Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Analyse von Datenvorverarbeitungsmethoden zur Verbesserung der Diversität in Klassifikationsensembles

Sarah Dosdall, B.Sc.

Studiengang: Informatik

Prüfer/in: Prof. Dr.-Ing. habil. Bernhard Mitschang

Betreuer/in: Julius Voggesberger, M.Sc.

Beginn am: 15. Mai 2024

Beendet am: 22. November 2024

Kurzfassung

Maschinelles Lernen wird mittlerweile in verschiedenen Bereichen verwendet. Eine wesentliche Einschränkung von maschinellen Lernalgorithmen liegt jedoch darin, dass diese häufig Annahmen über die Daten treffen, die in realen Szenarien nicht erfüllt sind. So wird oft vorausgesetzt, dass jede Klasse im Datensatz gleichmäßig repräsentiert ist oder alle enthaltenen Informationen relevant sind. Beim Fehlen dieser Vorraussetzungen in einem Datensatz hat sich die Verwendung von Ensembles bewährt, da diese eine bessere Vorhersagegenauigkeit als einzelne Klassifikatoren erreichen.

Im Rahmen dieser Arbeit wird ein Konzept entwickelt und evaluiert, das durch die gezielte Anwendung von Datenvorverarbeitungsmethoden, entsprechend den Datencharakteristiken eines Datensatzes, die Diversität in Klassifikationsensembles erhöht. Ziel ist es, durch die gezielte Vorverarbeitung die Diversität im Ensemble zu erhöhen und damit die Klassifikationsleistung des Ensembles zu verbessern. Das erarbeitete Konzept wird prototypisch in einem AutoML-Ensemble Framework implementiert, welches die automatische Erstellung und gleichzeitige Optimierung von Vorverarbeitungsmethoden sowie Ensembles unterstützt. Anschließend erfolgt die Evaluation des Konzepts anhand von 23 Datensätzen, die sowohl Echtwelt- als auch synthetische Datensätze umfassen. Zur Bewertung des Konzepts werden Vergleichsläufe durchgeführt. Hierbei weist das erarbeitete Konzept im Schnitt die beste Performanz sowie Diversität auf. Die Ergebnisse der Evaluation zeigen, dass der erarbeitet Ansatz vielversprechend ist, um Diversität in Klassifikationsensembles zu erzeugen. Allerding werden mehrere Faktoren identifiziert, die die Klassifikationsleistung der Ensembles beeinträchtigen. Daher ist weitere Forschung erforderlich, um die Schwächen des entwickelten Konzepts gezielt zu adressieren und dessen Effektivität zu steigern.

Abstract

Machine learning is now used in various areas. However, a major limitation of machine learning algorithms is that they often make assumptions about the data that are not fulfilled in real-life scenarios. For example, it is often assumed that each class in the data set is equally represented or that all the information it contains is relevant. In case that these assumptions are not fulfilled in a data set, the use of ensembles has proven its worth, as these achieve better prediction accuracy than individual classifiers.

In this work, a concept is developed and evaluated that aims to enhance diversity in classification ensembles by the targeted application of data preprocessing methods tailored to the specific characteristics of a dataset. The goal is to increase ensemble diversity through tailored preprocessing, thereby improving the classification performance of the ensemble. The developed concept is prototypically implemented in an AutoML ensemble framework, which supports the automatic creation and simultaneous optimisation of preprocessing methods and ensembles. The concept is then evaluated using 23 data sets, including both real-world and synthetic data sets. Comparison runs were carried out to evaluate the concept. On average, the developed concept shows the best performance and diversity. The results of the evaluation illustrate that the developed approach is promising for generating diversity in classification ensembles. However, several factors are identified that affect the classification performance of the ensembles. Further research is therefore required to specifically address the weaknesses of the developed concept and to increase its effectiveness.

Inhaltsverzeichnis

1 Einleitung **19**

2 Grundlagen **21**
2.1 Klassifikation 21
2.2 Datencharakteristiken 22
2.3 Datenvorverarbeitungsmethoden 25
2.4 Ensemble-Lernen 33
2.5 AutoML 38

3 Verwandte Arbeiten **41**
3.1 Implizite Diversität in Klassifikationsensembles 41
3.2 Integrierte Vorverarbeitungsmethoden in AutoML-Frameworks 43
3.3 Einfluss von Datenvorverarbeitung auf Ensemble-Performanz 44
3.4 Zusammenfassung 45

4 Konzept **47**
4.1 Allgemeines Konzept 47
4.2 Anforderungen an Datenvorverarbeitungsmethoden 48
4.3 Detailliertes Konzept 50

5 Prototypische Implementierung **59**
5.1 Verwendetes AutoML-Framework 59
5.2 Implementierungsentwurf des Konzeptes 61
5.3 Prototypische Implementierung in AutoML-Ensemble 64

6 Evaluation **69**
6.1 Versuchsaufbau 69
6.2 Verwendete Baselinemethoden 75
6.3 Evaluation und Diskussion der Performanz 76
6.4 Evaluation und Diskussion der Diversität 90
6.5 Weitere Analysen 94
6.6 Zusammenfassung der Evaluierungsergebnisse 98

7 Zusammenfassung **101**

8 Ausblick **103**

Literaturverzeichnis **105**

Abbildungsverzeichnis

2.1 Die Abbildung basiert auf [GLH15]. Mittels schwarzer Linie wird die Entscheidungsgrenze des Klassifikators für ein binäres Klassifikationsproblem illustriert. Die blauen Dateninstanzpunkte gehören derselben Klasse an und stellen die verschiedenen Arten von Dateninstanz dar. Dabei entsprechen die Dateninstanzen ohne Rand den *sicheren Dateninstanzen*, d.h. sie gehören eindeutig der blauen Klasse an. Die Dateninstanzen, mit einem durchgezogenen dunkelblauen Rand, sind *Grenz-Dateninstanzen*, da sie an der Entscheidungsgrenze des Klassifikators liegen. Die Dateninstanzen mit einem gestrichelten blauen Rand weisen *Klassenrauschen* auf, d.h. sie gehören eigentlich der lila Klasse an. 24

2.2 Die Abbildungen zeigen jeweils die Datencharakteristik des *Klassenungleichgewichts* mit gleich starker Ausprägung. In Abbildung (a) liegt lediglich ein *Klassenungleichgewicht* vor. Abbildung (b) zeigt sowohl *Klassenüberschneidung* als auch *Klassenungleichgewicht*. Der grün hinterlegte Bereich markiert den Bereich der Klassenüberschneidung. 24

2.3 Übersicht über die in der Arbeit behandelten Datenvorverarbeitungsphasen und die dazugehörigen Datenvorverarbeitungsschritte. 27

2.4 Ein Ensemble setzt sich aus k Basis-Klassifikatoren zusammen, deren Vorhersagen in einem Fusionsschritt F zu einer einzigen Vorhersage kombiniert werden. Abbildung basierend auf [VRM23]. 34

2.5 Darstellung einer AutoML-Pipeline, welche die Datenvorverarbeitungsmethoden mit optimiert. 39

4.1 Die Abbildung zeigt ein Ensemble, welches sich aus k Basis-Klassifikatoren zusammensetzt, wobei jeder Basis-Klassifikator eine eigene Datenvorverarbeitungspipeline (DVP) aufweist. 48

4.2 Die Abbildung bietet eine Übersicht über die ausgewählten Datencharakteristiken, die zugehörigen Vorverarbeitungsschritte sowie die jeweils ausgewählten Vorverarbeitungsmethoden für jeden Schritt der Vorverarbeitung. Die gestrichelte Linie bei der Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht bedeutet, dass lediglich bei Vorliegen dieser Datencharakteristik nach der Überabtastung eine Unterabtastung durchgeführt wird. Dies ist nicht der Fall, wenn ein Mangel an Daten vorliegt. 56

4.3 Die dargestellte Vorverarbeitungspipeline zeigt die Reihenfolge der Anwendung der ausgewählten Vorverarbeitungsschritte, basierend auf der Arbeit Garcías et al. [GLH16]. Die weiß hinterlegten Vorverarbeitungsschritte entsprechen den in Garcías et al. Arbeit behandelten Methoden, während die grau hinterlegten Schritte zusätzliche Erweiterungen darstellen, die in dieser Arbeit ergänzt werden. 57

5.1 Schematische Darstellung der implementierten Pipeline in *AutoML-Ensemble*. . 60

5.2 Schematische Darstellung der implementierten Pipeline in *AutoML-Ensemble* mit Integration des entwickelten Konzeptes. Die Datenvorverarbeitungsmethoden werden gemeinsam mit den Klassifikatoren im 2.) Schritt optimiert. 62
5.3 Das Diagramm zeigt eine vereinfachte Darstellung des Pipeline-Pakets des *AutoML-Ensemble* Frameworks. BasePreprocessor zeigt die generische Klasse, deren Struktur von den Vorverarbeitungsmethoden übernommen wird. Die hinzugefügten Vorverarbeitungsmethoden sind in den Paketen featur_engineering und sampling enthalten. 65

6.1 Schematische Darstellung der implementierten *statischen Pipeline* im *AutoML-Ensemble* Framework. Im Gegensatz zur *optimierten Pipeline* (vgl. Abbildung 5.1) werden die Datenvorverarbeitungsmethoden direkt auf den Datensatz angewendet, bevor das Training des Ensembles stattfindet. 76
6.2 Vergleich der Leistung zwischen dem Ensemble mit der optimierten Pipeline und dem besten Basis-Klassifikator, für alle 23 Datensätze, anhand der Performanzmetrik Balanced Accuracy. 77
6.3 Aufzeigen des Performanzunterschieds anhand der Balanced Accuracy für alle Datensätze. Balken, welche blau und positiv sind, zeigen, dass das Ensemble besser abgeschnitten hat, negative orange Balken bedeuten, dass der beste Basis-Klassifikator besser ist. 79
6.4 Aufzeigen des Performanzunterschieds anhand der Balanced Accuracy für alle Datensätze, welche mindestens die Datencharakteristik eine geringe Datenmenge enthalten. Blaue Balken zeigen, dass das Ensemble besser abschneidet. Orangefarbene Balken bedeuten, dass der beste Basis-Klassifikator besser ist. 80
6.5 Vergleich der Performanz des Ensembles mit der optimierten Vorverarbeitungspipeline und der Performanz des Ensembles mit der statischen Vorverarbeitungspipeline. 82
6.6 Aufzeigen des Performanzunterschieds anhand der Balanced Accuracy für alle Datensätze. Balken, welche blau und positiv sind, zeigen, dass das Ensemble mit der optimierten Pipeline besser abschneidet. Rote Balken bedeuten, dass das Ensemble mit der statischen Pipeline besser ist. 82
6.7 Darstellung des Zusammenhangs der Anzahl an generierten Klassifikatoren während der Klassifikatoroptimierungsphase und der erreichten Balanced Accuracy eines Durchlaufs. Die Durchläufe der optimierten Pipeline sind als Punkte dargestellt, während die Durchläufe der statischen Pipeline durch Dreiecke gekennzeichnet sind. 84
6.8 Darstellung der Performanzunterschiede der verschiedenen Durchläufe für die Datensätze der Gruppen 3 und 4. Die verschiedenen Durchläufe werden in dem Diagramm als 1.Lauf, 2. Lauf und 3. Lauf betitelt. Der blaue Balken zeigt den Unterschied der Performanz zwischen dem optimierten Ensemble und dem besten Basis-Klassifikator. Der rote Balken zeigt den Performanzunterschied zwischen der optimierten und der statischen Vorverarbeitungspipeline. 86
6.9 Vergleich der Performanz, anhand der Balanced Accuracy, für die Ensembles der optimierten Pipeline gegenüber den Ensembles mit impliziter Diversität. Die Ensembles mit impliziter Diversität sind Random Forest und AdaBoost. 90
6.10 Vergleich der Diversität zwischen den Ensembles der optimierten Pipeline und den Ensembles der statischen Pipeline, anhand der normalisierten Doppelfehlerrate. . 91

6.11 Darstellung der Performanz- und Diversitätsdifferenz für die Ensembles der optimierten und der statischen Pipelines. Die dunkle Farbe gibt hierbei die Performanzdifferenz der Balanced Accuracy an. Die helleren Farben geben die Diversitätsdifferenz an. Zeigen die Balken nach oben, so hat das Ensemble mit der optimierten Pipeline eine höhere Performanz oder Diversität erreicht. Wenn die Balken nach unten zeigen, so hat Random Forest eine höhere Performanz oder Diversität erreicht 92
6.12 Vergleich der Diversität zwischen den Ensembles mit der optimierten Pipeline (blau), den Random Forest Ensembles (dunkelgrün) und AdaBoost (hellgrün). . . 92
6.13 Darstellung der Performanz- und Diversitätsdifferenz zwischen den Ensembles mit optimierter Pipeline und den Random Forest Ensembles. Die dunkle Farbe gibt hierbei die Performanzdifferenz an und die hellere Farbe die Diversitätsdifferenz. Zeigen die Balken nach oben und sind blau, so schneidet die optimierte Pipeline besser ab. 93
6.14 Darstellung der Performanz- und Diversitätsdifferenz für die Ensembles mit optimierter Pipeline und AdaBoost. Die dunkle Farbe gibt hierbei die Performancedifferenz an und die hellere Farbe die Diversitätsdifferenz. Zeigen die Balken nach oben und sind blau, so schneidet die optimierte Pipeline besser ab. 94
6.15 Darstellung der durchschnittlich erreichten Performanz und Diversität über alle Datensätze, sowie der erreichten durchschnittlichen Performanz und Diversität für die verschiedenen Datencharakteristiken. Die dargestellten Datencharakteristiken sind: *Geringe Datenmenge (GD)*, *Hochdimensionale Daten (HD)*, *Klassenrauschen (KA)* und *Klassenüberschneidung mit Klassenungleichgewicht (KK)* 95
6.16 Vergleich der Häufigkeit, mit der jede Datenvorverarbeitungsmethode in ihrem Datenvorverarbeitungsschritt ausgewählt wird. Die Vorverarbeitungsmethoden sind wie folgt abgekürzt: Ensemble-Filter (EF), Feature Importance (FI), Mutual Information (MI), Kernel PCA (K. PCA), Random Oversampling (ROS). 97
6.17 Häufigkeitsanteil, wie oft jedes Rauschunterdrückungsverfahren in einer Gruppe gewählt wurde. 97
6.18 Performanzdifferenz zwischen der verwendeten und der neuen Version des AutoML-Frameworks: Lila Balken zeigen Datensätze, bei denen die neue Version eine bessere Performanz erzielt hat, während blaue Balken Datensätze darstellen, bei denen die verwendete Version überlegen war. 99

Tabellenverzeichnis

2.1 Die Tabelle zeigt den prozentualen Anteil identischer und komplementärer Entscheidungen zweier Klassifikatoren. 37

4.1 Auflistung der vergleichbaren Anforderungen V2-V4 für die ausgewählten Methoden. Die Hyperparameter sind wie folgt angegeben: Gesamt (kategorische/ Numerische). Die Symbole für die Laufzeiten sind wie folgt: n - Anzahl der Dateninstanz eines Datensatzes, $n_{minority}$ - Anzahl der Minderheitsinstanzen, $n_{synthetisch}$ - Anzahl der synthetischen Instanzen, CV - Anzahl der Splits für die Cross-Validierung, m - Anzahl der Merkmale eines Datensatzes, i - Iterationen einer Methode, V_x/V_y - Anzahl der Permutationen für die jeweilige Variable, num_{Tree} - Anzahl der Bäume im Random Forest (RF), m' - Anzahl zu betrachtender Merkmale an jedem Knoten im RF, n' - Bootstrapped Trainingsdaten für RF, $|S|$ - Anzahl der Merkmalsuntermengen, k - Anzahl der Nachbarn, d - Anzahl der Input-Dimensionen, m_{new} - Anzahl der neuen, niedrigeren Dimensionen. 51

5.1 Auflistung der angewandten Vorverarbeitungsschritte und verwendeten Vorverarbeitungsmethoden. Es werden die verwendete Bibliothek sowie die Anzahl der Hyperparameter aufgelistet. Die Hyperparameter sind wie folgt angegeben: Gesamt (kategorische/ numerische) 63

6.1 Die Tabelle gibt eine Übersicht über die verschiedenen Gruppen, die jeweils enthaltenen IDs in einer Gruppe, sowie die Anzahl der Datencharakteristiken, die in einer Gruppe betrachtet werden. 70

6.2 Auflistung der verwendeten Datensätze zur Evaluierung des Konzepts. Für jeden Datensatz wird die ID, der Name, die enthaltenen Datencharakteristiken sowie die Anzahl der Merkmale, Instanzen und Klassen angegeben. Datensätze, welche im Namen *Synth* enthalten, sind synthetische Datensätze. Die restlichen Datensätze sind Echtwelt Datensätze. Die Abkürzungen der Datencharakteristiken sind wie folgt: *Geringe Menge an Daten (GD)*, *Hochdimensionale Daten (HD)*, *Klassenrauschen (KA)* und *Klassenüberschneidung mit Klassenungleichgewicht (KK)* 71

6.3 Angabe der verwendeten Python-Bibliotheken des AutoML-Frameworks, so wie deren Verwendungszweck. 74

6.4 Vergleich der Ergebnisse des Ensembles mit der optimierten Pipeline gegenüber den Ergebnissen des besten Basis-Klassifikators aus einem Ensemble. Die Werte sind auf drei Nachkommastellen gerundet, der dick hinterlegte Lauf für einen Datensatz erzielte das beste Ergebnis. Unterstrichene Ergebnisse weisen dieselbe Performanz für beide Ansätze auf. Die letzte Zeile stellt die durchschnittliche Performanz eines Ansatzes dar. 78

6.5 Vergleich der Ergebnisse des Ensembles mit der optimierten Pipeline gegenüber den Ergebnissen der Ensembles der statischen Pipeline. Die Diversität basiert auf der normalisierten Doppelfehlerrate. Die Werte sind auf drei Nachkommastellen gerundet, der dick hinterlegte Lauf für einen Datensatz erzielte das beste Ergebnis. Unterstrichene Ergebnisse haben dieselbe Performanz für beide Ansätze. Die letzte Zeile stellt die durchschnittliche Performanz eines Ansatzes dar. 83
6.6 Vergleich der Ergebnisse des Ensembles mit der optimierten Pipeline gegenüber den Ergebnissen der Ensembles mit impliziter Diversität (Random Foresst und AdaBoost). Die Diversität basiert auf der normalisierten Doppelfehlerrate. Die Werte sind auf drei Nachkommastellen gerundet, der dick hinterlegte Durchlauf für einen Datensatz erzielte das beste Ergebnis. 89
6.7 Die Tabelle zeigt die durchschnittlich erreichte Performanz und Diversität über alle Datensätze, sowie die erreichte durchschnittliche Performanz und Diversität für die verschiedenen Datencharakteristiken. 95

