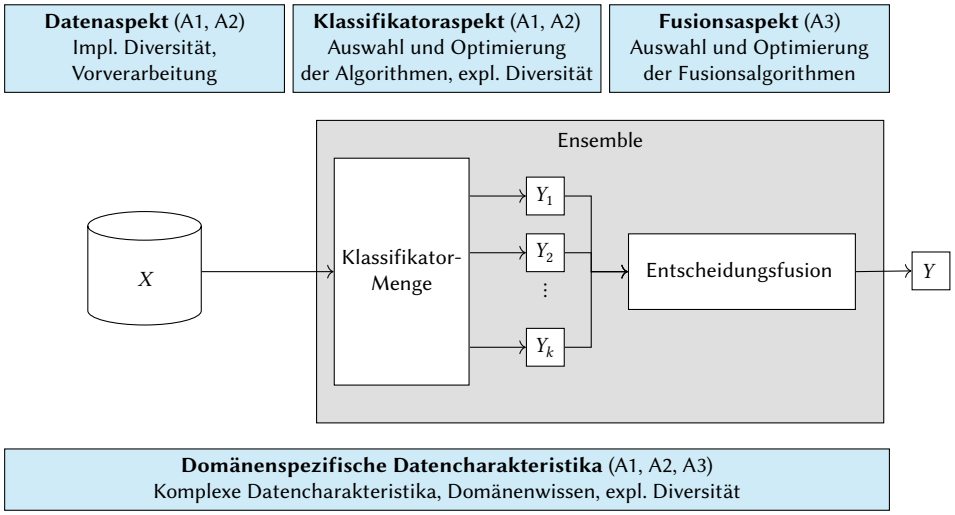

Abbildung 1: Schematische Darstellung eines Ensembles. Die Eingabe X ist ein Datensatz, Y_i sind die Vorhersagen der einzelnen Klassifikatoren und Y ist die fusionierte Ausgabe des Ensembles. Blau hinterlegt sind die möglichen Aspekte zur Lösung der Anforderungen A1-A3.

2.2. AutoML für Ensembles

Automated Machine Learning (AutoML) bezeichnet die automatisierte Erstellung eines maschinellen Lernmodells für gegebene Daten innerhalb eines bestimmten Budgets [16]. Im AutoML-Bereich existieren bereits viele verschiedene Frameworks, jedoch unterstützen nur einige wenige die Erstellung und Optimierung von Ensembles. So erlauben dies z. B. Auto-sklearn [16], H2O [15], ESMBO [17], Bayesian Optimization for Ensembles (BOE) [18] und AutoGluon-Tabular [19]. Jedoch nehmen diese an, dass die Verwendung von unterschiedlichen Algorithmen und Hyper-Parametrisierungen eine ausreichend diverse Menge an Klassifikatoren erzeugt. Zusätzlich dazu wird bei der Mehrheit der Frameworks der Entscheidungsfusionsalgorithmus nicht optimiert, sondern ein spezifischer Algorithmus vorgegeben. Eine Ausnahme bildet H2O, welches eine begrenzte Optimierung der Fusion erlaubt, indem verschiedene Algorithmen für die Stacked Generalization verwendet werden können.

3. Lösungsansätze

In diesem Abschnitt werden vier verschiedene Aspekte vorgestellt, mit denen die Anforderungen A1-A3 zur Erstellung eines Klassifikator-Ensembles gelöst werden können (siehe Abbildung 1). Im Folgenden werden diese Aspekte definiert, sowie deren Ansatzpunkte zur Lösung der Anforderungen präsentiert.