Verzeichnis der Listings

5.1 Anwendungsbeispiel des Frameworks AutoMLEnsemble Frameworks mit angewandter Datenvorverarbeitung. 66
5.2 Ausgabe für das Anwendungsbeispiel Listing 5.1. 67

Abkürzungsverzeichnis

AutoML Automatisierte Maschinelle Lernen. 38

CASH Combined Algorithm Selection and Hyperparameter Optimization. 38

DF Doppelfehlerrate. 37

ENN Edited-Nearest-Neigbour. 28

IPF Iterative Partitioning Filter. 27

kNN k-Nearest-Neighbour. 28

LDA Linear Discriminant Analysis. 26

MRMR Minimum Redundancy, Maximum Relevance. 30

PCA Principal Component Analysis. 31

RBF Radial Basis Function. 53

Symbole und Notationen

Auf dieser Seite werden Symbole und Notationen aufgeführt, welche im Verlauf der Arbeit häufiger verwendet werden. Für die Notation der Symbole gilt allgemein folgende Konvention:

Skalar	klein	x, y
Vektor	klein, fett	$\mathbf{x}, \mathbf{y}$
Menge	große	X, Y, S
Matrix	groß, fett	$\mathbf{X}, \mathbf{Y}$

Nachstehend sind die häufig verwendeten Symbole aufgeführt:

Symbol	**Erklärung**
$\mathbf{X}$	Datensatz
n	Größe einer Menge u. a. Anzahl der Dateninstanzen eines Datensatzes
m	Anzahl der Merkmale eines Datensatzes
T_i	Teilmenge i des Datensatzes D
$\mathbf{x_i}$	i-te Dateninstanz eines Datensatzes
y_i	Klassenlabel der Dateninstanz $\mathbf{x_i}$
Ω	Menge an möglichen Klassen eines Datensatzes
ω_i	Klasse i aus Ω
F	Menge an Klassifikatoren
f	Klassifikator

1 Einleitung

Maschinelles Lernen wird in verschiedenen Bereichen verwendet und findet beispielsweise Anwendung in der Medizin zum Treffen von Diagnosen sowie in der Industrie zur Identifikation von mangelhaften Produkten [HRM19; MS21]. Eine wesentliche Einschränkung maschineller Lernalgorithmen besteht jedoch darin, dass diese häufig Annahmen über die Struktur der Daten voraussetzen, die in realen Szenarien oft nicht erfüllt sind. So wird typischerweise angenommen, dass jede Klasse im Datensatz gleichmäßig repräsentiert ist, dass alle enthaltenen Merkmale relevant sind und dass ausreichend Daten zum Trainieren eines Lernalgorithmus vorhanden sind [DDC18]. Beim Vorliegen derartiger Datencharakteristiken in einem Datensatz haben sich Ensembles, welche aus mehreren Klassifikatoren bestehen, bewährt. Diese erreichen bei komplexen Datencharakteristiken eine bessere Leistung als einzelne Klassifikatoren [Die00; Pol06].

Ensembles stellen eine vielversprechende Lösung für Klassifikationsprobleme dar, bei denen einzelne Algorithmen des maschinellen Lernens keine zufriedenstellende Leistung erzielen. Sie sind dafür bekannt, komplexe Datencharakteristiken wie eine geringe oder eine große Menge an Daten effizient zu handhaben [Die00; Pol06]. Es hat sich gezeigt, dass das Integrieren von komplementären Klassifikatoren in einem Ensemble zur Minimierung einer fehlerhaften Vorhersage beiträgt. Daher ist es das Ziel, ein möglichst diverses Ensemble zu erstellen. Ein Ensemble gilt als divers, wenn die einzelnen Klassifikatoren des Ensembles auf unterschiedlichen Dateninstanzen fehlerhafte Vorhersagen treffen [Kun04; Pol06]. Es existieren verschiedene Methoden zur Erhöhung der Diversität in einem Ensemble. Jedoch wurde bislang nicht untersucht, inwieweit sich die Diversität eines Ensembles durch die gezielte Anwendung von Vorverarbeitungsmethoden, entsprechend den Datencharakteristiken eines Datensatzes, erhöhen lässt.

Ziel ist es, durch die gezielte Vorverarbeitung die Diversität im Ensemble zu erhöhen und damit die Klassifikationsleistung des Ensembles zu verbessern. Aus diesem Grund werden für das Konzept vier Datencharakteristiken und entsprechende Vorverarbeitungsmethoden ausgewählt.

Die Beiträge dieser Arbeit bestehen zum einen in der Erarbeitung eines Konzepts, das durch den gezielten Einsatz von Vorverarbeitungsmethoden die Diversität und Klassifikationsgenauigkeit eines Ensembles steigert. Zum anderen in der Evaluierung des Konzepts und der damit verbundenen Untersuchung der Zusammenhänge von Datencharakteristiken, Vorverarbeitungsmethoden, Diversität und Performanz. Zu diesem Zweck werden Anforderungen für die Auswahl geeigneter Vorverarbeitungsmethoden erarbeitet. Ein wichtiger Aspekt ist dabei die prototypische Implementierung des Konzepts in einem AutoML-Tool, da diese Einfluss auf die Auswahl geeigneter Vorverarbeitungsmethoden hat. Das Integrieren des Konzepts in ein AutoML-Framework hat den Vorteil, dass die Auswahl und Optimierung der Vorverarbeitungsmethoden und Klassifikationen automatisiert durchgeführt wird. Das entwickelte Konzept wird anschließend anhand von 23 Datensätzen evaluiert, die sowohl Echtwelt- als auch synthetische Datensätze umfassen. Zur Bewertung des Konzepts werden Vergleichsläufe durchgeführt, um zu untersuchen, wie sich das Konzept auf die Diversität und Klassifikationsgenauigkeit des Ensembles auswirkt.

Die Arbeit ist wie folgt gegliedert:

Kapitel 2 – Grundlagen: In den Grundlagen werden zentrale Begriffe und Terminologien eingeführt, auf die die nachfolgenden Kapitel aufbauen. Insbesondere werden Datencharakteristiken, Datenvorverarbeitungsmethoden, automatisiertes maschinelles Lernen sowie Ensembles vorgestellt. Bei den Ensembles liegt ein zentraler Aspekt auf dem Erzeugen von Diversität im Ensemble.

Kapitel 3 – Verwandte Arbeiten: Um den Beitrag dieser Arbeit von der bisherigen Forschung abzugrenzen, stellt dieses Kapitel Ansätze zur Erzeugung von impliziter Diversität in Klassifikationsensembles vor. Weiter werden Studien vorgestellt, die den Einfluss von Vorverarbeitungsmethoden auf die Klassifikationsgenauigkeit von Ensembles und einzelnen Klassifikatoren untersuchen. Darüber hinaus werden AutoML-Tools betrachtet, welche die Erstellung von Ensembles sowie die Anwendung von Vorverarbeitungsmethoden unterstützen.

Kapitel 4 – Konzept: Hier werden die Anforderungen für die Wahl der Vorverarbeitungsmethoden identifiziert und es wird diskutiert, wie sich anhand dieser die Diversität in einem Ensemble erhöhen lässt. Auf Basis dieser Anforderungen wird ein Konzept vorgestellt, welches, durch den gezielten Einsatz von Vorverarbeitungsmethoden in Abhängigkeit von Datencharakteristiken, die Klassifikationsgenauigkeit und Diversität eines Ensembles erhöhen soll.

Kapitel 5 – Prototypische Implementierung: Dieses Kapitel befasst sich mit der prototypischen Implementierung des erarbeiteten Konzepts in einem AutoML-Ensemble Framework.

Kapitel 6 – Evaluation: Hier wird anhand von Echtwelt- und synthetischen Datensätzen das erarbeitete Konzept evaluiert.

Kapitel 7 – Zusammenfassung: Abschließend wird der Beitrag dieser Arbeit zusammengefasst und ein Fazit zum erarbeiteten Konzept gegeben.

Kapitel 8 - Ausblick: In diesem Kapitel wird ausgehend von den Ergebnissen und dem Fazit der Arbeit auf mögliche Anknüpfungspunkte für zukünftige Arbeiten eingegangen.

2 Grundlagen

In diesem Kapitel werden die grundlegenden Konzepte und Begriffe dargelegt, die für das Verständnis und die Nachvollziehbarkeit dieser Masterarbeit von Bedeutung sind. Der Aufbau dieses Kapitels orientiert sich an den klassischen Schritten einer Pipeline für maschinelles Lernen, welche unter anderem folgende Phasen umfasst [Kég17; Zho18]:

1. Datensammlung
2. Datenvorverarbeitung
3. Modellgenerierung und -evaluation

In Abschnitt 2.1 erfolgt zunächst eine formale Definition der Klassifikation sowie des Klassifikators, welche für das Verständnis der nachfolgenden Phasen des maschinellen Lernens von Bedeutung sind und zugleich eine Einschränkung hinsichtlich der Auswahl der Datenvorverarbeitungsmethoden mit sich bringt. Abschnitt 2.2 erläutert, welche Datencharakteristiken ein Datensatz aufweisen kann. Der nachfolgende Abschnitt 2.3 widmet sich den verschiedenen Phasen der Datenvorverarbeitung und den dabei verwendeten Vorverarbeitungsmethoden, wobei der Fokus auf jenen Vorverarbeitungsmethoden liegt, welche im Verlauf der Arbeit verwendet werden. In Abschnitt 2.4 wird das Konzept des Ensemble-Lernens erläutert. Abgeschlossen wird das Kapitel mit Abschnitt 2.5, in dem das Konzept des automatisierten maschinellen Lernens dargelegt wird.

2.1 Klassifikation

Die vorliegende Arbeit beschäftigt sich mit überwachtem maschinellem Lernen für Klassifikationsprobleme, die in tabellarischer Form beschrieben sind. Hierbei liegen die Daten zu den Instanzen als Matrix der Form $\mathbf{X} \in \mathbb{R}^{n \times m}$ vor, wobei die n Zeilen die einzelnen Dateninstanzen repräsentieren, welche durch m Merkmale beschrieben sind. Der Zugriff auf die i-te Dateninstanz und das j-te Merkmale einer Instanz wird als $x_{i,j}$ definiert. Basierend auf den m Merkmalen wird jeder Instanz des Datensatzes eine der Klassen ω_j aus $\Omega = \{\omega_1, \omega_2, ..., \omega_l\}$ zugeordnet. Diese Zuordnung wird als Erkennen von Mustern in den Daten bezeichnet [Kun04].

Klassifikation bezeichnetdas automatisierte Zuordnen einer Klasse zu einer nicht klassifizierten Dateninstanz $\mathbf{x_i}$ [Bur19]. Ein Klassifikator wird durch die Funktion $f : \mathbf{X} \longrightarrow \mathbf{Y}$ definiert, wobei die zugeordnete Klasse für die Instanz $\mathbf{x_i}$ als $\widehat{y_i}$ benannt wird. Diese Zuordnung wird als Vorhersage des Klassifikators angesehen. Für die Erstellung eines Klassifikators werden bereits klassifizierte Dateninstanzen benötigt. Das bedeutet, dass den Instanzen zuvor eine Klasse zugewiesen wird. Mit diesen Dateninstanzen kann der Klassifikator f trainiert und anschließend verwendet werden, um neue, bisher nicht klassifizierte Dateninstanzen zu klassifizieren. Dem Klassifikator liegt ein Lernalgorithmus zugrunde, auf dessen Basis der Klassifikator erstellt wurde.

Der Lernalgorithmus definiert die Regeln, auf deren Basis das Modell erstellt wird; das Modell ist somit das Ergebnis des Trainingsprozesses des Lernalgorithmus. Das erstellte Modell ist abhängig von den zugrundeliegenden Daten und Hyperparametern, die für die Erstellung des Modells verwendet wurden. Klassifikatoren sind ein Modell, das speziell für die Klassifikation geeignet ist [BN06; Bur19; Kun04].

Die nachfolgenden behandelten Methoden beziehen sich ausschließlich auf diesen Bereich des maschinellen Lernens.

2.2 Datencharakteristiken

Das Feststellen von Datencharakteristiken eines Datensatzes kann helfen, die Auswahl für geeignete Vorverarbeitungsmethoden und Klassifikatoren einzugrenzen [BCSV08]. Abhängig davon, ob die Merkmale in numerischer oder kategorischer Form vorliegen, werden beispielsweise bestimmte Klassifikatoren bevorzugt verwendet. Es gibt verschiedene Arten von Datencharakteristiken, wobei einige durch deskriptive Messungen leicht festgestellt werden können, unter anderem [BCSV08]:

- Anzahl der Dateninstanzen, n;
- Anzahl der Merkmale, m;
- Anzahl der Klassen, $|\Omega|$;
- Anteil der diskreten Merkmale;
- Anteil der fehlenden Werte eines Merkmals;
- Anzahl der Dateninstanzen pro Klasse.

Neben den durch deskriptive Messungen festzustellenden Datencharakteristiken existieren auch komplexe Datencharakteristiken. In dieser Arbeit sind komplexe Datencharakteristiken als solche definiert, die schwer durch deskriptive Messungen festzustellen sind oder keiner eindeutigen mathematischen Definition folgen. Im Folgenden werden ausschließlich Datencharakteristiken aufgeführt, die im Verlauf der Arbeit behandelt werden.

Irrelevante und redundante Merkmale

Irrelevante Merkmale sind Merkmale, die keinen Einfluss auf die Vorhersage des Klassifikators haben und deren Werte für jede Dateninstanz zufällig vorliegen [KKP06]. Ein *redundantes Merkmal* liegt vor, wenn ein anderes Merkmal dieselben Informationen enthält [KKP06]. Das Einbinden irrelevanter und redundanter Merkmale verlängert die Trainingszeit von Klassifikatoren, ohne nützliche Informationen beizutragen, und kann die Performanz des Klassifikators verschlechtern [JM06; KKP06; WJJ21].

Geringe Menge an Daten

Eine *geringe Datenmenge* beschreibt das Szenario, in dem ein Klassifikator nicht genügend Daten zur Verfügung hat, um zugrundeliegende Muster in den Daten zu erkennen [FGG+18]. Das Vorhandensein einer zu geringen Datenmenge kann zu einer Überanpassung des Klassifikators führen [FGG+18]. Ein überangepasster Klassifikator erzielt eine ausgezeichnete Performanz auf den Trainingsdaten, versagt jedoch bei der Vorhersage neuer, unbekannter Dateninstanzen [Bur19]. Das Problem der geringen Datenmenge tritt unter anderem im medizinischen Bereich auf [Abu13; MOAD21; MS21; MY19; RRL+20].

Hochdimensionale Daten

Die Datencharakteristik *Hochdimensionale Daten* bezieht sich auf Datensätze mit einer Vielzahl an Merkmalen im Verhältnis zur Anzahl der Dateninstanzen. Der Begriff Dimension wird hierbei als Synonym für den Begriff Merkmal verwendet. Hochdimensionale Datensätze sind vor allem solche, bei denen die Anzahl der Merkmale die der Dateninstanzen übersteigt [HTFF09]. Weiter wird zwischen Datensätzen, die viele Merkmale, aber nur wenige Dateninstanzen aufweisen, und solchen, die sowohl viele Merkmale als auch viele Dateninstanzen enthalten, unterschieden [Sch20]. Grundsätzlich wächst die Menge an Daten, die benötigt werden, um einen genauen und generalisierten Klassifikator zu erstellen, exponentiell mit der Anzahl der Merkmale [Sch20]. Dies wird auch als „Fluch der Dimensionalitäten“ bezeichnet und stellt eine Herausforderung dar, da die Erstellung eines Klassifikators erheblich erschwert und äußerst rechenintensiv wird [Sch20]. Hochdimensionale Daten treten in verschiedenen Bereichen auf, unter anderem in der Genomik [HTFF09], den Finanzen, der Wirtschaft und der Astronomie [Sch20].

Rauschen

Es gibt zwei verschiedene Arten von Rauschen: *Merkmalsrauschen* und *Klassenrauschen* [FGG+18]. Merkmalsrauschen bezieht sich auf Situationen, in denen ein oder mehrere Merkmale einer Dateninstanz Rauschen enthalten [GLH16]. Der Begriff Klassenrauschen bezeichnet Dateninstanzen, die ein falsches Klassenlabel erhalten haben [GLH16]. Das Vorhandensein von Klassenrauschen erschwert es dem Klassifikator, Muster innerhalb der Daten zu erlernen. Als Folge dessen wird die Performanz des Klassifikators hinsichtlich der Genauigkeit sowie der benötigten Erstellungszeit beeinträchtigt [GLH15; ZW04]. Fehlerhafte Klassenlabels treten häufig in Echtwelt-Datensätze. auf [FGG+18]. In Abbildung 2.1 ist dargestellt, wie Klassenrauschen in einem Datensatz aussehen kann.

Klassenungleichgewicht

Eine ungleichmäßige Verteilung der Klassen führt zu einem *Klassenungleichgewicht* und ist in Abbildung 2.2a dargestellt. Der Grad des Klassenungleichgewichts für einen Datensatz mit zwei Klassen kann durch das *Ungleichgewichtsverhältnis (UV)* ermittelt werden [FGG+18]:

$$UV = \frac{Mehrheitsklasse}{Minderheitsklasse}$$

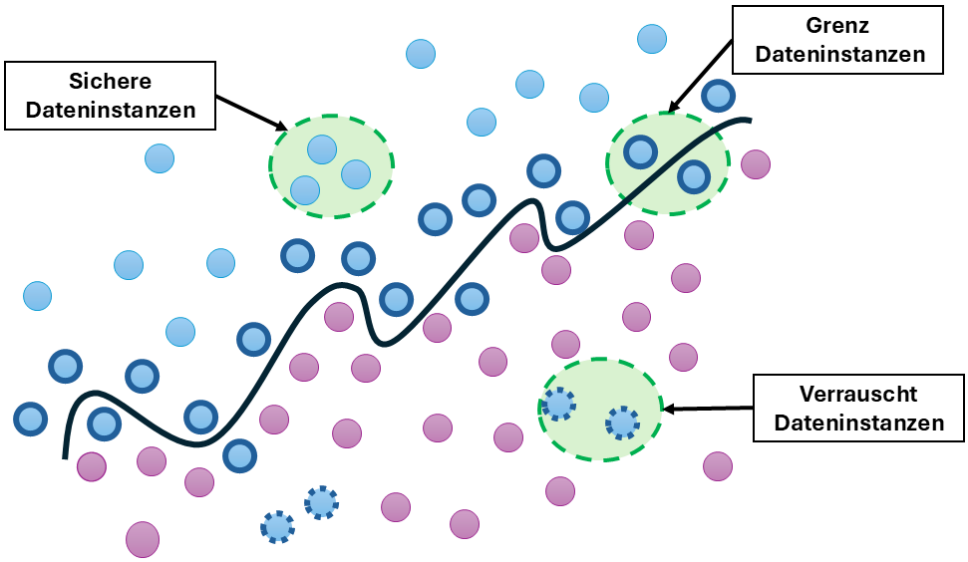

Abbildung 2.1: Die Abbildung basiert auf [GLH15]. Mittels schwarzer Linie wird die Entscheidungsgrenze des Klassifikators für ein binäres Klassifikationsproblem illustriert. Die blauen Dateninstanzpunkte gehören derselben Klasse an und stellen die verschiedenen Arten von Dateninstanz dar. Dabei entsprechen die Dateninstanzen ohne Rand den *sicheren Dateninstanzen*, d.h. sie gehören eindeutig der blauen Klasse an. Die Dateninstanzen, mit einem durchgezogenen dunkelblauen Rand, sind *Grenz-Dateninstanzen*, da sie an der Entscheidungsgrenze des Klassifikators liegen. Die Dateninstanzen mit einem gestrichelten blauen Rand weisen *Klassenrauschen* auf, d.h. sie gehören eigentlich der lila Klasse an.

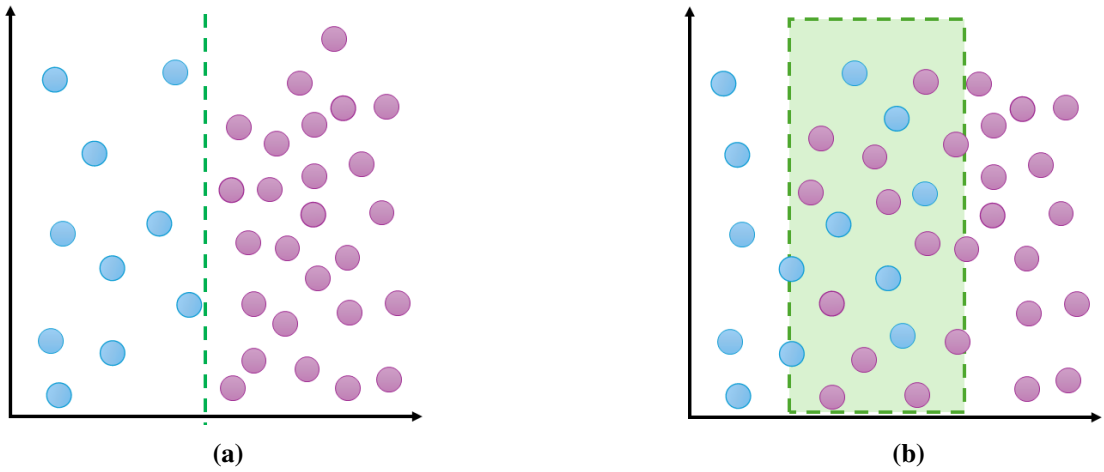

Abbildung 2.2: Die Abbildungen zeigen jeweils die Datencharakteristik des *Klassenungleichgewichts* mit gleich starker Ausprägung. In Abbildung (a) liegt lediglich ein *Klassenungleichgewicht* vor. Abbildung (b) zeigt sowohl *Klassenüberschneidung* als auch *Klassenungleichgewicht*. Der grün hinterlegte Bereich markiert den Bereich der Klassenüberschneidung.

Die *Mehrheitsklasse* ist diejenige Klasse, die die meisten Dateninstanzen in einem Datensatz aufweist, während die *Minderheitsklasse* diejenige ist, die die wenigsten Dateninstanzen im Datensatz aufweist. Ein Datensatz wird als unausgeglichen bezeichnet, wenn $UV > 1$. Ein Klassenungleichgewicht ist insbesondere dann problematisch, wenn die Minderheitsklasse signifikant weniger Dateninstanzen enthält als die Mehrheitsklasse. Das führt häufig dazu, dass Klassifikatoren zugunsten der Mehrheitsklasse verzerrt sind, da der Klassifikator darauf ausgerichtet ist, eine hohe Genauigkeit zu erreichen. Bei einem signifikanten Ungleichgewicht ist das Erreichen einer hohen Genauigkeit durch das Ignorieren der Minderheitsklasse möglich [DDC18; FGG+18; LFG+13]. Zum Beispiel würde ein Klassifikator in einem Szenario, in dem zwei Klassen vorliegen und die Mehrheitsklasse 95 % der Dateninstanzen ausmacht, eine Genauigkeit von 95 % erreichen, wenn dieser ausschließlich die Mehrheitsklasse vorhersagt. Das Problem des Klassenungleichgewichts tritt in einer Vielzahl von Echtwelt-Datensätzen auf, darunter in der Erkennung von Eindringlingen in ein System und in der Medizin [SLSH15; VEP21].

Klassenüberschneidung

Eine *Klassenüberschneidung* liegt vor, wenn Dateninstanzen aus verschiedenen Klassen denselben Merkmalsraum teilen [LFG+13; VEP21], siehe Abbildung 2.2b. Teilen sich Klassen denselben Merkmalsraum, so weisen Merkmale der Dateninstanzen verschiedener Klassen dieselben Werte auf. Die Ausprägung der Klassenüberschneidung kann variieren, je stärker die Klassenüberschneidung, desto schlechter ist die Performanz eines Klassifikators [VEP21]. Bisher ist das Problem der Klassenüberschneidung nicht formal mathematisch definiert [SAJ+22; SW11]. Dateninstanzen, die sich in der Grenzregion der Entscheidungsgrenze eines Klassifikators befinden, werden auch als Grenz-Dateninstanzen bezeichnet, siehe Abbildung 2.1. Die Entscheidungsgrenze eines Klassifikators definiert, ab welchen Merkmalswerten eine Dateninstanz einer bestimmten Klasse zugeordnet wird [Bur19].

In der Literatur wird das Problem der *Klassenüberschneidung* häufig zusammen mit dem Problem des *Klassenungleichgewichts* behandelt, da beide Probleme erhebliche Herausforderungen für die Klassifikation darstellen [SAJ+22]. Liegt in dem Bereich der Klassenüberschneidung auch ein Klassenungleichgewicht vor, verschiebt sich die Entscheidungsgrenze tendenziell zugunsten der Mehrheitsklasse [SAJ+22]. Zwischen den beiden Problemen besteht eine bisher unvollständig verstandene Abhängigkeit [KKP06; SAJ+22]. Faktoren wie die Dimensionalität der Daten, die Komplexität der Entscheidungsgrenze und die lokalen Eigenschaften der Überschneidungsregion, einschließlich des lokalen Klassenungleichgewichts, sind dabei von Bedeutung [GMS08; MSA+18; SAJ+22; WW17]. Das Problem der Klassenüberschneidung mit Klassenungleichgewicht wird mit zunehmender Dimensionalität der Merkmale eines Datensatzes abgeschwächt. Dies kann auf die Veränderungen der Datendichte in höheren Dimensionen zurückgeführt werden [GMS08].

2.3 Datenvorverarbeitungsmethoden

Die Datenvorverarbeitung stellt einen wesentlichen Schritt im Rahmen des maschinellen Lernens dar. Lernalgorithmen benötigen eine möglichst exakte und störungsfreie Darstellung der Daten, um eine optimale Performanz zu erreichen. Störungsfreie Daten sind solche, die unter anderem frei von Rauschen und fehlenden Werten sind. Die Wahl der Vorverarbeitungsmethode ist dabei von den

zugrunde liegenden Daten sowie dem gewählten maschinellen Lernalgorithmus abhängig [AKV19; GLH16].

Die Datenvorverarbeitung lässt sich in verschiedene Phasen einteilen. In dieser Arbeit wird die Einteilung in *Datenbereinigung, Datentransformation, Datenreduktion* und *Datenerweiterung* gewählt. In der Literatur sind verschiedene Einteilungen der Phasen zu finden [FCW+21; GLH15; MJAE22]. Abhängig von dem Ziel der Phase sind entsprechende Schritte enthalten. So beinhaltet die Datenbereinigung Schritte zur Identifikation und Entfernung von fehlerhaften oder schwer zu klassifizierenden Instanzen. Schritte der Datentransformation zielen darauf ab, die Darstellung der Daten so zu transformieren, dass diese dazu geeignet sind, einen Klassifikator zu trainieren. Die Phase der Datenreduktion strebt an, die Datenmenge zu reduzieren, wobei keine wichtigen Informationen entfernt werden sollten. Das Ziel der Datenerweiterung ist es, zusätzliche Dateninstanzen zu generieren.

Eine Übersicht der verschiedenen Schritte, die in den jeweiligen Phasen der Datenvorverarbeitung enthalten sind, kann der Abbildung 2.3 entnommen werden. Die Abbildung umfasst lediglich Vorverarbeitungsschritte, welche für diese Arbeit relevant sind. Im Folgenden werden ausschließlich Datenvorverarbeitungsmethoden behandelt, die für tabellarische Daten und Klassifikationsprobleme geeignet sind und im weiteren Verlauf der Arbeit verwendet werden.

2.3.1 Datenbereinigung

Die *Datenbereinigung* dient der Aufbereitung der Rohdaten, sodass diese zum Trainieren von Klassifikatoren geeignet sind. Dazu gehören die Identifikation und Eliminierung von Ausreißern und Rauschen sowie die Korrektur von Inkonsistenzen [GLH16].

Rauschunterdrückungsverfahren für Klassenrauschen

Die Verwendung von *Rauschunterdrückungsverfahren* erlaubt die Identifikation und Eliminierung von *Klassenrauschen*. Eine Möglichkeit zur Entfernung von Klassenrauschen ist die Anwendung von *Rauschfiltern* [AKV19]. Rauschfilter, die auf Ensemble-Basis arbeiten, sind beliebt, da diese sich nicht auf einen einzelnen Klassifikator stützen, um Klassenrauschen zu ermitteln [GLH16]. Eine häufig verwendete Methode ist die Ensemble-Filter Methode [BF99] und der Iterative Partitioning Filter [KR07].

Ensemble-Filter [BF99] stellen einen einfachen Ansatz zur Erkennung und Entfernung von *Klassenrauschen* dar [GLH16]. Die Grundidee der Methode ist, dass Dateninstanzen, die von einer Menge an Klassifikatoren falsch klassifiziert werden, als verrauscht gelten. Zum Klassifizieren der Instanzen werden die drei Klassifikatoren Entscheidungsbaum [Qui14], 1-Nearest-Neigbour [FH89] und Linear Discriminant Analysis (LDA) [Fis36] verwendet. Um festzustellen, ob eine Instanz verrauscht ist, kann ein Mehrheits- oder Konsensvotum verwendet werden. Beim Mehrheitsvotum ist es ausreichend, wenn zwei der Klassifikatoren eine Dateninstanz falsch klassifizieren, während beim Konsensvotum alle drei Klassifikatoren die Instanz falsch klassifizieren müssen, damit diese als

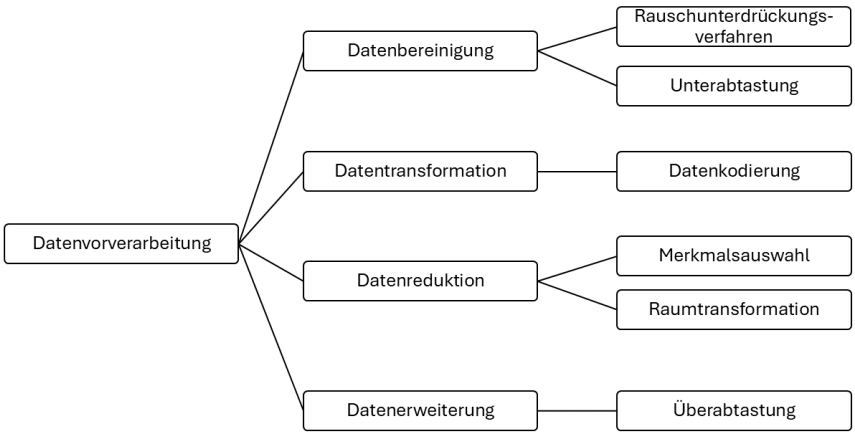

Abbildung 2.3: Übersicht über die in der Arbeit behandelten Datenvorverarbeitungsphasen und die dazugehörigen Datenvorverarbeitungsschritte.

verrauscht gilt. Eine falsch klassifizierte Dateninstanz ist dabei eine Instanz, der vom Klassifikator eine falsche Klasse zugeordnet wurde. Anschließend werden die Dateninstanzen, die als verrauscht gelten, aus dem Datensatz entfernt.

Der **Iterative Partitioning Filter (IPF)** [KR07] ist ein Ensemble basierter Filter und zeichnet sich dadurch aus, dass sowohl *Klassen-* als auch *Merkmalsrauschen* erkannt und entfernt wird [SLSH15]. Durch ein iteratives Verfahren werden in jedem Schritt verrauschte Dateninstanzen entfernt. Der iterative Prozess beinhaltet folgende Schritte: Zunächst wird der aktuelle Datensatz D in k gleich große Teilmengen $D = T_1, T_2, .., T_K$ aufgeteilt. Auf jeder Teilmenge T_j wird anschließend ein Entscheidungsbaum f_j trainiert, was die Klassifikatorenmenge $F = f_1, f_2, ..., f_K$ ergibt. Jeder dieser Klassifikatoren evaluiert dann den gesamten Datensatz D. Damit eine Dateninstanz $\mathbf{x_i}$ als verrauscht gilt, müssen zwei Kriterien erfüllt sein:

1. Die Dateninstanz $\mathbf{x_i} \in T_j$ muss von dem Klassifikator f_j, falsch klassifiziert werden.
2. Es wird ein Mehrheits- oder Konsensvotum angewandt, für das die Vorhersagen aller F Klassifikatoren, einschließlich der von f_j, berücksichtigt werden.

Im nächsten Schritt werden die als verrauscht identifizierten Instanzen aus dem Datensatz D entfernt, und die nächste Iteration wird auf dem reduzierten Datensatz durchgeführt.
Die Methode stoppt, wenn weniger als p Prozent der Instanzen über s aufeinanderfolgende Iterationen als verrauscht gekennzeichnet werden. Um die Laufzeit der Methode zu verkürzen, können vor dem Start einer neuen Iteration i Prozent der korrekt klassifizierten Instanzen aus dem Datensatz D entfernt werden. Diese werden nach dem Abschluss der Methode dem Datensatz wieder hinzugefügt.

Unterabtastung

Verfahren zur *Unterabtastung* erstellen durch Eliminierung ausgewählter Dateninstanzen eine modifizierte Version des originalen Datensatzes [LFG+13]. Die Unterabtastung wird häufig in Kombination mit einer *Überabtastungsmethode* als Lösungsansatz für die Datencharakteristik *Klassenungleichgewicht* oder *Klassenungleichgewicht mit Klassenüberschneidung* eingesetzt [SAJ+22].

Sind diese Datencharakteristiken in einem Datensatz vorhanden, werden typischerweise Instanzen der Mehrheitsklasse entfernt, mit dem Ziel, eine ausgewogene Klassenverteilung zu erlangen. So wird es dem Klassifikator erleichtert, eine Entscheidungsgrenze zu finden [FGG+18]. Dabei werden häufig Dateninstanzen der Mehrheitsklasse entfernt, welche nahe den Instanzen der Minderheitsklasse liegen. Eine weitere Möglichkeit zum Entfernen von Dateninstanzen ist das *Random Undersampling*, hier werden zufällig Instanzen der Mehrheitsklasse entfernt [FGG+18; LFG+13]. Weitere bekannte Algorithmen zur Unterabtastung, welche als Standard gelten [FGG+18], sind *Tomek-Link* [Tom76] und *Edited-Nearest-Neighbour* [Wil72].

Das Anwenden der **Tomek-Link Methode** [Tom76] nach einer *Überabtastungsmethode* kann dazu beitragen, dass die Klassen klarer unterscheidbar sind. [FGG+18]. Ein Tomek-Link zwischen zwei Dateninstanzen $\mathbf{x_i}$ und $\mathbf{x_j}$, welche zwei unterschiedlichen Klassen ω_1, ω_2 angehören wird wie folgt definiert: Es existiert ein Tomek-Link zwischen $E_{i,1} = (x_i, \omega_1)$ und $E_{j,2} = (x_j, \omega_2)$, wenn es keine weitere Dateninstanz $\mathbf{x_z}$ gibt, für die gilt: $d(E_{i,1}, E_{z,2}) < d(E_{i,1}, E_{j,2})$ oder $d(E_{j,2}, E_{z,1}) < d(E_{i,1}, E_{j,2})$, wobei $d(\cdot)$ die Distanz zwischen zwei Dateninstanzen darstellt und $E_{i,1} = (x_i, \omega_1)$ bedeutet, dass die Dateninstanz x_i der Klasse ω_1 angehört [Tom76]. Wird ein Tomek-Link gefunden, dann werden entweder die Dateninstanz der Mehrheitsklasse oder beide Dateninstanz entfernt [FGG+18; Tom76]. Durch das Anwenden der Tomek-Link Methode werden Dateninstanzen entfernt, die entweder Grenz-Dateninstanzen oder Verrauscht-Dateninstanzen sind (vgl. Abbildung 2.1). Das ist darauf zurückzuführen, dass nur diese zwei Arten an Dateninstanzen einen Tomek-Link aufweisen können [HM13].