Datenaspekt: Der Datenaspekt schließt die Verwendung von Vorverarbeitungs- und Feature-Engineering-Methoden und den Einsatz impliziter Diversitätsmethoden in der Datenvorbereitung ein. Über die Manipulation der Daten kann die Genauigkeit und Diversität der trainierten Klassifikator-Menge beeinflusst werden (Anforderungen A1 und A2). Vorverarbeitungs- und Feature-Engineering-Methoden beeinflussen insbesondere die Vorhersagegenauigkeit der einzelnen Klassifikatoren (A1). Diese Methoden bereiten die Daten auf, sodass diese von Klassifikationsalgorithmen besser verarbeitet werden können. Hierfür können z. B. kategorische Daten numerisch kodiert oder die Dimensionalität der Daten reduziert werden. Um mit Hilfe der Vorverarbeitungs- und Feature-Engineering-Methoden zusätzlich die Diversität zu erhöhen (A2), können je Klassifikator unterschiedliche Methoden auf die Daten angewendet werden. Damit wird jeder Klassifikator auf unterschiedlich vorbereiteten Daten trainiert und angepasst, sodass die gesamte Menge an Klassifikatoren insgesamt diverser wird.

Alternativ werden innerhalb gängiger Ensemble-Algorithmen, wie z. B. Random Forests, implizite Diversitätsmethoden verwendet, um die Diversität zwischen den einzelnen Klassifikatoren zu erhöhen (vgl. Abschnitt 2.1). Hierbei werden die Trainingsdaten für jeden Klassifikator zufällig ausgewählt, z. B. über die Auswahl von Dateninstanzen mit Bootstrapping [11] oder einer Attribut-Teilmenge mit der Random-Subspace-Methode [22].

Klassifikatoraspekt: Unter diesem Aspekt wird die Auswahl und Optimierung der Klassifikatoren des Ensembles anhand der Vorhersagegenauigkeit und Diversität betrachtet (A1+A2). Diversität kann hierbei explizit über die Verwendung von Diversitätsmetriken optimiert werden [23]. Mit Hilfe dieser Metriken, wie z. B. der Dop-

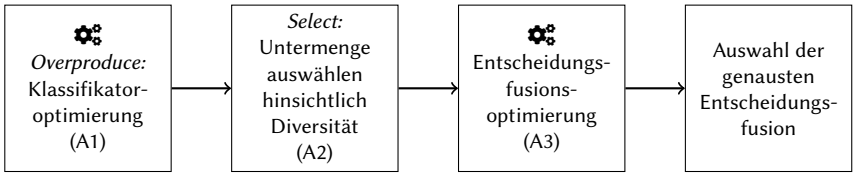

Abbildung 2: Darstellung des gewählten Ansatzes für das AutoML-Frameworks für Ensembles. = Optimierungsproblem.

pelfehlerrate oder Q-Statistik, kann die Diversität eines Ensembles gemessen und so eine Aussage über sie getroffen werden. Jedoch ist aktuell der Zusammenhang zwischen der mit diesen Metriken gemessenen Diversität und der Genauigkeit des Ensembles aus Klassifikatoren nicht ausreichend untersucht. Für die Optimierung kann z. B. AutoML und Meta-Learning verwendet werden. Meta-Learning bezeichnet das Lernen aus vorherigen Erfahrungen, um schneller ein Modell mit hoher Vorhersagegenauigkeit für einen neuen Datensatz zu finden [24]. Als Erfahrungen werden hierbei Metadaten bezeichnet, die vorherige Datensätze und darauf evaluierte Machine-Learning-Algorithmen beschreiben [25, 26].

Entscheidungsfusionsaspekt: Der Fusionsaspekt beschreibt die Auswahl und Optimierung des Fusionsalgorithmus. Die Optimierung erfolgt im Gegensatz zu den Klassifikatoren lediglich anhand der Vorhersagegenauigkeit (A3). Jedoch kann hier ebenfalls AutoML und Meta-Learning für die Optimierung verwendet werden.

Domänenspezifische Datencharakteristika: Die Ausnutzung domänenspezifischer Datencharakteristika ermöglicht es, Wissen über die Daten und die Datendomäne zu verwenden. Dies schließt komplexe, domänenspezifische Datencharakteristika wie z. B. ungleiche Klassenverteilungen, verschiedene Arten von Bias in den Daten sowie zeitliche Änderungen der statistischen Datenverteilungen mit ein [27, 28, 29, 30, 31]. Beispielsweise stellen Hirsch et al. [4] einen Ansatz vor, der domänenspezifische Datencharakteristika sowie Domänenwissen explizit innerhalb einer Vorbereitung der Daten ausnutzt, um die Eingabedaten X in mehrere Teilmengen aufzuteilen. Durch diese Datenpartitionierung können auf den Teilmengen spezialisierte Klassifikatoren trainiert werden, die für die jeweilige Datenteilmenge genauere Vorhersagen trifft als ein auf der gesamten Datenmenge X trainierter Klassifikator (A1). Da jeder dieser Klassifikatoren auf einer anderen Teilmenge der Daten trainiert wurde, wird die gesamte Menge an Klassifikatoren divers und kann für ein Ensemble genutzt werden (A2). Analog kann Wissen über die Problemdomäne verwendet werden, um geeignete Klassifikatoren und Fusionsalgorithmen auszuwählen sowie die Größe des Ensembles zu bestimmen (A3). Insbesondere kann dieses Wissen für die Erstellung einer diversen Klassifikatormenge verwendet werden, um so explizit anhand der Datencharakteristika Diversität zu erzeugen.

4. Erste Ergebnisse

Die Diskussion bestehender Arbeiten in Abschnitt 2 zeigt, dass aktuell kein Framework die Anforderungen A1-A3 erfüllt. Um dieses Problem zu lösen, haben wir [32] einen Ansatz entwickelt, der auf die in Abschnitt 3 vorgestellten Lösungsansätze eingeht. Im Folgenden werden der Ansatz und seine prototypische Implementierung vorgestellt sowie erste Evaluationsergebnisse präsentiert.

4.1. AutoML-Ansatz für Ensembles

Der hier vorgestellte Ansatz setzt die in Abschnitt 3 vorgestellten Klassifikations- und Fusionsaspekte mithilfe von AutoML um (siehe Abb. 2). Für die Erstellung der Klassifikator-Menge wird ein sogenannter „Overproduce and Select“-Ansatz verfolgt: In einem *Overproduce*-Schritt werden mehrere Klassifikatorkonfigurationen anhand ihrer Vorhersagegenauigkeit durch AutoML optimiert und evaluiert. Jede Konfiguration und ihre Evaluation werden abgespeichert, sodass nach der Optimierung in einem *Select*-Schritt eine Untermenge aus allen evaluierten Konfigurationen ausgewählt werden kann. Dies erlaubt es, eine Menge an Klassifikatoren anhand ihrer Diversität und Genauigkeit auszuwählen und somit die Anforderungen A1 und A2 zu erfüllen.

Für die Optimierung der Entscheidungsfusion wird hingegen ausschließlich AutoML verwendet. Die Optimierung und Auswahl des besten Entscheidungsfusionsalgorithmus erfolgt somit anhand der Vorhersagegenauigkeit, um die Anforderung A3 zu erfüllen.

Durch dieses Vorgehen wird ein Ensemble erstellt, dass die Anforderungen A1-A3 erfüllt. Weiterhin ist die Komplexität des Suchraums und der Optimierung niedrig, da die Klassifikator-Menge und die Entscheidungsfusion getrennt voneinander optimiert werden [32]. Dies ermöglicht eine kürzere Laufzeit und erhöht die Wahrscheinlichkeit eine genaue und generalisierende Ensemble-Konfiguration zu finden.

Tabelle 2
Vorhersagegenauigkeit der Baseline, statischen Fusion und des optimierten Ensembles. Für die Baseline bezeichnet der Entscheidungsfusionsalgorithmus den Klassifikationsalgorithmus. SVM=Support Vector Machine, AVG=„Simple Averaging“, WV=Gewichtetes Mehrheitsvotum, RF=Random Forest, NN=Neurales Netzwerk.