Die **Edited-Nearest-Neigbour (ENN)** Methode [Wil72] verwendet den k-Nearest-Neighbour (kNN)-Algorithmus [FH89], um von einer Dateninstanz $\mathbf{x_i}$ die benachbarten Dateninstanzen zu identifizieren. Die Methode beinhaltet mehrere Schritte, zur Prüfung, ob eine Dateninstanz entfernt werden sollte. Zunächst werden für eine Instanz $\mathbf{x_i}$ die k nächsten Nachbarn berechnet. Anschließend wird die Klassenzugehörigkeit der k nächsten Nachbarn überprüft. Entspricht $\mathbf{x_i}$ nicht der Mehrheitsklasse seiner Nachbarinstanzen, so wird $\mathbf{x_i}$ entfernt. Wie auch die *Tomek-Link Methode* entfernt ENN Grenz-Dateninstanzen und verrauschte Dateninstanzen, allerdings weniger konservativ wie die Tomek-Link Methode, insbesondere wenn gilt $k > 1$ [GLH16]. Gilt, $k = 1$ entspricht ENN der Tomek-Link Methode.

2.3.2 Datentransformation

Die *Datentransformation* als Vorverarbeitungsphase bezeichnet in dieser Arbeit die Kodierung von Daten, sodass diese für maschinelle Lernalgorithmen geeignet sind [GLH15].

Die **Datenkodierung** bezeichnet die Umwandlung von kategorischen Daten in eine numerische Darstellung. Dies ist notwendig, da manche Klassifikationsalgorithmen ausschließlich mit nummerischen Daten arbeiten können [Bur19; GLH15]. Im Folgenden werden zwei Methoden vorgestellt, welche kategorische Daten in eine numerische Darstellung transformieren: Zum einen das *Label-Encoding*, welches auf die Klassenlabels angewandt wird, zum anderen das *One-hot-Encoding*, welches kategorische Merkmale in eine numerische Darstellung konvertiert.

Label-Encoding wird angewendet, wenn die Klassenlabel eines Datensatzes in kategorischer Form vorliegen. Durch das Anwenden eines *Label-Encoders*, wird jedem kategorischen Merkmal ein eindeutiger numerischer Wert zugewiesen. Diese Werte sind in der Regel aufeinanderfolgende ganze

Zahlen, die zwischen 0 und Klassenanzahl − 1 liegen. Wenn beispielsweise für das Klassifizieren von Pilzsorten die Klassenlabels *giftig* und *nicht giftig* vorhanden sind, so könnte das Mapping des *Label-Encoders* wie folgt aussehen: giftig → 0, nicht giftig → 1.

One-hot-Encoding dient dazu, eine Menge an endlichen kategorische Merkmale auf einen binären Vektor abzubilden. Weist ein Merkmal n mögliche Werte auf, so werden für dieses eine Merkmal n neue binäre Merkmale erstellt, wobei jedes Merkmal die Werte 0 oder 1 annehmen kann [MJAE22]. Liegt etwa das Merkmal Pilzfarbe vor, welches die Werte rot, weiß und braun annehmen kann, so werden für das Merkmal Pilzfarbe drei neue Merkmale erstellt: *Pilzfarbe_rot*, *Pilzfarbe_weiß* und *Pilzfarbe_braun*. Hat das ursprüngliche Merkmal *Pilzfarbe* den Wert *rot*, dann nehmen die neu erstellten Merkmale folgenden Werte an: *Pilzfarbe_rot* → 1, *Pilzfarbe_weiß* → 0 und *Pilzfarbe_Braun* → 0. Dieses Muster wird auf alle anderen Dateninstanzen entsprechend angewandt [MJAE22].

2.3.3 Datenreduktion

Das Ziel der Anwendung einer *Datenreduktion* ist es, die Menge der zu verarbeitenden Daten zu verringern, ohne dabei wesentliche Informationen oder Klassenmuster zu verlieren. Ein Vorteil der Datenreduktion ist, dass Klassifikatoren dadurch schneller und effizienter arbeiten können und der Speicherbedarf reduziert wird [AKV19; GLH16].

Merkmalsauswahl

Das effiziente Reduzieren von Daten kann durch die *Merkmalsauswahl* erfolgen, bei der *irrelevante* und *redundante Merkmale* identifiziert und entfernt werden [GLH16; JBB15]. Es gibt verschiedene Verfahren für die Merkmalsauswahl, in dieser Arbeit werden lediglich Merkmalsauswahlmethoden betrachtet, welche zu den *Filter-Methoden* gehören. Die Anwendung von Filter-Methoden bietet im Vergleich zu anderen Methoden die Vorteile eines geringeren Speicherbedarfs und einer kürzeren Laufzeit [FCW+21]. Zur Auswahl von Merkmalen können verschiedene Filter-Methoden herangezogen werden, die sich in *univariaten* und *multivariaten* Methoden differenzieren lassen [JBB15]. Bei der Wahl einer Methode ist zu beachten, dass diese sowohl für numerische als auch für kategorische Daten geeignet ist.

Univariate Methoden fokussieren sich auf einzelne Merkmale [JBB15; SIL07] und zeichnen sich durch ihre kurze Laufzeit aus [SIL07]. Sie quantifizieren beispielsweise den Informationsgehalt eines Merkmals in Bezug auf die Zielvariable oder messen die Korrelation zwischen zwei Merkmalen. Die Einschränkung von univariaten Methoden besteht darin, dass diese entweder die Beziehung zwischen den Merkmalen oder die Beziehung zwischen einem einzelnen Merkmal und der Zielvariablen evaluieren. Wird die Beziehung zwischen den Merkmalen evaluiert, kann dies dazu führen, dass *irrelevante Merkmale* in die Merkmalsuntermenge aufgenommen werden. Bei dem Evaluieren der Beziehung zwischen einem einzelnen Merkmal und der Zielvariablen können *redundante Merkmale* in der Merkmalsuntermenge vorhanden sein.

Im Gegensatz dazu berücksichtigen multivariate Methoden nicht nur einzelne Merkmale, sondern auch die Wechselwirkung zwischen Merkmalen sowie die Relevanz eines Merkmals in Bezug auf die Zielvariable [SIL07]. Multivariate Methoden zielen darauf ab, die möglichst optimale Merkmalsuntermenge zu finden, die lediglich *relevante Merkmale* umfasst. Als Relevante Merkmale werden dabei Merkmale bezeichnet, welche einen Einfluss auf die Klassenzugehörigkeit einer Dateninstanz haben und nicht durch andere Merkmale dargestellt werden können [JBB15]. Ein Nachteil von multivariaten Methoden besteht darin, dass diese in der Regel eine höhere Laufzeit als univariate Methoden aufweisen [SIL07]. Ein Beispiel für eine multivariate Methode ist die *Minimum Redundancy, Maximum Relevance*-Methode [PLD05].

Mutual Information [Mur22] gehört zu den univariaten Methoden und quantifiziert den Informationsgehalt eines Merkmals in Bezug auf die Zielvariable. Die Mutual Information kann sowohl für kategorische als auch für numerische Merkmale berechnet werden. Zur Berechnung der Mutual Information eines Merkmals sind drei Schritte erforderlich, welche im Folgenden für kategorische Daten vorgestellt werden [Mur22]:

1. Berechnung der Shannonschen Entropie für die Zielvariable Y [Sha48].
2. Berechnung der Shannonschen Entropie für ein Merkmal. Dies bedeutet, dass für ein Merkmal X_i die bedingte Entropie in Bezug auf die Zielvariable Y berechnet wird.
3. Berechnung der Mutual Information, um den Informationsgehalt des Merkmals X_i zu erhalten. Dazu wird die Differenz zwischen den Ergebnissen aus Schritt 1 und Schritt 2 berechnet.

Wird die Mutual Information zur Auswahl von k Merkmalen verwendet, so werden die k Merkmale mit dem höchsten Mutual Information Wert genommen. Zur Berechnung der Mutual Information für kontinuierliche Variablen kann etwa die Kullback-Leibler Divergenz [KL51] verwendet werden [Mur22].

Die **Feature Importance** wird auf der Basis eines trainierten *Random Forest Klassifikator* [Bre01] berechnet und gehört zu den univariaten Methoden. Der Random Forest Klassifikator erstellt N Entscheidungsbäume, wobei jeder Entscheidungsbaum auf einer zufällig gezogenen Stichprobe mit Zurücklegen trainiert wird.An jedem Knoten eines Baumes wird eine zufällige Untermenge der Merkmale betrachtet. Das Merkmal, das die beste Aufteilung der verbliebenen Daten an diesem Knoten ermöglicht, wird ausgewählt. Um festzustellen, welches Merkmal für die Einteilung am besten geeignet ist, wird die Unreinheit eines Merkmals berechnet. Ein Merkmal weist eine hohe Unreinheit auf, wenn es die Daten schlecht nach ihren Klassen trennt. Die Feature Importance für ein Merkmal wird basierend auf dessen Unreinheitswerten berechnet. Zu diesem Zweck werden die Unreinheitswerte eines Merkmals über alle Bäume hinweg aggregiert. Merkmale, die zu einer geringen Unreinheit führen, weisen einen hohen Feature Importance Wert auf. Anschließen werden die k Merkmale mit der höchsten Feature Importance ausgewählt.

Die **Minimum Redundancy, Maximum Relevance (MRMR)**-Methode [PLD05] zählt zu den multivariaten Methoden. Diese Methode berücksichtigt nicht nur die Beziehung der Merkmale zur Zielvariablen, sondern auch die Abhängigkeiten der Merkmale untereinander. Hierbei müssen zwei zentrale Kriterien erfüllt sein:

1. Bestimmung einer maximal relevanten Teilmenge an Merkmalen in Bezug auf die Zielvariable.
2. Ausschluss von Merkmalen innerhalb dieser Teilmenge, die redundant zueinander sind.

Da für einen Datensatz mit n-Merkmalen 2^n mögliche Merkmalsuntermengen existieren, wäre es bei hochdimensionalen Datensätzen rechnerisch zu aufwendig, sämtliche Merkmalskombinationen zu testen. Zur Lösung dieses Problems wird daher ein inkrementeller Suchalgorithmus verwendet, der schrittweise mögliche Teilmengen von Merkmalen generiert und evaluiert, bis die optimale Teilmenge gefunden wird.

Dimensionsreduktion

Die Zielsetzung von *Dimensionsreduktionsmethoden* besteht in der *Reduktion der Merkmale* [GLH16], wobei neue Merkmale auf Basis vorhandener Merkmale mittels linearer oder nicht linearer Kombinationen erstellt werden [FCW+21]. Diese neuen Merkmale ersetzen dann mehrere der ursprünglichen Merkmale. Der Fokus liegt auf der Generierung aussagekräftigerer Merkmale, dabei ist von entscheidender Bedeutung, dass sich die Merkmale als Kombinationen ausdrücken lassen und in numerischer Form vorliegen [GLH16]. Als Beispiele für lineare Dimensionsreduktion können die *Principal Component Analysis* [Dun89] und die *Linear Discriminative Analysis* [Fis36] angeführt werden. Eine nicht lineare Dimensionsreduktion führt die *Isomap-Methode* [TSL00] durch.

Bei der Anwendung der **Principal Component Analysis (PCA)** [Dun89] auf einen Datensatz wird angenommen, dass die wesentlichen Informationen der Merkmale in ihrer Varianz enthalten sind. Folglich besteht das Ziel bei der Anwendung von PCA darin, möglichst viel der Varianz der Merkmale zu bewahren. Zu diesem Zweck werden die ursprünglichen Merkmale in neue, unkorrelierte Hauptkomponenten transformiert, die entlang der Richtung der maximalen Varianz im Datensatz liegen. Zur Ermittlung der Hauptkomponenten werden die Eigenwerte und Eigenvektoren der Kovarianzmatrix des Datensatzes berechnet, wobei angestrebt wird, etwa 95% der Varianz der Daten zu erklären [GLH16]. Eine Einschränkung von PCA ist, dass keine Klasseninformation berücksichtigt wird, wodurch die Anwendung von PCA nicht zwangsläufig zu einer besseren Trennbarkeit der Klassen führt [Bru23].

Das Ziel bei der Anwendung der **Linear Discriminative Analysis (LDA)** [Fis36] besteht in der Maximierung der Trennbarkeit der Klassen. Dies wird erreicht, indem eine lineare Kombination der ursprünglichen Merkmale berechnet wird, um eine Diskriminante zu finden, welche die Trennbarkeit der Klassen am besten erfasst. Im Falle von LDA kann die Diskriminante als eine Funktion betrachtet werden, die die Merkmale gewichtet und projiziert, um die Klassen möglichst gut zu trennen. Hierzu wird die Intercluster-Distanz maximiert, was bedeutet, dass die Distanz zwischen Dateninstanzen verschiedener Klassen möglichst groß sein sollte. Gleichzeitig wird die Intracluster-Distanz minimiert, diese gibt die Distanz zwischen Dateninstanzen derselben Klasse an. Die Anwendung von LDA führt zu einem schlechten Ergebnis, wenn die diskriminierende Information in der Varianz der Daten enthalten ist, da LDA in solchen Fällen nicht die relevante Struktur der Daten erfasst [Bru23]. Dies liegt daran, dass LDA die Klassenseparierbarkeit mittels Intercluster- und Intracluster-Distanz berechnet, welche sich auf Mittelwert und Streuung der Klassen beziehen.

Die **Isomap**-Methode [TSL00] ist eine nicht linare Dimensionsreduktionstechnik, bei der die relative Struktur der Daten erhalten bleibt. Der Begriff relative Struktur bezeichnet hier die Beziehungen und Anordnungen der Dateninstanzen. Die Methode lässt sich in drei Teilschritte zerlegen:

1. *Erstellung eines Nachbarschaftsgraphen:* Der Nachbarschaftsgrap G wird erstellt, indem für jede Dateninstanz $\mathbf{x_i}$ die benachbarten Dateninstanzen ermittelt werden, beispielsweise mit kNN. Anschließend wird jede Dateninstanz mit ihren k nächsten Nachbarn verbunden.
2. *Berechnung des kürzesten Pfades:* Für alle Paare $d(\mathbf{x_i}, \mathbf{x_j})$ im Graphen G wird der kürzeste Pfad zueinander approximiert. Das Ergebnis dieser Approximation wird in einer Distanzmatrix $\mathbf{Dis_G}$ gespeichert.
3. *Transformation in einen niedrigerdimensionalen Raum:* Auf die Distanzmatrix $\mathbf{Dis_G}$ wird die Methode der Multidimensionalen Skalierung (MDS) [BG07] angewandt, um die Daten in einen niedrigerdimensionalen Raum zu transformieren. Das bedeutet, dass die transformierten Daten anschließend eine geringere Anzahl an Merkmalen aufweisen. Die MDS ermöglicht eine Darstellung der Datenpunkte im niedrigerdimensionalen Raum, die die Abstände zwischen den Dateninstanzen aus dem höherdimensionalen Raum bestmöglich repräsentiert. Zu diesem Zweck wendet MDS eine Hauptkomponentenanalyse auf die Distanzmatrix an.

2.3.4 Datenerweiterung

Ziel der *Datenerweiterung* ist es, die Anzahl und Vielfalt der verfügbaren Daten zu erhöhen. Dies ist insbesondere dann von Vorteil, wenn nur eine geringe Datenmenge zur Verfügung steht oder die Klassenverteilung unausgeglichen ist [FCW+21].

Das Anwenden von **Überabtastungs-Methoden** zielt darauf ab, Dateninstanzen zu replizieren oder zu generieren [FGG+18]. Beim Vorliegen der Datencharakteristiken *Klassenungleichgewicht* oder *Klassenüberschneidung mit Klassenungleichgewicht* werden Überabtastungs-Methoden eingesetzt, um das Klassenungleichgewicht zu verringern, indem neue Dateninstanzen für die Minderheitsklasse generiert werden [SAJ+22]. Durch das Generieren neuer Dateninstanzen wird die Repräsentation der Minderheitsklasse erhöht, wodurch die Verzerrung eines Klassifikators zugunsten der Mehrheitsklasse abgeschwächt werden kann. Gängige Verfahren zur Erstellung zusätzlicher Dateninstanzen sind *Random Oversampling* [FGG+18], *SMOTE* [CBHK02] und *ADASYN* [HBGL08].

Random Oversampling [FGG+18] generiert zusätzliche Dateninstanzen durch die zufällige Replikation von vorhandenen Dateninstanzen der Minderheitsklasse. Diese Methode hat den Nachteil, dass das Erstellen exakter Kopien zu Überanpassung des Klassifikators führen kann [FGG+18]. Dennoch führt das Verwenden von Random Oversampling meist zu guten Ergebnissen und zeichnet sich durch eine geringe Laufzeit aus [BPM04].

Das Überabtastungsverfahren **SMOTE** (Synthetic Minority Oversampling Technique) [CBHK02], dient als Grundlage für nahezu alle Überabtastungsmethoden im Bereich des Klassenungleichgewichts [FGG+18]. Das Schlüsselkonzept von SMOTE besteht darin, neue synthetische Dateninstanzen in einem iterativen Prozess zu generieren. Die Schritte sind wie folgt [FGG+18]:

1. Zufällige Auswahl einer Dateninstanz $\mathbf{x_i}$ der Minderheitsklasse.
2. Ermittlung der k nächsten Nachbarn von $\mathbf{x_i}$.
3. Wähle t der k nächsten Nachbarn aus, um durch Interpolation zwischen diesen t Nachbarn und der Dateninstanz $\mathbf{x_i}$ eine neue Dateninstanz zu generieren.

Das iterative Verfahren wird fortgesetzt, bis die gewünschte Klassenverteilung erreicht ist.
Ein Nachteil von SMOTE ist, dass es bei der Erstellung von neuen Dateninstanzen potenzielle Klassenüberschneidungen nicht berücksichtigt. Das bedeutet, dass das Anwenden von SMOTE nicht garantiert, dass der Klassifikator eine verbesserte Entscheidungsgrenze findet. Angesichts dessen wird nach der *Überabtastung* häufig noch eine *Unterabtastung* durchgeführt, um den Klassifikator bei der Optimierung der Entscheidungsgrenze zu unterstützen [SAJ+22].