Datensatz	Typ	Entscheidungs-fusion	Ensemblegröße	Accuracy (%)	Ausgewogene Accuracy (%)	Makro F1 (%)
Kropt	Baseline	SVM	-	86,56	82,86	83,68
	Statische Fusion	AVG	5	86,07	79,00	81,66
	Optimiertes Ensemble	WV	5	**88,56**	**84,90**	**85,81**
Ldpa	Baseline	RF	-	86,59	64,05	69,29
	Statische Fusion	AVG	10	81,72	54,17	57,73
	Optimiertes Ensemble	NN	10	**87,52**	**72,93**	**77,68**

4.2. Implementierung

Die Implementierung des Prototyps erfolgte in der Programmiersprache Python. Für die Bereitstellung von Klassifikationsalgorithmen wird die Bibliothek scikit-learn 1.0.2 [33] verwendet und Entscheidungsfusionsalgorithmen sind aus dem PUSION-Framework 1.3.5 [9] adaptiert. Als AutoML-Optimierung wird eine Zufallssuche [34] verwendet, während für die Auswahl der Klassifikator-Menge ein Clustering-Ansatz zum Einsatz kommt [35, 25]. Durch den Clustering-Ansatz werden Klassifikatoren mit einer niedrigen Diversität zueinander in ein Cluster gruppiert, und pro Cluster der Klassifikator mit der höchsten Vorhersagegenauigkeit ausgewählt. Weitere Software-Bibliotheken, die für die Implementierung verwendet werden, sind NumPy 1.22.2, Pandas 1.4.0 und Scipy 1.8.0. Der Prototyp wurde auf einem Cluster mit 16 virtuellen CPU-Kernen und 32 GB RAM ausgeführt.

4.3. Experimentelle Evaluation

Der vorgestellte Ansatz wird anhand zweier Echtweltdatensätze evaluiert, die ungleiche Klassenverteilungen aufweisen. Hierfür wurde der Schach-Datensatz Kropt[1] mit 18 und der Bewegungs-Datensatz Ldpa[2] mit 11 Klassen ausgewählt. Als Baseline der Evaluation wird der einzelne Klassifikator gewählt, der während der AutoML-Optimierung die höchste Vorhersagegenauigkeit erzielt. Um die Optimierung der Entscheidungsfusion separat zu beurteilen, wird zusätzlich das vom Prototyp optimierte Ensemble mit einem Ensemble ohne Fusionsoptimierung verglichen. Für dieses wird als statisch gewählte Entscheidungsfusion das „Simple Averaging“ (AVG) [3, 23] auf die Klassifikator-Menge des optimierten Ensembles angewendet.

Die Ergebnisse der Evaluation sind in Tabelle 2 aufgelistet. Der Vergleich erfolgt anhand der Metriken Accuracy, Ausgewogener Accuracy und Makro F1, wobei letzteren beiden besser für Mehrklassenprobleme geeignet sind. Um eine geeignete Größe für die Klassifikator-Menge zu finden, werden alle Größen zwischen 2 und 30 getestet. Abschließend wird das Ensemble mit der höchsten Vorhersagegenauigkeit gewählt. Für die Optimierung der Klassifikatoren wird ein Limit von einer Stunde und für die Fusionsoptimierung von zehn Minuten festgelegt.

Vergleich von Baseline und optimiertem Ensemble: Im Vergleich zwischen der Baseline und unserem optimierten Ensemble erreicht Letzteres für beide Datensätze die höhere Vorhersagegenauigkeit. Die Verbesserung des Ensembles beträgt für den Kropt-Datensatz $+2,00$%-Punkte Accuracy, $+2,05$%-Punkte ausgewogene Accuracy und $+2,13$%-Punkte Makro F1 und für den Ldpa-Datensatz $+0,93$%-Punkte Accuracy, $+8,88$%-Punkte ausgewogene Accuracy und $+8,39$%-Punkte Makro F1. Bei letzterem Datensatz fällt die Differenz von Makro F1 und gewichteter Accuracy Metrik mit mehr als +8%-Punkte bedeutend größer aus als für die Accuracy. Dies ist darauf zurückzuführen, dass beide Metriken die Klassen in ihren Berechnungen gleich gewichten, unabhängig von deren tatsächlichen Häufigkeit in den Daten.

Vergleich von statischer Fusion und optimiertem Ensemble: Bei der Betrachtung der Fusionsoptimierung zeigt sich sogar eine Verschlechterung zur Baseline, falls die Entscheidungsfusion nicht optimiert wird, sondern eine statische Fusion verwendet wird. So verringert sich die Vorhersagegenauigkeit des Ensembles ohne Optimierung um $-4,87$%-Punkte Accuracy, $-3,86$%-Punkte ausgewogene Accuracy und $-2,02$%-Punkte Makro F1 für den Kropt-Datensatz und um $-0,49$%-Punkte Accuracy, $-9,88$%-Punkte ausgewogene Accuracy und $-11,56$%-Punkte Makro F1 für den Ldpa-Datensatz. Im Vergleich zu den optimierten Ensembles fällt die Verbesserung durch die Fusionsoptimierung im Vergleich zur statischen Fusion folglich nochmals höher aus. So beträgt die Verbesserung für den Kropt-Datensatz $+6,87$%-Punkte Accuracy, $+5,90$%-Punkte ausgewogene Accuracy und $+4,15$%-Punkte Makro F1 sowie $+1,42$%-Punkte

[1] https://www.openml.org/d/184
[2] https://www.openml.org/d/1483

Accuracy, +18, 76%-Punkte ausgewogene Accuracy und +19, 95%-Punkte Makro F1 für den Ldpa-Datensatz.

Diskussion: Durch die Evaluation konnte gezeigt werden, dass die durch unseren Ansatz optimierten Ensembles für beide Datensätze die Vorhersagen mit der höchsten Genauigkeit erzielen. Insbesondere mit Bezug auf ungleichverteilte Mehrklassenprobleme zeigt sich eine große Verbesserung der Vorhersagegenauigkeit, wie anhand der Makro F1 und ausgewogenen Accuracy-Metriken gesehen werden kann. Zusätzlich zeigt sich diese Verbesserung vor allem in Bezug auf die Optimierung der Entscheidungsfusionsmethode. Wird diese nicht optimiert, so kann sich die Vorhersage des Ensembles im Vergleich zu einzelnen Klassifikatoren verschlechtern. Diese Verbesserung lässt sich dadurch erklären, dass eine statisch gewählte Entscheidungsfusionsmethode nicht auf die gewählte Menge an Klassifikatoren sowie die verwendeten Daten angepasst ist. Aus diesem Grund führt die hier verwendete Optimierung der Entscheidungsfusion zu einer höheren Vorhersagegenauigkeit.

5. Zusammenfassung und nächste Schritte

In dieser Arbeit haben wir einen AutoML-Ansatz vorgestellt, der die automatische Optimierung eines Ensembles ermöglicht. Hierbei wird die Klassifikator-Menge hinsichtlich Diversität und Genauigkeit sowie die Entscheidungsfusion hinsichtlich ihrer Genauigkeit optimiert. Somit werden aktuell der Klassifikator- und Entscheidungsfusionsaspekt behandelt. Der Ansatz wurde an zwei Datensätzen mit unausgewogenen Klassenverteilungen evaluiert. Anhand der Evaluation ist zu sehen, dass die Erstellung eines optimierten Ensembles und die Fusionsoptimierung eine Verbesserung der Vorhersagegenauigkeit erzielen.

In zukünftigen Arbeiten planen wir, den vorgestellten Prototypen hinsichtlich des Datenaspektes zu erweitern. So können Methoden für die implizite Diversität und Vorverarbeitung integriert werden. Zusätzlich ist eine ausführlichere Evaluation anhand weiterer Datensätze wie z. B. MNIST[3] und Covertype[4] geplant. Hierbei sollen weitere Aspekte wie der Einfluss der Diversität auf die Optimierung des Ensembles mit betrachtet werden.