ADASYN (Adaptive Synthetic Sampling Approach) [HBGL08] ist eine weitere Überabtastungsmethode, die ähnlich wie SMOTE durch Interpolation neue Dateninstanzen erstellt. Im Gegensatz zu SMOTE legt ADASYN dabei den Fokus auf Dateninstanzen der Minderheitsklasse, welche als schwer zu klassifizieren gelten. Schwer zu klassifizieren Dateninstanzen werden anhand ihrer k benachbarten Dateninstanzen ermittelt, es gilt: Je mehr der k benachbarten Dateninstanzen zur Mehrheitsklasse gehören, desto schwieriger ist die Dateninstanz zu klassifizieren und desto mehr synthetische Dateninstanzen werden in dem Bereich um $\mathbf{x_i}$ generiert. Das Erstellen einer neuen Dateninstanz erfolgt durch Interpolation der schwer zu klassifizierenden Instanz $\mathbf{x_i}$ mit einer benachbarten Instanz der Minderheitsklasse. Durch das Anwenden von ADASYN auf Datensätze werden speziell in den Bereichen synthetische Daten erzeugt, in denen eine Klassenüberschneidung vorhanden und die Minderheitsklasse unterrepräsentiert ist. Dies macht ADASYN besonders effektiv bei Datensätzen, die sowohl ein Klassenungleichgewicht als auch eine Klassenüberschneidung aufweisen [VEP21].

2.4 Ensemble-Lernen

Das Ensemble-Lernen ist eine Methode des maschinellen Lernens, bei der unterschiedliche Klassifikatoren auf einem gemeinsamen Problem trainiert und deren Vorhersage zu einer einzigen Vorhersage kombiniert wird. Das Ziel ist es, eine höhere Klassifikationsleistung zu erlangen als die eines einzelnen Klassifikators. Aus diesem Grund sollen möglichst diverse Klassifikatoren erzeugt und zu einem Ensemble kombiniert werden.

2.4.1 Ensembles

Die einzelnen Klassifikatoren, aus denen ein Ensemble besteht, werden im weiteren Verlauf der Arbeit als *Basis-Klassifikatoren* bezeichnet. In Abbildung 2.4 ist ein Ensemble zu sehen. Ein Ensemble besteht aus k Basis-Klassifikatoren $f_1, f_2, \ldots, f_k$. Liegt allen Basis-Klassifikatoren f_i derselbe Algorithmus zugrunde, so wird von einem homogenen Ensemble gesprochen. Im Gegensatz dazu besteht ein heterogenes Ensemble aus Basis-Klassifikatoren, die auf verschiedenen Algorithmen mit unterschiedlichen Hyperparametern basieren. Zur Erzeugung der Vorhersagen eines Ensembles gibt es zwei mögliche Ansätze: die *Selektionsfunktion* oder die *Fusionsfunktion*, welche als F dargestellt werden. Wird eine Vorhersage $\mathbf{Y}$ des Ensembles mittels einer Selektionsfunktion generiert, so wird jeder Basis-Klassifikator auf einen bestimmten Teil des Datensatzes trainiert. Zum Erzeugen einer Vorhersage y_i für eine Instanz $\mathbf{x_i}$ wird ein Basis-Klassifikator ausgewählt. Wird hingegen eine Fusionsfunktion verwendet, so werden alle Basis-Klassifikatoren auf den gesamten Trainingsdaten trainiert. Zum Klassifizieren einer Dateninstanz $\mathbf{x_i}$ trifft jeder Basis-Klassifikator eine Vorhersage, die anschließend zu einer gemeinsamen Vorhersage kombiniert werden. Eine gängige Fusionsfunktion ist das Mehrheitsvotum. Das bedeutet, dass für eine Instanz $\mathbf{x_i}$ die Klasse,

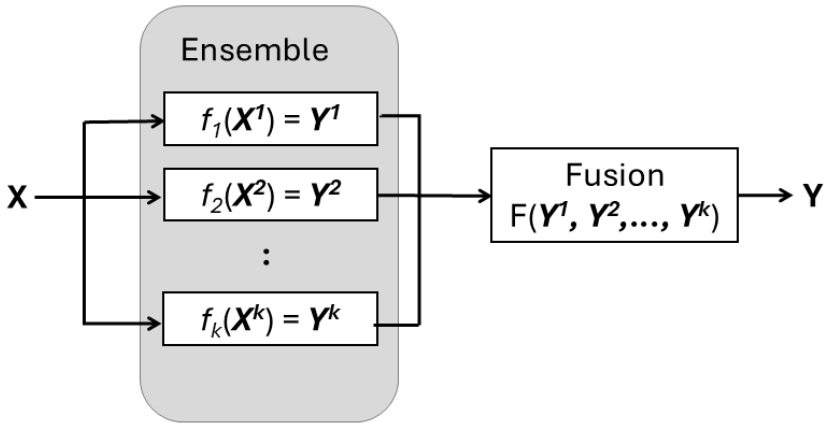

Abbildung 2.4: Ein Ensemble setzt sich aus k Basis-Klassifikatoren zusammen, deren Vorhersagen in einem Fusionsschritt F zu einer einzigen Vorhersage kombiniert werden. Abbildung basierend auf [VRM23].

die von den meisten Basis-Klassifikatoren vorhergesagt wird, die endgültige Vorhersage y_i des Ensembles darstellt [Kun04].

Das Verwenden von Ensembles im Gegensatz zu einzelnen Klassifikatoren bringt verschiedene Vorteile mit sich. Dazu gehört unter anderem, dass Ensembles den Einfluss bestimmter Datencharakteristiken minimieren können:

- *Große Menge an Daten:* Die Nutzung von Ensembles ermöglicht ein effizientes Training auf großen Datenmengen, da es möglich ist, die Basis-Klassifikatoren parallel und auf Teilmengen des Datensatzes zu trainieren [Pol06].
- *Geringe Menge an Daten:* Hier können Ensembles die Vorhersagegenauigkeit verbessern, indem mehrere Modelle auf den wenigen Daten trainiert werden. Beispielsweise können mithilfe von Resampling-Techniken wie Bootstraping unterschiedliche Sichten auf die Daten generiert werden, um eine bessere Klassifikationsleistung zu erreichen [Pol06]. Bootstrap-Sampling [Zho12] wählt für jeden Basis-Klassifikator einen neuen Trainingsdatensatz aus. Dieser wird aus dem gesamten Trainingsdatensatz durch Zufallsziehung mit Zurücklegen erstellt.
  Ein weiterer Vorteil von Ensembles bei dieser Datencharakteristik liegt darin, dass durch das Verwenden von unterschiedlichen Basis-Klassifikatoren die Generalisierbarkeit erhöht wird, da einzelne Klassifikatoren bei einer geringen Datenmenge zur Überanpassung neigen [Die00].
- *Komplexe Entscheidungsgrenzen:* Weisen die Klassen eines Datensatzes komplexe Entscheidungsgrenzen auf, sind einzelne Klassifikatoren häufig nicht in der Lage, die zugrundeliegende Datenstruktur ausreichend zu erfassen. Ensembles wirken diesem Problem entgegen, indem sie unterschiedliche Basis-Klassifikatoren kombinieren, die jeweils verschiedene Aspekte der Datenstruktur erfassen können. Auf diese Weise ermöglichen Ensembles die Generierung komplexerer Entscheidungsgrenzen, die der tatsächlichen Verteilung der Klassen besser entsprechen [Pol06].

2.4.2 Diversität

Ein Ensemble gilt als divers, wenn die Basis-Klassifikatoren auf unterschiedlichen Dateninstanzen fehlerhafte Vorhersagen treffen. Dies trägt zur Minimierung des Gesamtrisikos einer fehlerhaften Vorhersage bei und ist ein Grund für die verbesserte Performanz eines Ensembles gegenüber einem einzelnen Klassifikator [Kun04; Pol06; Zho12]. Zur Erhöhung der Diversität innerhalb eines Ensembles existieren verschiedene Ansätze, die sich in implizite und explizite Diversität unterteilen lassen [Kun04; WGC14].

Implizite Diversität

Das Generieren von impliziter Diversität erfolgt über das Training der Basis-Klassifikatoren, ohne dass explizite Maßnahmen zur Steigerung der Diversität zwischen den Basis-Klassifikatoren ergriffen werden [BWHY05; WGC14]. Dies kann auf unterschiedliche Weise erfolgen, beispielsweise durch Sampling Techniken, welche es den Basis-Klassifikatoren ermöglichen, unterschiedliche Sichtweisen auf die Daten zu entwickeln. Unterschiedliche Sichtweisen bedeuten in diesem Zusammenhang, dass die Basis-Klassifikatoren jeweils andere Aspekte der Daten erkennen, da sie beispielsweise nur auf einer Teilmenge der Merkmale trainiert wurden und dadurch unterschiedliche Fehlentscheidungen treffen. Im Folgenden werden verschiedene Mechanismen zur Generierung von impliziter Diversität erläutert, die sich in zwei Kategorien unterteilen lassen. Zum einen gibt es Verfahren, welche die Darstellung des Datensatzes modifizieren, und zum anderen solche, die Diversität durch die Wahl der Basis-Klassifikatoren erzeugen. Diese Methoden können auch in Kombination verwendet werden [Zho12]:

Datenbasierte Ansätze

- *Verwendung unterschiedlicher Trainingsdaten:* Eine Technik zur Erzeugung von Diversität in Ensembles ist die Verwendung unterschiedlicher Dateninstanzen für das Training der Basis-Klassifikatoren [Pol06; WGC14; Zho12]. Eine Möglichkeit dafür stellt das Bootstrap-Sampling [Zho12] dar. Liegt für ein Klassifikationsproblem nur eine geringe Menge an Daten vor, kann die Diversität des Ensembles durch das Erzeugen von künstlichen Trainingsdaten für die verschiedenen Basis-Klassifikatoren gefördert werden [MM05].
- *Verwendung verschiedener Merkmale:* Die Basis-Klassifikatoren werden auf allen Instanzen des Datensatzes trainiert, verwenden allerdings unterschiedliche Teilmengen der Merkmale [Pol06; WGC14; Zho12]. Dies führt zu unterschiedlichen Sichtweisen auf die Daten. Eine Methode zum Erzeugen von Merkmalsteilmengen ist die Random Subspace Methode [Ho95], welche z. B. vom Random Forest Algorithmus [Bre01] verwendet wird. Eine weitere Möglichkeit besteht darin, für jeden Basis-Klassifikator individuelle Vorverarbeitungsmethoden wie die Merkmalsauswahl oder die Dimensionsreduktion anzuwenden. Durch Variation der Methoden oder ihrer Hyperparameter lassen sich auf diese Weise unterschiedliche Merkmalsteilmengen erzeugen (vgl. Abschnitt 2.3).
- *Diversifizierung der Ausgabe:* Eine Methode zum Erzeugen von diversen Basis-Klassifikatoren besteht in der Manipulation des Klassifikationsproblems für jeden Basis-Klassifikator. Dieser Ansatz verfolgt die Idee, dass die Basis-Klassifikatoren lediglich einen Teil der zu dem Problem gehörigen Klassen kennen, wodurch sich die Basis-Klassifikatoren auf unterschiedliche Klassen spezialisieren. Die ist unter anderem möglich durch die Umwandlung

eines Multiklassen-Klassifikations Problems in ein Binäres-Klassifikations Problem, mittels der Eins-gegen-Alle-Methode [WGC14; Zho12]. Bei der Eins-gegen-Alle-Methode [DKC+03] werden alle Klassen bis auf jene, die gerade evaluiert wird, als "falsch"betrachtet. Dadurch wird jeder Basis-Klassifikator Experte für eine Klasse.

Modellbasierte Ansätze

- *Wahl der Hyperparameter:* Bei dieser Vorgehensweise werden diverse Modelle durch die Wahl der unterschiedlichen Hyperparameter des Basis-Klassifikators erzeugt [Pol06; Zho12]. Die Veränderung der unterschiedlichen Hyperparameter beeinflusst das Training der Basis-Klassifikatoren, wodurch sich deren Vorhersagen unterscheiden können. Beispielsweise können bei einem Multi-Layer Perceptron die Initialisierung der Kantengewichte sowie die Anzahl der Layer und die Anzahl der Knoten variiert werden [Zho12].
- *Heterogene Algorithmen:* Um ein Ensemble aus möglichst diversen Basis-Klassifikatoren zu erstellen, können heterogene Algorithmen verwendet werden. Das Ensemble profitiert davon, dass unterschiedliche Lernalgorithmen zu verschiedenen Modellverzerrungen führen. Eine Modellverzerrung bezieht sich auf die Fehler, die ein Modell produziert, beispielsweise weil es die zugrunde liegenden Muster der Daten nicht vollständig erfasst [Bur19]. Daher ist die Verwendung von unterschiedlichen Algorithmen vorteilhaft, da sie verschiedene Muster in den Daten erkennen [Pol06].
- *Instabile und schwache Lernalgorithmen:* Instabile Lernalgorithmen sind solche, bei denen eine kleine Modifikation der Trainingsdaten zu unterschiedlichen Klassifikatoren führt, die sich in ihren Vorhersagen unterscheiden [ZC01]. Zu dieser Art von Klassifikatoren gehören etwa Entscheidungsbäume. Schwache Lernalgorithmen sind dadurch gekennzeichnet, dass dessen Performanz nur geringfügig besser ist als die des zufälligen Ratens. Durch das Kombinieren mehrerer schwachen Lerner entsteht ein starker Lerner, welcher eine hohe Klassifikationsleistung erreicht [Pol06]. Ein bekanntes Beispiel für ein Ensemble, welches auf schwachen Lernern basiert, ist AdaBoost [FS97]. Die Verwendung instabiler und schwacher Lernalgorithmen führt zur Generierung einer Vielzahl von Entscheidungsgrenzen, welche die Diversität zwischen den Modellen fördern [Pol06].

Explizite Diversität

Die explizite Erzeugung von Diversität erfolgt durch die Optimierung einer Diversitätsmetrik [Kun04]. Dabei wird zwischen *paarweisen* und *nicht-paarweisen* Metriken unterschieden. Zur Berechnung von nicht-paarweisen Diversitätsmetriken werden alle Basis-Klassifikatoren gemeinsam betrachtet. Diese Metriken werden im Folgenden nicht weiter betrachtet, da sie im weiteren Verlauf der Arbeit nicht verwendet werden. Eine weitere Methode zum Berechnen der Diversität sind die paarweisen Metriken. Diese berechnen für jedes Paar von Basis-Klassifikatoren innerhalb eines Ensembles die Diversität. Dies impliziert, dass für k Basis-Klassifikatoren jeweils $\frac{k(k-1)}{2}$ paarweise Diversitätsmetriken berechnet werden. Um die Gesamtdiversität des Ensembles zu erhalten, wird das arithmetische Mittel der paarweisen Diversitätsmetriken gebildet [Bro04; KW03].

Zunächst wird dabei eine Notation eingeführt, die für jedes Paar an Basis-Klassifikatoren (f_i, f_j) den prozentualen Anteil ermittelt, für den ihre Entscheidungen übereinstimmen oder gegensätzlich sind. Dies kann in Form einer Tabelle dargestellt werden, siehe Tabelle 2.1. Dabei gibt (a) an, wie häufig

		f_i	
		Richtig	Falsch
f_j	Richtig	a	b
	Falsch	c	d

Tabelle 2.1: Die Tabelle zeigt den prozentualen Anteil identischer und komplementärer Entscheidungen zweier Klassifikatoren.

beide Basis-Klassifikatoren auf derselben Dateninstanz eine richtige Vorhersage getroffen haben, (d) zeigt an, wie oft beide Basis-Klassifikatoren dieselbe Dateninstanz falsch klassifiziert haben. (b) beschreibt die Anzahl der Fälle, in denen f_i eine richtige Vorhersage auf einer Dateninstanz getroffen hat, während f_j eine falsche Klasse gewählt hat, und (c) entspricht (b) allerdings mit umgekehrten Positionen der Basis-Klassifikatoren.

Kuncheva untersucht in ihrer Arbeit [KW03] die Beziehung zwischen verschiedenen Diversitätmetriken, mit dem Ergebnis, dass viele dieser Metriken miteinander korrelieren. Im Folgenden werden zwei Diversitätmetriken vorgestellt, die keine Korrelation zueinander aufweisen. Diese *Q-Statistik* wird von Kuncheva aufgrund ihrer einfachen Umsetzung sowie ihrer intuitiven Bedeutung empfohlen [Pol06]. Sie misst die Korrelation zwischen zwei Basis-Klassifikatoren f_i und f_j:

$$Q_{i,j} = \frac{ad - bc}{ad + bc} \quad \text{mit} \quad Q \in [-1, 1]$$

Ein positiver Q-Wert bedeutet, dass die Klassifikatoren in ihren Vorhersagen übereinstimmen, was durch hohe Werte für a und d dargestellt wird. Ein negativer Q-Wert deutet darauf hin, dass die Basis-Klassifikatoren die Dateninstanzen gegensätzlich klassifizieren, in dem Fall weisen b und c hohe Werte auf. Daraus folgt, dass q-Werte von -1 und 1 einer geringen Diversität entsprechen. Ein Q-Wert von 0 bedeutet, dass die Basis-Klassifikatoren statistisch unabhängig sind und somit divers.

Die Doppelfehlerrate (DF) zeichnet sich durch ihre Einfachheit aus und misst, wie oft beide Basis-Klassifikatoren denselben Fehler machen. Kuncheva zeigt in ihrer Untersuchung, dass die DF mit keiner anderen der untersuchten Metriken korreliert [KW03].

$$DF_{i,j} = d \quad \text{mit} \quad DF \in [0, 1]$$

Folglich werden zwei Klassifikatoren, die eine hohe DF aufweisen, als wenig divers bezeichnet, da sie einen identischen Fehler produzieren.

Die explizite Diversität lässt sich zur Erstellung eines Ensembles verwenden. Dabei stehen verschiedene Möglichkeiten zur Verfügung:

- Das Erweitern eines Ensembles bis zur Erreichung einer gewünschten Genauigkeit oder Diversität [Kun04].
- Die Integration der Diversität in die Verlustfunktion, die zur Optimierung der Modelle verwendet wird [BWHY05].

- Das Erstellen einer Vielzahl von Modellen sowie die nachfolgende Selektion derjenigen Modelle, welche die höchste Diversität aufweisen [Kun04].

Da die Diversität ein zentraler Aspekt im Erstellungsprozess von Ensembles ist, könnte gefolgert werden, dass eine höhere Diversität zu einer verbesserten Leistung des Ensembles führt. Ein formaler Beweis für diese Annahme existiert jedoch nicht, sodass kein eindeutiger Zusammenhang zwischen Diversität und Genauigkeit nachgewiesen werden kann [KW03]. Es wird angenommen, dass implizite Diversität die Performanz von Ensembles stärker verbessert als explizite Diversität [Kun04].