Weiterhin kann Meta-Learning für die Auswahl von Klassifikatoren und Entscheidungsfusionsmethoden betrachtet werden. Zuletzt bietet die Betrachtung domänenspezifischer Datencharakteristika neue Ansätze, um z. B. Klassifikatoren und Entscheidungsfusionsmethoden auszuwählen und zu optimieren.

[3] https://www.openml.org/d/554
[4] https://www.openml.org/d/180

Acknowledgments

Der Autor bedankt sich bei Bernhard Mitschang und Peter Reimann für ihr Feedback zur Verbesserung des Papers.

Literatur

[1] I. H. Sarker, Machine learning: Algorithms, real-world applications and research directions, SN computer science 2 (2021) 160.

[2] T. G. Dietterich, Ensemble Methods in Machine Learning, in: G. Goos, J. Hartmanis, J. van Leeuwen (Eds.), Multiple Classifier Systems, volume 1857, 2000, pp. 1–15. doi:10.1007/3-540-45014-9_1.

[3] R. Polikar, Ensemble Based Systems in Decision Making, IEEE Circuits and Systems Magazine 6 (2006) 21–45. doi:10.1109/MCAS.2006.1688199.

[4] V. Hirsch, P. Reimann, D. Treder-Tschechlov, H. Schwarz, B. Mitschang, Exploiting Domain Knowledge to Address Class Imbalance and a Heterogeneous Feature Space in Multi-Class Classification, The VLDB Journal (2023). doi:10.1007/s00778-023-00780-6.

[5] V. Hirsch, P. Reimann, B. Mitschang, Data-Driven Fault Diagnosis in End-of-Line Testing of Complex Products, in: Proc. of the 6th IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2019, pp. 492–503. doi:10.1109/DSAA.2019.00064.

[6] Y. Wilhelm, U. Schreier, P. Reimann, B. Mitschang, H. Ziekow, Data Science Approaches to Quality Control in Manufacturing: A Review of Problems, Challenges and Architecture, in: Proc. of the 14th Symposium on Service-Oriented Computing (SummerSOC), Communications in Computer and Information Science (CCIS), 2020, pp. 45–65. doi:10.1007/978-3-030-64846-6_4.

[7] L. Hansen, P. Salamon, Neural Network Ensembles, IEEE Trans. on Pattern Analysis and Machine Intelligence 12 (Oct./1990) 993–1001. doi:10.1109/34.58871.

[8] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity Creation Methods: A Survey and Categorisation, Information Fusion 6 (2005) 5–20. doi:10.1016/j.inffus.2004.04.004.

[9] Y. Wilhelm, P. Reimann, W. Gauchel, S. Klein, B. Mitschang, PUSION- A Generic and Automated Framework for Decision Fusion, in: Proc. of the 39th International Conference on Data Engineering (ICDE), 2023.

[10] Y. Wilhelm, P. Reimann, W. Gauchel, B. Mitschang, Overview on Hybrid Approaches to Fault Detection and Diagnosis: Combining Data-driven, Physics-

based and Knowledge-based Models, Procedia CIRP 99 (2021) 278–283. doi:10.1016/j.procir.2021.03.041.

[11] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140. doi:10.1007/BF00058655.

[12] R. E. Schapire, The Strength of Weak Learnability, Machine Learning 5 (1990) 197–227. doi:10.1007/BF00116037.

[13] W. Wegier, M. Koziarski, M. Wozniak, Multicriteria Classifier Ensemble Learning for Imbalanced Data, IEEE Access 10 (2022) 16807–16818. doi:10.1109/ACCESS.2022.3149914.

[14] R. Moussa, G. Guizzo, F. Sarro, MEG: Multi-objective Ensemble Generation for Software Defect Prediction, in: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2022, pp. 159–170. doi:10.1145/3544902.3546255.

[15] E. LeDell, S. Poirier, H2O AutoML: Scalable Automatic Machine Learning, in: Proc. of the AutoML Workshop at ICML, 2020.