2.5 AutoML

Das Anwenden von maschinellem Lernen auf Datensätze ist ein aufwendiger Prozess und benötigt Expertenwissen [ZH21]. Zunächst müssen die Daten angemessen vorverarbeitet werden, beispielsweise durch Datenbereinigung, -reduktion und -erweiterung. Nach Durchführung dieser Schritte muss ein geeigneter Klassifikator ausgewählt und die Hyperparameter bestimmt werden, um die Leistungsfähigkeit des Klassifikators zu optimieren. Eine Möglichkeit, diese Herausforderung zu adressieren, bietet das Automatisierte Maschinelle Lernen (AutoML). AutoML automatisiert mit einem begrenzten Rechenbudget die gesamte Prozesskette von der Rohdatenaufbereitung bis zur Erstellung von Modellen, welche zur Vorhersage von neuen Dateninstanzen verwendet werden können [HZC21; ZH21].

Die klassische AutoML-Pipeline lässt sich in verschiedene Schritte unterteilen, hier wird die folgende Darstellung gewählt [ZH21]:

1. Datensammlung
2. Datenvorverarbeitung (inklusive Feature Engineering)
3. Modellgenerierung und -evaluation

Die Datenvorverarbeitung umfasst unter anderem die Datensäuberung sowie -erweiterung. In dieser Arbeit ist das Feature Engineering ein Teil der Datenvorverarbeitung und dient zum Generieren und Auswählen von Merkmalen. Der letzte Schritt, die Modellgenerierung stellt den Kern von AutoML dar und umfasst die Auswahl eines Klassifikator, dessen Hyperparameteroptimierung sowie die Evaluation des trainierten Klassifikator [HZC21; ZH21].

Um in AutoML-Anwendungen die optimale Pipeline zu finden, wird das sogenannte Combined Algorithm Selection and Hyperparameter Optimization (CASH)-Problem [LBBH98] gelöst. Dabei wird ein Suchraum über die unterschiedlichen Klassifikatoren und ihre Hyperparameterkombination erstellt und durch ein Optimierungsverfahren abgetastet, um so die bestmöglichen Klassifikatoren zu erhalten. Neben der Generierung und Evaluierung der verschiedenen Klassifikatoren kann der Suchraum um Datenvorverarbeitungsmethoden erweitert werden, sodass diese mit optimiert werden. Für gewöhnlich weist ein AutoML-Framework statische und dynamische Vorverarbeitungsmethoden auf. Statische Vorverarbeitungsmethoden sind solche, deren Konfigurationen festgelegt und nicht mit optimiert werden [ZH21], wie die Datenkodierung (vgl. Abschnitt 2.3.2). Die dynamischen Vorverarbeitungsmethoden werden zusammen mit den Modellen optimiert. Eine schematische Darstellung einer AutoML-Pipeline, welche die Datenvorverarbeitung mit optimiert, ist in Abbildung 2.5 enthalten. Bei der Erstellung des Suchraums ist zu beachten, dass mit wachsender

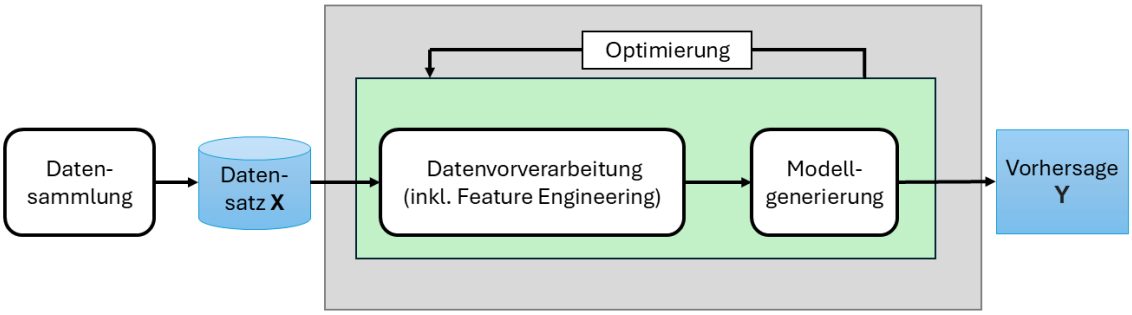

Abbildung 2.5: Darstellung einer AutoML-Pipeline, welche die Datenvorverarbeitungsmethoden mit optimiert.

Anzahl an möglichen Klassifikatoren und Vorverarbeitungsmethoden sowie den dazugehörigen Hyperparametern die Komplexität des Suchraums zunimmt. Dies führt dazu, dass die Lösung des Optimierungsproblems mehr Zeit in Anspruch nimmt, da die Anzahl der möglichen Konfigurationen steigt.

Neben den klassischen Schritten der AutoML-Pipeline kann diese auch weitere Schritte wie die Erstellung eines Ensembles umfassen. Dies ist beispielsweise bei der AutoML-Anwendung von Voggesberger et al. [VRM23] der Fall. Die Erstellung eines Ensembles erfolgt nach der Modellbildung und beinhaltet die Selektion sowie Fusion der ausgewählten Modelle zu einem Ensemble.

3 Verwandte Arbeiten

Dieses Kapitel befasst sich mit Publikationen, die mit dem Thema dieser Masterarbeit in Zusammenhang stehen. Im Rahmen der Literaturrecherche konnten keine wissenschaftlichen Beiträge gefunden werden, die sich explizit mit dem Einfluss von Vorverarbeitungsmethoden auf die Diversität eines Ensembles beschäftigen. Im Folgenden wird Literatur betrachtet, welche sich mit impliziter Diversität, Ensembles und Datenvorverarbeitung beschäftigt. In Abschnitt 3.1 werden Arbeiten vorgestellt, die die Erzeugung impliziter Diversität in Klassifikationsensembles thematisieren. Der Abschnitt 3.2 befasst sich mit AutoML-Frameworks, welche Datenvorverarbeitung unterstützen und Ensembles für die Klassifikation verwenden. Um einen Einblick zu erhalten, wie sich Datenvorverarbeitungsmethoden auf die Klassifikationsleistung eines Klassifikators auswirken, werden in Abschnitt 3.3 Studien aufgeführt, welche die Performanz von Klassifikatoren vergleichen. Jeder Klassifikator wird zu diesem Zweck einmal auf den originalen Daten und einmal auf vorverarbeiteten Daten trainiert. In jeder Studie wird mindestens ein Ensemble als Klassifikationsalgorithmus verwendet. So kann untersucht werden, ob sich Ensembles unter Anwendung von Vorverarbeitungsmethoden anders verhalten als Klassifikatoren, denen kein Ensemble zugrunde liegt. Abschließend werden in Abschnitt 3.4 die Ergebnisse aus diesem Kapitel unter Einbezug des vorhergehenden Kapitel 2 zusammengefasst.

3.1 Implizite Diversität in Klassifikationsensembles

In der Forschung zum Ensemble-Lernen spielt die Diversität der Modelle innerhalb eines Ensembles eine zentrale Rolle. Es gibt verschiedene Techniken, welche die implizite Diversität fördern (vgl. Abschnitt 2.4.2). In diesem Kapitel werden drei Ansätze vorgestellt: Bagging, die Random Subspace Methode und Boosting. Diese Verfahren nutzen unterschiedliche implizite Diversitätsmethoden, um diverse Basis-Klassifikatoren zu erstellen. Ziel ist es die Vorhersagegenauigkeit des Ensembles zu verbessern.

3.1.1 Verwendung unterschiedlicher Trainingdaten

Der von Breiman entwickelte Algorithmus **„Bagging"** [Bre96] ist ein Ensemble-basierter Algorithmus, der Bootstrapping (vgl. Abschnitt 2.4) verwendet, um die Trainingsdatensätze für die verschiedenen Basis-Klassifikatoren zu erstellen. Diese Trainingsdatensätze werden unabhängig von der bisherigen Performanz der Basis-Klassifikatoren auf dem Datensatz erzeugt. Zu, klassifizieren von neuen Dateninstanzen mit dem Ensemble, werden die Vorhersagen der Basis-Klassifikatoren durch ein Mehrheitsvotums zu einer einzigen Vorhersage fusioniert [Kun04]. Darüber hinaus werden instabile Klassifikatoren verwendet, um die Diversität im Ensemble zu erhöhen. Bagging verwendet

datenbasierte und modellbasierte Techniken zur Erzeugung von impliziter Diversität innerhalb des Ensembles, was zur Entstehung von diversen Ensembles mit einer guten Performanz führt [Pol06].

Random Forest [Bre01] bedient sich mehrerer Methoden zur Generierung impliziter Diversität, darunter Bagging und die Verwendung instabiler Klassifikatoren. Neben den bereits genannten Vorteilen von Bagging zeichnet sich Random Forest durch eine hohe Effizienz bei der Verarbeitung großer Datenmengen aus [Pol06; SR18].

3.1.2 Verwendung verschiedener Merkmale

Die **Random Subspace Methode** [Ho95] ist auf Ho zurückzuführen. Bei dieser Methode soll die Korrelation der Basis-Klassifikatoren reduziert werden, indem diese auf zufällig gewählten Merkmals-Untermengen trainiert werden. Dadurch, dass jeder Basis-Klassifikator auf einem modifizierten Trainingsdatensatz trainiert wird, entsteht implizite Diversität (vgl. Abschnitt 2.4.2). Diese Methode ist einfach zu implementieren und effizient für Datensätze mit vielen Merkmalen [SR18].

Der Klassifikator **Extremly Randomized Trees** [GEW06] dient der Generierung eines Ensembles aus Entscheidungsbäumen. Im Rahmen dessen wird die Random Subspace Methode angewendet, um die Trainingsdaten für die verschiedenen Bäume zu generieren. Die Anwendung von Extremly Randomized Trees ist besonders für die Datencharakteristik Hochdimensionale Daten geeignet [GEW06; SR18].

3.1.3 Verwendung Instabiler und schwacher Lernalgorithmen

Boosting [Sch90] ist eine sequenzielle Ensemble-Methode, die erstmals 1990 von Schapire vorgestellt wird. Schapire kann nachweisen, dass es möglich ist, schwache Lerner zu einem starken Klassifikator zu fusionieren. Dieser starke Klassifikator klassifiziert alle Dateninstanzen bis auf eine kleine Restanzahl richtig. Das Erstellen des Ensembles ist ein iterativer Prozess, bei dem die Eingabedaten zunächst einem schwachen Lerner zur Vorhersage übergeben werden. Die Vorhersagen dieses schwachen Lerners werden anschließend zur Berechnung des nächsten schwachen Lerners herangezogen. Dabei erhalten die falsch klassifizierten Dateninstanzen ein höheres Gewicht als die richtig klassifizierten Dateninstanzen. Auf diese Weise wird der Fokus auf die falsch klassifizierten Instanzen gelegt, wodurch das Erstellen eines Basis-Klassifikators gefördert wird, welcher komplementär zu den bisherigen Basis-Klassifikatoren ist. Der Prozess wird wiederholt, bis eine vordefinierte Anzahl von schwachen Lernern erzeugt wurde [Pol06; SR18]. Liegt kein Klassifikationsrauschen innerhalb der Trainingsdaten vor, zeigt sich Boosting als überlegen gegenüber Bagging [FF12]. Allerdings hat Boosting den Nachteil, dass es nicht für verrauschte Daten geeignet ist, da es sonst anfällig für Überanpassung ist [FGE14]. Zudem ist Boosting aufgrund des sequenziellen Trainings rechenintensiv [MS22].

AdaBoost [FS97] gilt als der bekannteste Boosting-basierte Algorithmus. Als schwache Lerner werden häufig Decision Stumps verwendet. Dies sind Entscheidungsbäume, welche aus einem einzigen Knoten bestehen. Zur Klassifizierung von neuen Instanzen wird ein gewichtetes Mehrheitsvotum verwendet, sodass Basis-Klassifikatoren, welche während des Trainingsprozesses eine

gute Leistung zeigen, ein höheres Gewicht erhalten. In den meisten Fällen kommt es in den ersten Iterationen des Trainings zu einem relativ schnellen und kontinuierlichen Rückgang des Fehlers, der mit der Anzahl der hinzugefügten Basis-Klassifikatoren korreliert [SR18].

Eine weitere Ensemble-Methode, die aus instabilen Lernalgorithmen besteht und eine abgewandelte Form des Boostings verwendet, wird von Melville und Mooney [MM05] vorgestellt. Ihr Ensemble **DECORATE** (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) besteht aus Entscheidungsbäumen, welche in einem iterativen Prozess erstellt werden. Während beim Boosting der Fokus durch eine höhere Gewichtung auf falsch klassifizierte Dateninstanzen liegt, wird bei DECORATE dieser Fokus durch das Erstellen von künstlichen Trainingsdaten erzeugt, die dem Trainingsdatensatz der aktuellen Iteration hinzugefügt werden. So werden in jeder Iteration des DECORATE-Algorithmus neue Dateninstanzen, basierend auf den ursprünglichen, Dateninstanzen generiert. Diese neu erstellten Instanzen erhalten Klassenlabels, welche von der aktuellen Vorhersage des Ensembles abweichen, um die Diversität des Ensembles zu maximieren. Im Vergleich zu anderen Ensemble Methoden erreicht der DECORATE-Algorithmus eine höhere Genauigkeit, wenn nur eine geringe Menge an Trainingsdaten zur Verfügung steht.

3.2 Integrierte Vorverarbeitungsmethoden in AutoML-Frameworks

Es existieren verschiedene Frameworks, welche die Anwendung von AutoML unterstützen. Diese unterscheiden sich in ihren Pipelines und Optimierungsmethoden. Im Folgenden werden drei ausgewählte Frameworks betrachtet, welche Ensembles verwenden und die Möglichkeit anbieten, Datenvorverarbeitungsmethoden anzuwenden. Die hier betrachteten Frameworks sind Auto-Sklearn, H20 und AutoGluon.

Auto-Sklearn [FKE+15] ist ein in Python geschriebenes Framework, das zur Integration der Algorithmen die Bibliothek Scikit-learn verwendet. Das Framework integriert unter anderem Ensembles, wobei die einzelnen Modelle während der Modelloptimierungsphase evaluiert und anschließend mittels Auswahlverfahren ausgewählt und durch eine gewichtete Fusionsmethode kombiniert werden. Auto-Sklearn ermöglicht die Anwendung von vier Vorverarbeitungsmethoden und vierzehn Featurevorverarbeitungsmethoden. Die vier Datenvorverarbeitungsmethoden (One-hot-encoding, Imputation, Balancing und Rescaling) werden stets angewandt, sofern dies notwendig ist, und verändern die Merkmalswerte. Im Gegensatz dazu wird von den Featurevorverarbeitungsmethoden nur eine oder keine angewandt. Die Featurevorverarbeitungsmethoden werden gemeinsam mit den Modellen in der Modelloptimierungsphase optimiert. Diese Methoden verändern die Menge der Merkmale. Es wird angenommen, dass die Diversität der Ensembles durch die Verwendung verschiedener Modelle und Hyperparameterkonfigurationen gewährleistet wird.

Das **H20** [LP20] Framework ist ein weiteres AutoML-Framework, das Ensemble-Lernen integriert. Hierfür werden die Algorithmen der H20 Plattform genutzt. Diese beinhalten sowohl Algorithmen zur Modellgenerierung als auch zur Datenvorverarbeitung. Zur Auswahl der Modelle wird eine zufallsbasierte Rastersuche verwendet und zur Fusionierung der Modelle kommt eine gestapelte Generalisierung zum Einsatz. Die von H20 unterstützten Datenvorverarbeitungsmethoden werden nicht gemeinsam mit den maschinellen Lernalgorithmen optimiert. Stattdessen erfolgt die Auswahl der Vorverarbeitungsmethoden abhängig von den vorliegenden Daten oder dem gewählten maschinellen Lernalgorithmus. Beispielsweise wird One-hot-encoding angewendet, wenn ein XGBoost

Modell verwendet wird. Bei fehlendem Wert erfolgt eine automatische Imputation derselben. Eine Normalisierung der Daten wird bei Bedarf ebenfalls automatisch durchgeführt. Die Diversität in den Ensembles basiert auf der zufallsbasierten Optimierung, die eine Vielzahl verschiedener Modelle generiert.

AutoGluon [EMS+20] ist eine Python-Bibliothek für automatisiertes maschinelles Lernen, die fortgeschrittene Datenvorverarbeitung, Deep Learning, Repeated Bagging sowie Multilayer-Stack Ensembling integriert. AutoGluon beinhaltet zwei Formen von Vorverarbeitung: einerseits die Model-Agnostische Vorverarbeitung, welche Standardoperationen, wie die Erkennung des Merkmalstyps, durchführt und andererseits die modellspezifische Vorverarbeitung, welche nur auf bestimmte Kopien der Daten angewendet wird, welche zum Trainieren eines spezifischen Modells verwendet werden. Die Diversität des Ensembles in AutoGluon wird durch die Verwendung von vielfältigen Modellen, dem Multilayer-Stacking sowie dem Repeated Bagging erreicht.

3.3 Einfluss von Datenvorverarbeitung auf Ensemble-Performanz

In diesem Abschnitt werden Studien vorgestellt, welche den Einfluss von Datenvorverarbeitungsmethoden auf die Klassifikationsleistung eines Klassifikators untersuchen. Zu diesem Zweck werden die Klassifikatoren einmal auf dem vorverarbeiteten Datensatz trainiert und das andere Mal auf dem originalen Datensatz. In jeder Studie wird mindestens ein Ensemble als Klassifikationsalgorithmus verwendet, um einen Einblick zu erhalten, ob Vorverarbeitungsmethoden Ensembles anders beeinflussen als Einzelklassifikatoren.

In der Arbeit von Misra et al. [MY19] werden Datensätze aus dem medizinischen Bereich analysiert, die sich durch eine geringe Anzahl an Dateninstanzen auszeichnen. Ziel der Arbeit ist es, die Klassifikationsleistung der Klassifikatoren auf den Datensätzen mittels des Vorverarbeitungsschritts Merkmalsauswahl zu verbessern. Zur Merkmalsauswahl wird die Feature Importance verwendet (vgl. Abschnitt 2.3.3). Zur Evaluierung der Auswirkung von Vorverarbeitungsmethoden auf die Leistung der Klassifikatoren wird eine Klassifikation der vorverarbeiteten Daten sowie der Originaldaten durchgeführt. In der Studie wird zur Klassifikation, neben SVM, Logistic Regression und einem Multiy-Layer Perceptorn, der Ensemble-Algorithmus Random Forest verwendet. Die Ergebnisse der Studien zeigen, dass sich die Klassifikationsgenauigkeit durch den Einsatz von Merkmalsauswahl verbessern lässt. Dies gilt sowohl für Ensembles als auch für einzelne Klassifikatoren. Ein Mangel der Studie besteht darin, dass keine Begründung für die Wahl der Methoden gegeben wird.