[16] M. Feurer, et al., Efficient and Robust Automated Machine Learning, in: C. Cortes, et al. (Eds.), Advances in Neural Information Processing Systems, volume 28, 2015.

[17] A. Lacoste, H. Larochelle, M. Marchand, F. Laviolette, Sequential Model-Based Ensemble Optimization, in: Proc. of the 13th Conference on Uncertainty in Artificial Intelligence, UAI'14, 2014, p. 440–448.

[18] J.-C. Lévesque, C. Gagné, R. Sabourin, Bayesian Hyperparameter Optimization for Ensemble Learning, in: A. Ihler, D. Janzing (Eds.), Proc. of the 32nd Conference on Uncertainty in Artificial Intelligence, 2016.

[19] N. Erickson, et al., AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data, 2020. doi:10.48550/arXiv.2003.06505. arXiv:2003.06505.

[20] Y. Freund, R. E. Schapire, A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, Journal of Computer and System Sciences 55 (1997) 119–139. doi:10.1006/jcss.1997.1504.

[21] L. Breiman, Random Forests, Machine Learning 45 (2001) 5–32. doi:10.1023/A:1010933404324.

[22] Tin Kam Ho, The Random Subspace Method for Constructing Decision Forests, IEEE Trans. on Pattern Analysis and Machine Intelligence 20 (Aug./1998) 832–844. doi:10.1109/34.709601.

[23] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, J. Wiley, 2004.

[24] J. Vanschoren, Meta-Learning, Springer International Publishing, 2019, pp. 35–61. doi:10.1007/978-3-030-05318-5_2.

[25] A. G. Villanueva Zacarias, C. Weber, P. Reimann, B. Mitschang, AssistML: A Concept to Recommend ML Solutions for Predictive Use Cases, in: Proc. of the 8th International Conference on Data Science and Advanced Analytics (DSAA), 2021, pp. 148–155. doi:10.1109/DSAA53316.2021.9564168.

[26] C. Weber, P. Hirmer, P. Reimann, A Model Management Platform for Industry 4.0 - Enabling Management of Machine Learning Models in Manufacturing Environments, in: Proc. of the 23th International Conference on Business Information Systems (BIS), 2020, pp. 403–417. doi:https://doi.org/10.1007/978-3-030-53337-3_30.

[27] Treder-Tschechlov, Dennis, P. Reimann, H. Schwarz, B. Mitschang, Approach to Synthetic Data Generation for Imbalanced Multi-class Problems with Heterogeneous Groups, in: BTW, 2023, pp. 329–351. doi:10.18420/BTW2023-16.

[28] H. Suresh, J. Guttag, A framework for understanding sources of harm throughout the machine learning life cycle, EAAMO '21, 2021. doi:10.1145/3465416.3483305.

[29] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, ACM Comput. Surv. 54 (2021). doi:10.1145/3457607.

[30] Moreno-Torres, et al., A unifying view on dataset shift in classification, Pattern recognition 45 (2012) 521–530.

[31] M. Spieß, P. Reimann, C. Weber, B. Mitschang, Analysis of Incremental Learning and Windowing to handle Combined Dataset Shifts on Binary Classification for Product Failure Prediction, in: Proc. of the 24th International Conference on on Enterprise Information Systems (ICEIS), SCITEPRESS, 2022, pp. 394–405. doi:https://doi.org/10.5220/0011093300003179.

[32] J. Voggesberger, P. Reimann, B. Mitschang, Towards the Automatic Creation of Optimized Classifier Ensembles, in: Proc. of the 25th Int. Conference on Enterprise Information Systems (ICEIS) - Volume 1, 2023, pp. 614–621.

[33] F. Pedregosa, et al., Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[34] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (2012) 281–305.

[35] G. Giacinto, F. Roli, An Approach to the Automatic Design of Multiple Classifier Systems, Pattern Recognition Letters 22 (2001) 25–33. doi:10.1016/S0167-8655(00)00096-9.