Die Studie von Mienye et al. [MS21] verwendet ebenfalls Daten aus dem medizinischen Bereich, welche eine geringe Anzahl an Dateninstanzen sowie eine unausgeglichene Klassenverteilung aufweisen. Als Vorverarbeitungsschritte wird zuerst die Methoden SMOTE-ENN, gefolgt von Recursive Feature Selection angewendet. Zu den verwendeten Klassifikatoren zählen die Ensembles Random Forest und XGBoost, sowie vier weitere Klassifikatoren, welche auf keiner Ensemblemethode basieren. Das Ergebnis zeigt, dass sich die Klassifikationsleistung durch den Einsatz von Vorverarbeitungsmethoden verbessern lässt. Insgesamt verbesserte sich aufgrund der Vorverarbeitung die Accuracy von jedem Klassifikator um mindestens 10 %, die höchste Verbesserung erzielte XGBoost, dieser erreichte eine Verbesserung um 13 %. Die Studie zeigt, dass durch die Anwendung von Vorverarbeitungsmethoden den Datencharakteristiken einer geringen

Datenmenge sowie Klassenungleichgewicht entgegengewirkt werden kann. Weiter macht die Studie deutlich, dass der Einfluss von Vorverarbeitungsmethoden auf Ensembles und einzelne Klassifikatoren in diesem Fall gleich ist.

Abusamra et al. [Abu13] untersuchen in ihrer Studie die Auswirkungen von acht Merkmalsauswahlmethoden auf zwei Datensätze aus der Medizin. Beide Datensätze weisen die Datencharakteristik einer geringen Datenmenge bei gleichzeitig hoher Dimensionalität auf. Der erste Datensatz umfasst 74 Dateninstanzen, der zweite Datensatz enthält 98 Instanzen. Beide Datensätze weisen jeweils 12.287 Merkmalen auf. Zur Klassifizierung werden SVM, Random Forest und 3-NN verwendet. Bei der Betrachtung der Ergebnisse wird deutlich, dass eine automatische Verbesserung der Klassifikationsleistung durch die Anwendung von Merkmalsauswahlmethoden nicht gegeben ist. Für den ersten Datensatz kann die Performanz von Random Forest durch die Merkmalsauswahl überwiegend verbessert werden. Für die Klassifikatoren SVM und 3-NN kannn keine Verbesserung durch die Merkmalsauswahl festgestellt werden. Im Gegensatz dazu hat sich die Performanz von SVM auf dem zweiten Datensatz durch die Anwendung der Merkmalsauswahlmethoden in den meisten Fällen verbessert. Die Performanz von Random Forest und 3-NN hingegen hat sich durch die Anwendung einer Merkmalsauswahlmethode nicht wesentlich verändert. Es ist hervorzuheben, dass die Merkmalsauswahlmethode Gini-Index die Leistung von Random Forest für den ersten Datensatz verbessert hat, während dies bei dem zweiten Datensatz nicht der Fall ist. Die Studie zeigt, dass abhängig vom vorliegenden Datensatz, der verwendeten Merkmalsauswahlmethode und dem angewandten Klassifikator der Einfluss der Merkmalsauswahlmethode auf die Performanz des Klassifikators variiert.

Der Beitrag von Reddy et al. [RRL+20], analysiert den Einfluss von Dimensionsreduktionsmethoden auf die Klassifikationsgenauigkeit. Zur Reduktion der Dimensionen wird PCA und LDA verwendet. Zur Klassifizierung wurde das Ensemble Random Forest, sowie die einzelnen Klassifikatoren Entscheidungsbaum, SVM und Naive Bayes verwendet. Insgesamt werden drei Datensätze analysiert. Zwei der Datensätze stammen aus dem medizinischen Bereich und weisen eine Merkmalsanzahl von 20 bzw. 36 Merkmalen auf. Durch die Anwendung der Dimensionsreduktion auf die medizinischen Datensätze kann eine Verschlechterung der Klassifikationsleistung festgestellt werden. Der dritte Datensatz umfasst 3023 Merkmale, hier kann durch das Anwenden von PCA eine Verbesserung der Performanz der Klassifikatoren festgestellt werden. Der Autor kommt zu dem Ergebnis, dass die Anwendung von Dimensionsreduktionstechniken einen negativen Einfluss auf die Klassifikationsgenauigkeit von Klassifikatoren hat, wenn die Anzahl der Merkmale eines Datensatzes zu gering ist.

3.4 Zusammenfassung

Das Ziel dieser Arbeit besteht darin, zu untersuchen, wie sich durch die Anwendung von Vorverarbeitungsmethoden die Diversität und Genauigkeit von Klassifikationsensembles steigern lassen. Daher betrachtet Kapitel 2 zunächst die verschiedenen Phasen der Datenvorverarbeitung (vgl. Abschnitt 2.3), da sich durch das gezielte Anwenden von Datenvorverarbeitungsmethoden die Klassifikationsgenauigkeit von Klassifikatoren potenziell verbessern lässt. Ein weiterer wichtiger Punkt aus den Grundlagen ist das Ensemble-Lernen, hier liegt der Fokus darauf, wie

Diversität in Ensembles erzeugt werden kann (vgl. Abschnitt 2.4). Besonders interessant ist in diesem Zusammenhang die implizite Diversität, die durch datenbasierte oder modellbasierte Ansätze erzeugt werden kann. Bei den datenbasierten Ansätzen kommen häufig Samplingtechniken zum Einsatz, wie Boosting oder Bagging. Das Erstellen und Optimieren von Klassifikatoren ist oft mit erheblichem Aufwand verbunden, dem AutoML-Ansätze entgegenwirken können (vgl. Abschnitt 2.5). Das Konzept von AutoML besteht darin, automatisiert eine Vielzahl an Modellen zu generieren und anschließend entweder ein einzelnes Modell oder ein Ensemble auszuwählen.

In der Datenvorverarbeitung existieren bereits viele Ansätze und Algorithmen, welche die Performanz von Klassifikatoren verbessern sollen. Jedoch liegt bislang keine Forschung dazu vor, wie sich Datenvorverarbeitungsmethoden auf die Klassifikationsleistung und Diversität von Ensembles auswirken. Vor diesem Hintergrund werden Studien analysiert (vgl.Abschnitt 3.3), welche den Einfluss von Datenvorverarbeitungsmethoden auf die Klassifikationsgenauigkeit von einzelnen Klassifikatoren und Ensembles untersuchen. Die wichtigsten Erkenntnisse aus den Studien sind lassen sich wie folgt zusammenfassen:

- Die Anwendung von Datenvorverarbeitungsmethoden führt nicht zwangsläufig zu einer verbesserten Klassifikationsleistung [Abu13; RRL+20].
- Es kann kein wesentlicher Unterschied festgestellt werden, wie Vorverarbeitungsmethoden die Klassifikationsleistung eines einzelnen Klassifikators im Vergleich zur Klassifikationsleistung eines Ensembles beeinflussen [Abu13; MS21; MY19; RRL+20].

Im Bereich des Ensembles-Lernens gibt es bereits etablierte Verfahren, die auf impliziter Diversität beruhen (vgl. Abschnitt 3.1), wobei einige dieser Verfahren Samplingtechniken zur Erzeugung von Diversität nutzen. Allerdings verwendet keines der etablierten Verfahren Vorverarbeitungsmethoden zum Generieren von impliziter Diversität. Darüber hinaus ermöglichen AutoML-Frameworks zwar die Bildung von Ensembles sowie die Anwendung von Datenvorverarbeitungsmethoden, jedoch wird der Einfluss der Vorverarbeitungsmethoden auf die Diversität und Klassifikationsgenauigkeit eines Ensembles bisher nicht untersucht (vgl. Abschnitt 3.2). Daher widmet sich die vorliegende Arbeit der Frage, inwieweit Vorverarbeitungsmethoden dazu beitragen können, die Diversität und damit verbundene Vorhersagegenauigkeit eines Ensembles zu erhöhen

4 Konzept

In diesem Kapitel wird das Konzept der vorliegenden Arbeit erläutert. Hierzu wird zunächst in Abschnitt 4.1 das allgemeine Konzept auf Basis der bisherigen Erkenntnisse aus den verwandten Arbeiten eingeführt. In Abschnitt 4.2 werden die allgemeinen und vergleichbaren Anforderungen für die Auswahl geeigneter Datenvorverarbeitungsmethoden definiert. Anschließend wird in Abschnitt 4.3 das detaillierte Konzept vorgestellt, wobei auf die Auswahl der Vorverarbeitungsmethoden sowie die Reihenfolge ihrer Anwendung auf einen Datensatz eingegangen wird.

4.1 Allgemeines Konzept

Gegenstand dieser Arbeit ist die Untersuchung, wie sich durch den gezielten Einsatz von Datenvorverarbeitungsmethoden in Abhängigkeit von den vorliegenden Datencharakteristiken eines Datensatzes sowohl die Diversität als auch die Klassifikationsgenauigkeit eines Ensembles erhöhen lassen. Basierend auf den in Abschnitt 3.3 diskutierten Arbeiten kann abgeleitet werden, dass die Anwendung von ungeeigneten Vorverarbeitungsmethoden auf einen Datensatz nicht zwingend zu einer Leistungssteigerung in der Klassifikation führt. Daher ist es sinnvoll, vor dem Anwenden von Vorverarbeitungsmethoden zunächst festzustellen, welche komplexen Datencharakteristiken in einem Datensatz vorliegen (vgl. Abschnitt 2.2), um so gezielte Vorverarbeitungsmethoden anzuwenden, welche die Problematik der vorliegenden Datencharakteristiken adressieren. Vor diesem Hintergrund wird in dieser Arbeit der Ansatz gewählt, Vorverarbeitungsmethoden entsprechend den Datencharakteristiken eines Datensatzes auszuwählen und anzuwenden.

Wie die Studie von Abusamra et al. [Abu13] aufzeigt, kann eine bestimmte Datenvorverarbeitungsmethode bei einem Datensatz zu einer Leistungssteigerung führen, während sie die Klassifikationsleistung bei einem anderen Datensatz negativ beeinflusst. Daher werden pro Datencharakteristik 2–3 Vorverarbeitungsmethoden ausgewählt, um zu gewährleisten, dass es für die verschiedenen Datensätze eine geeignete Vorverarbeitungsmethode gibt. Ein weiterer Grund für die Auswahl mehrerer Vorverarbeitungsmethoden besteht darin, implizite Diversität im Ensemble zu generieren. Durch die Anwendung unterschiedlicher Vorverarbeitungsmethoden wird der Datensatz auf verschiedene Weise modifiziert, wodurch verschiedene Variationen und somit Sichtweisen auf die Daten entstehen. Die Strategie zur Erhöhung der Diversität in einem Klassifikationsensemble basiert darauf, für jeden Basis-Klassifikator eine individuelle Datenvorverarbeitungspipeline zu erstellen, siehe Abbildung 4.1. Dadurch wird jeder Basis-Klassifikator auf einer unterschiedlich modifizierten Darstellung des Datensatzes trainiert, um so durch die Anwendung der Vorverarbeitungsmethoden gemäß Abschnitt 2.4.2 eine datenbasierte, implizite Diversität im Ensemble zu erzeugen.

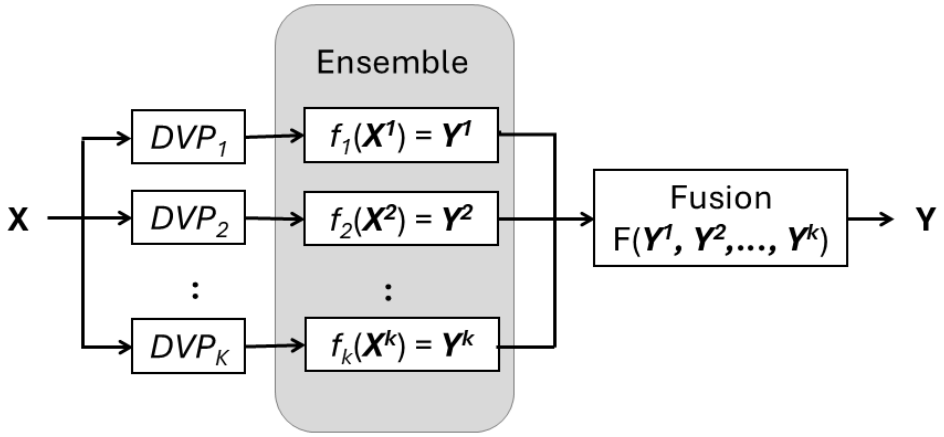

Abbildung 4.1: Die Abbildung zeigt ein Ensemble, welches sich aus k Basis-Klassifikatoren zusammensetzt, wobei jeder Basis-Klassifikator eine eigene Datenvorverarbeitungspipeline (DVP) aufweist.

Das erarbeitete Konzept wird zur Evaluierung prototypisch in einem AutoML-Framework implementiert, welches die Bildung von Ensembles unterstützt. Die Vorverarbeitungsmethoden werden dann gemeinsam mit den Modellen in der Optimierungsphase generiert und evaluiert (vgl. Abschnitt 2.5).

4.2 Anforderungen an Datenvorverarbeitungsmethoden

Basierend auf dem gewählten Ansatz lassen sich allgemeine und vergleichbare Anforderungen an die Auswahl der Datenvorverarbeitungsmethoden für das vorgestellte Konzept ableiten.

Allgemeine Anforderungen

Die hier aufgelisteten Anforderungen sind von allen Vorverarbeitungsmethoden zu erfüllen, damit diese dazu geeignet sind, die Klassifikationsleistung in einem AutoML-Framework zu erhöhen und den Einfluss der vorliegenden Datencharakteristiken abzuschwächen.

G1: Anwendung entsprechend Datencharakteristiken

Die Vorverarbeitungsmethoden müssen auf die vorliegenden Datencharakteristiken des jeweiligen Datensatzes abgestimmt sein, um so die Problematik der Datencharakteristik gezielt abzuschwächen.

G2: Eignung für Klassifikationsprobleme

Die ausgewählten Vorverarbeitungsmethoden müssen sowohl für ein Binäres- als auch ein Multiklassenklassifikationsproblem geeignet sein.

G3: Eignung für numerische und One-Hot-encodierte Daten

Die ausgewählten Vorverarbeitungsmethoden müssen sowohl für numerische Daten als auch für One-hot-encodierte, kategorische Daten (vgl. Abschnitt 2.3.2) anwendbar sein.

Vergleichbare Anforderungen

Basierend auf den vergleichbaren Anforderungen V1 bis V4 lassen sich verschiedene Vorverarbeitungsmethoden miteinander vergleichen und bewerten. Dies führt zur Auswahl der bestmöglichen Methoden, die dafür geeignet sind, sowohl die Diversität als auch die Vorhersagegenauigkeit in einem Ensemble zu erhöhen. Weiter sollten die Methoden für die Anwendung in einem AutoML-Framework geeignet sein. Folgende Anforderungen sind dabei von Relevanz:

V1: Erfassung unterschiedlicher Muster

Die Vorverarbeitungsmethoden müssen sich in ihren Ansätzen zur Datenmodifikation unterscheiden, sodass diese unterschiedliche Muster in den Daten erkennen und diese auf verschiedene Weise modifizieren.

V2: Laufzeit

Es müssen Vorverarbeitungsmethoden gewählt werden, die eine möglichst geringe Laufzeit haben. Es kann vorkommen, das zum einen mehr Vorverarbeitungsmethoden angewandt werden, abhängig von der Anzahl der komplexen Datencharakteristiken, die ein Datensatz aufweist. Zum anderen werden die Datenvorverarbeitungsmethoden in einem AutoML-Framework gemeinsam mit den Modellen optimiert, sodass sich auch hier geringe Laufzeiten der Vorverarbeitungsmethoden positiv auswirken.

V3: Hyperparameter

Datenvorverarbeitungsmethoden mit weniger Hyperparametern sind zu bevorzugen, da diese gemeinsam mit den Modellen in der Modelloptimierungsphase eines AutoML-Frameworks evaluiert werden. Je mehr Hyperparameter eine Vorverarbeitungsmethode aufweist, desto komplexer wird das zu lösende CASH-Problem (vgl. Abschnitt 2.5). Gleichzeitig müssen die Methoden jedoch noch genügend Hyperparameter aufweisen, um das Vorgehen der Methoden zu variieren und somit das Erzeugen impliziter Diversität zu unterstützen.

V4: Bewährtheit

Es sollen etablierte Vorverarbeitungsmethoden gewählt werden, welche als bewährt in ihrem Bereich gelten. Dies kann durch eine weitverbreitete Nutzung der Methode, ihrer Funktion als Baseline-Methode oder die Implementierung in Bibliotheken wie scikit-learn nachgewiesen werden. Sind Methoden in Bibliotheken wie scikit-learn implementiert, zeigt dies, dass die Methode weitverbreitet und anerkannt ist.

4.3 Detailliertes Konzept

Das im Rahmen dieser Arbeit entwickelte Konzept betrachtet Datencharakteristiken wie eine geringe Menge an Daten, Klassenrauschen, Hochdimensionale Daten, sowie Klassenungleichgewicht mit Klassenüberschneidung (vgl. Abschnitt 2.2). Diese komplexen Datencharakteristiken können in verschiedenen Echtwelt-Datensätzen nachgewiesen werden. Dabei zeigte sich, dass diese die Klassifikationsleistung eines Klassifikators negativ beeinflussen [HRM19; Pol06]. Eine Analyse der aktuellen Literatur ergibt, dass bisher nicht untersucht worden ist, wie mit der gezielten Anwendung von Datenvorverarbeitungsmethoden entsprechend den Datencharakteristiken eines Datensatzes die Klassifikationsgenauigkeit sowie Diversität von Ensembles erhöht werden kann (vgl. Kapitel 3). Daher werden drei gut untersuchte Datencharakteristiken gewählt: eine kleine Menge an Daten, Hochdimensionale Daten und Klassenrauschen. Zusätzlich wird die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht ausgewählt, da Klassenüberschneidung als eine der schädlichsten Datencharakteristiken gilt und diese Charakteristik das Finden von Entscheidungsgrenzen erschwert [FGG+18; GMS08]. In Kombination mit Klassenungleichgewicht wird diese Problematik verstärkt [SAJ+22].

Für jede dieser Datencharakteristiken wird, basierend auf den Anforderungen G1-G3 sowie V1-V4, jeweils zwischen zwei bis drei Vorverarbeitungsmethoden ausgewählt. Die Details zur Vorgehensweise der ausgewählten Methoden können Abschnitt 2.3 entnommen werden. Bei der Auswahl der Vorverarbeitungsmethoden wird darauf geachtet, dass die Methoden zum einen die entsprechende Datencharakteristik gezielt adressieren und zum anderen die Problematik der verbliebenen Datencharakteristiken nicht verstärken. Dies gilt insbesondere unter Berücksichtigung des Aspektes, dass ein Datensatz mehrere der komplexen Datencharakteristiken enthalten kann.

4.3.1 Ausgewählte Vorverarbeitungsmethoden

In diesem Abschnitt werden die ausgewählten Vorverarbeitungsmethoden in Abhängigkeit von der vorliegenden Datencharakteristik vorgestellt. Die vergleichbaren Anforderungen V2-V4 für die ausgewählten Methoden sind in Tabelle 4.1 gelistet. V3 bezieht sich hierbei auf die Anzahl der Hyperparameter, die die Methode in der Erstveröffentlichung aufweist.

Klassenrauschen

Zur Reduktion von Klassenrauschen werden zwei ensemblebasierte Rauschfilter ausgewählt: der Ensemble-Filter [BF99] und der Iterative Partitioning Filter (IPF) [KR07] (vgl. Abschnitt 2.3.1). Neben den ensemblebasierten Filtern existieren auch Rauschfilter, die auf Ähnlichkeitsmessungen beruhen, wie die Methoden der DROP-Familie [WM00] oder die Edited-Nearest-Neigbour Methode [Wil72] (vgl. Abschnitt 2.3.1) [MLG+17]. Ein Nachteil dieser Verfahren besteht darin, dass sie zur Identifikation von verrauschten Instanzen ausschließlich kNN ohne Cross-Validierung verwenden. Dies kann problematisch werden, wenn neben dem Klassenrauschen noch ein Klassenungleichgewicht mit Klassenüberschneidung vorliegt. In solchen Fällen besteht die

Methode	V2	V3	V4
Rauschunterdrückungsverfahren			
Ensemble-Filter	$O(n_{CV} \cdot (m^3))$	2 (1/1)	Implementiert in: - Keel - NoiseR-Bibliothek
IPF	$O(i \cdot n_{CV} \cdot (m \cdot n^2 log(n)))$	5 (1/4)	
Merkmalsauswahlverfahren			
Mutual Information	$O(m \cdot n \cdot V_X \cdot V_Y)$	keine	Implementiert in: - scikit-learn
Feature Importance	$O(num_{Trees} \cdot (m' \cdot n'^2 logn'))$	1 (0/1)	
MRMR	$O(\|S\| * \|m\|)$	keine	Grundlage verschiedener anderer Methoden.
Dimensionsreduktionsmethoden			
Kernel PCA	$O(n^3)$	1 (1/0)	Implementiert in: - scikit-learn
LDA	$O(m^2 \cdot n), \quad n > m;$ $O(m^3), \quad m > n$	keine	
Isomap	$O(d \log(k) n \log(n))+$ $O[n^2(k + \log(n))]+$ $O(m_{new} n^2)$	2 (0/2)	
Überabtastungsmethoden			
ROS	$O(n_{synthetisch} + n_{synthetisch})$	?	Erzielt gute Ergebnisse.
SMOTE	$O(n_{minority} \cdot n \cdot m)$	3 (0/3)	Grundlage für viele Überabtastungsmethoden.
ADASYN	$O(n_{minority} \cdot n \cdot m)$	3 (0/3)	Vergleichsmethode für neue Verfahren.
Unterabtastungsmethoden			
Tomek-Link	$O(n^2 \cdot m)$	keine	Standardverfahren nach der Anwendung von SMOTE.
ENN	$O(n^2 \cdot m)$	0 (0/1)	

Tabelle 4.1: Auflistung der vergleichbaren Anforderungen V2-V4 für die ausgewählten Methoden. Die Hyperparameter sind wie folgt angegeben: Gesamt (kategorische/ Numerische). Die Symbole für die Laufzeiten sind wie folgt: n - Anzahl der Dateninstanz eines Datensatzes, $n_{minority}$ - Anzahl der Minderheitsinstanzen, $n_{synthetisch}$ - Anzahl der synthetischen Instanzen, CV - Anzahl der Splits für die Cross-Validierung, m - Anzahl der Merkmale eines Datensatzes, i - Iterationen einer Methode, V_x/V_y - Anzahl der Permutationen für die jeweilige Variable, num_{Tree} - Anzahl der Bäume im Random Forest (RF), m' - Anzahl zu betrachtender Merkmale an jedem Knoten im RF, n' - Bootstrapped Trainingsdaten für RF, $|S|$ - Anzahl der Merkmalsuntermengen, k - Anzahl der Nachbarn, d - Anzahl der Input-Dimensionen, m_{new} - Anzahl der neuen, niedrigeren Dimensionen.

Gefahr, dass viele Dateninstanzen, die der Minderheitsklasse angehören, im Bereich der Klassenüberschneidung entfernt werden können [SLSH15].

Im Gegensatz dazu wirken ensemblebasierte Rauschfilter dieser Problematik entgegen, indem sie zum Finden von verrauschten Instanzen die Vorhersage mehrerer Modelle verwenden, was zu einer verbesserten Generalisierungsleistung führt. Ein weiterer Vorteil dieser Methoden liegt darin, dass sich die Konservativität der Methoden durch das Festlegen des Votingsschemas kontrollieren lässt. Dies ist besonders bei Datensätzen mit weniger Dateninstanzen von Bedeutung, um zu vermeiden, dass zu viele Instanzen aus dem Datensatz entfernt werden. Darüber hinaus sind diese Methoden im Vergleich zu anderen Rauschunterdrückungsverfahren weniger zeitaufwendig und daher für größere Datensätze geeignet [GLH16].

Das Worst-Case-Szenario für die Laufzeit des Ensemble-Filters ist abhängig von der Relation der Anzahl der Merkmale n zu der Anzahl der Dateninstanzen m. Wenn $m > n$ gilt wird die Laufzeit durch LDA dominiert, diese beträgt dann $O(CV \cdot (m^3))$, wobei CV die Anzahl der Splits für die Cross-Validierung (CV) angibt [ASS21]. Gilt $n > m$, so wird die Laufzeit durch den 1-NN Algorithmus dominiert und liegt bei $O(n^2 \cdot m)$ [Ras18]. Der Algorithmus weist insgesamt zwei Hyperparameter auf. Die Laufzeit von IPF liegt bei $O(i \cdot CV \cdot (m \cdot n^2 log(n)))$, wobei i die Anzahl der Iterationen angibt. Eine Besonderheit der IPF-Methode zeigt sich darin, dass diese den Datensatz in jeder Iteration verkleinert und somit effizient auf große Datensätze angewandt werden kann. Dies ist relevant, da in dem Konzept auch hochdimensionale Datensätze betrachtet werden. Der Algorithmus weist insgesamt fünf Hyperparameter auf, davon einen kategorisches und vier numerische. Beide Algorithmen sind in der NoiseR-Bibliothek [MLG+17] und dem Data-Mining-Tool Keel [DGSH15] implementiert.

Hochdimensionale Daten

Um das Trainieren eines Klassifikators auf hochdimensionalen Daten zu erleichtern, ist eine Reduzierung des Datensatzes sinnvoll (vgl. Abschnitt 3.3), dies kann sowohl durch *Merkmalsauswahl* als auch *Dimensionsreduktion* erfolgen.

Bei der Identifizierung von geeigneten Methoden für die Merkmalsauswahl ist darauf zu achten, dass diese sowohl für kategorische als auch numerische Daten geeignet sind, da viele univariate Verfahren entweder für kategorische oder numerische Daten ausgelegt sind. Beispielsweise ist der Fisher-Score [DHG01] und die Correlation-based Feature Selection [WFH+05] ungeeignet für kategorische Daten, während der Chi-Quadrat-Test [Pea92] ungeeignet für numerische Daten ist. Da die univariaten Merkmalsauswahlverfahren *Mutual Information* [Mur22] und *Feature Importance* [Bre01] für beide Datentypen anwendbar sind, werden diese ausgewählt (vgl. Abschnitt 2.3.3).

Die Anwendung von multivariaten Methoden kann zu erheblichen Laufzeiten führen, abhängig von der Vorgehensweise einer multivariaten Methode. Dies gilt beispielsweise für Relief [KR92]. Diese Methode verwendet eine distanzbasierte Messung zum Feststellen von relevanten Merkmalen, eine Vorgehensweise, welche bei hochdimensionalen Daten zu einer erheblichen Laufzeit führt und somit weniger geeignet ist. Im Gegensatz dazu verwendet die multivariate Methode *Minimum*

Redundancy Maximum Relevance (MRMR) [PLD05] einen inkrementellen Suchalgorithmus, welcher eine Annäherung an die optimale Merkmals-Untermenge erlaubt (vgl. Abschnitt 2.3.3) . Zudem existiert für MRMR eine Variante, die speziell für die Anwendung auf hochdimensionalen Datensätzen ausgelegt ist [ZAW19].

Zusammenfassend bieten die ausgewählten Methoden den Vorteil, dass sie den Informationsgehalt der Merkmale ermitteln, ohne eine Annahme über die zugrundeliegende Verteilung zu treffen. Zudem sind sie für kategorische als auch numerische Merkmale geeignet und zeichnen sich durch vergleichsweise kurze Laufzeiten gegenüber anderen Merkmalsauswahlverfahren aus. Die Berechnung der Mutual Information für kategorische Daten weist eine Laufzeit von $O(m \cdot n \cdot V_X \cdot V_Y)$ auf. Hierbei gibt m die Anzahl der Merkmale und n die Anzahl der Dateninstanzen an. Die Variablen V_X und V_Y geben die Anzahl der Permutation für jedes Merkmal sowie für die Zielvariable Y an. Die Mutual Information Methode erfordert keine Angabe von Hyperparametern und ist in der Python-Bibliothek scikit-learn implementiert. Die Bestimmung der Feature Importance basiert auf einem Random Forest Klassifikator [Bre01], dieser dominiert auch die Laufzeit der Methode mit $O(num_{Trees} \cdot (m' \cdot n'^2 logn')$. Dabei gibt num_{Trees} die Anzahl der Bäume in dem Random Forest an, m' definiert die Anzahl der zu betrachtenden Merkmale an jedem Knoten und n' stellt das Trainingssubset dar, welches mittels Bootstrapping erzeugt wurde. Die anzugebenden Hyperparameter entsprechen denen des Random Forest, wobei es sich um einen numerischen Hyperparameter handelt. Auch diese Methode ist in scikit-learn implementiert. Die Variante von MRMR aus der Erstveröffentlichung basiert auf einem inkrementellen Suchalgorithmus, welcher eine Komplexität von $O(|S| * |m|)$ aufweist. Mit $|S|$ als Anzahl der Merkmalsuntermengen. Es werden keine Hyperparameter für die Methode benötigt. MRMR dient als Grundlage verschiedener anderer Methoden wie etwa für die Methoden von Zhao et al. [ZAW19], Thang et al.[TL+10], Ramirez et al. [RLM+17] und Senawi et al. [SWB17].

Eine weitere Vorgehensweise zur Reduzierung der vorhandenen Datenmenge ist die *Dimensionsreduktion*. Hierfür werden die Methoden *Kernel PCA* [SSM97], *LDA* [Fis36] und *Isomap* [TSL00] ausgewählt (vgl. Abschnitt 2.3.3). LDA zählt zu den bekanntesten Dimensionsreduktionsmethoden und erhöht die Separierbarkeit der Klasse. Kernel PCA stellt eine Erweiterung der klassischen PCA [Dun89] dar, bei der unterschiedliche Kernel dazu verwendet werden können, die Dateninstanzen in eine andere Dimensionalität zu übertragen. Dies dient zur effizienten Erfassung nicht-linearer Zusammenhänge. Beispielsweise kann dafür ein polynomialer Kernel oder ein Radial Basis Function (RBF) Kernel verwendet werden [SSM97]. Wird Kernel PCA mit einem linearen Kernel verwendet, ist es äquivalent zur klassischen PCA [VPV+09]. Als dritte Methode wird Isomap gewählt, die eine Erweiterung der Multidimensionalen Skalierung (MDS) [CC00] ist. Im Gegensatz zu MDS bewahrt Isomap die relative Distanz zwischen den Dateninstanzen und erhält somit die globale Struktur des Datenraums [TSL00]. Dies stellt den Hauptvorteil von Isomap gegenüber lokalen linearen Methoden wie Locally Linear Embedding (LLE) [RS00] dar, die lediglich die lokale Strukturen der Daten erfassen. Ein weiter Grund, warum LLE nicht ausgewählt wird, liegt darin, dass es als Spezialfall von Kernel PCA gilt und somit durch Kernel PCA abgedeckt wird [VPV+09].

Die Dimensionsreduktionsmethode Kernel PCA weist eine Laufzeit von $O(m^3)$ auf [ASS21]. Zudem kann ein kategorischer Hyperparameter festgelegt werden. Die Laufzeit von LDA beträgt $O(m^2 \cdot n)$, wenn gilt $n > m$. Gilt hingegen $n < m$, dann beträgt die Laufzeit $O(m^3)$ [ASS21]. LDA erfordert

keine Festlegung von Hyperparametern. Isomap weist eine Laufzeit von $O(d \log(k) n \log(n)) + O[n^2(k + \log(n))] + O(m_{new} n^2)$ auf. Für die Laufzeit gilt, dass k die Anzahl der Nachbarn, d die Anzahl der Inputdimensionen und m_{new} die Anzahl der neuen niedrigeren Dimension festlegt [PVG+11]. Die Methode besitzt zwei Hyperparameter. Alle drei Dimensionsreduktionstechniken sind in der scikit-learn Bibliothek implementiert.

Geringe Menge an Daten

Basierend auf den vorgestellten Arbeiten in Kapitel 3 sind die Durchführung einer *Merkmalsauswahl* oder die *Generierung synthetischer Daten* für Datensätze mit *einer geringen Datenmenge* besonders geeignete Vorverarbeitungsschritte. Das Vorhandensein einer geringen Menge an Daten kann zur Überanpassung des Klassifikators führen [FGG+18]. Daher ist es sinnvoll, den Datensatz entweder durch die Generierung von synthetischen Daten zu erweitern, oder die Anzahl der Merkmale zu reduzieren, da die Menge an benötigten Daten zum Erstellen eines genauen und generalisierten Klassifikators exponentiell mit der Anzahl der Merkmale wächst [Sch20].

Für die Merkmalsauswahl werden dieselben Verfahren verwendet wie bei den hochdimensionalen Daten (vgl. Abschnitt 4.3.1). Zum Generieren von synthetischen Dateninstanzen werden die *Überabtastungs*-Methoden *Random Oversampling (ROS)*, *SMOTE* [CBHK02] und *ADASYN* [HBGL08] ausgewählt (vgl. Abschnitt 2.3.4). ROS wird dabei aufgrund seiner kurzen Laufzeit sowie der guten Ergebnisse in der Anwendung gewählt [BPM04]. Die Laufzeit von ROS beträgt $O(n_{synthetisch} + n_{synthetisch})$, hierbei stellt $n_{synthetisch}$ die Anzahl der zu generierenden synthetischen Dateninstanzen dar. Die Methoden SMOTE und ADASYN werden dagegen in Hinblick auf die Datencharakteristik *Klassenüberschneidung mit Klassenungleichgewicht* ausgewählt. Durch die Generierung zusätzlicher Dateninstanzen soll das Klassenungleichgewicht verringert werden (vgl. Abschnitt 2.3.4).

SMOTE gilt als Grundlage für nahezu alle Überabtastungsmethoden im Bereich des Klassenungleichgewichts [FGG+18]. ADASYN basiert auf SMOTE, wendet jedoch zur Erstellung neuer Dateninstanzen eine verfeinerte Methode an (vgl.Abschnitt 2.3.4). Die Forschungen zu der Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht sind bislang noch nicht abgeschlossen, was durch das anhaltende Interesse und die fortlaufende Veröffentlichung neuer Arbeiten der wissenschaftlichen Gemeinschaft belegt wird [FGG+18]. Viele der neuen Methoden in diesem Bereich verwenden nach wie vor SMOTE und ADASYN als Vergleichsmethode zur Bewertung der Performanz ihrer eigenen Methode. Ein Nachteil der neueren Methoden ist die erhöhte Laufzeit im Vergleich zu SMOTE, da sie die Prozedur von SMOTE erweitern. Zu den Methoden, die auf SMOTE basieren und SMOTE sowie ADASYN als Vergleichsmethode verwenden, zählen unter anderem die Methoden NI-MWMOTE [WHY+20], MWMOTE [BIYM12] sowie CRR [KW17]. Die Laufzeit von SMOTE und ADASYN wird durch das Verwenden der kNN-Methode dominiert und beträgt $O(n_{minority} \cdot n \cdot m)$, dabei gibt $n_{minority}$ die Anzahl der Dateninstanzen der Minderheitsklasse an. Bei beiden Verfahren werden drei numerische Hyperparameter angegeben.

Klassenüberschneidung mit Klassenungleichgewicht

Eine mögliche Vorgehensweise, um dem Problem der *Klassenüberschneidung mit Klassenungleichgewicht* entgegenzuwirken, ist die *Überabtastung* gefolgt von der *Unterabtastung*. Da bereits Überabtastungsmethoden zur Behandlung einer geringen Datenmenge für den Datenvorverarbeitungsprozess ausgewählt wurden (vgl.Abschnitt 4.3.1), wird dieser Ansatz hier weiter verfolgt.

Für die Unterabtastung wurden die Methoden PF [SLSH15], Edited Nearest Neighbors (ENN) [Wil72] und Tomek-Link [Tom76] . Das Anwenden der Methoden ENN und Tomek-Link gilt als Standardverfahren nach der Anwendung von SMOTE, da es die Leistung von SMOTE nahezu immer verbessert [FGG+18]. Die IPF-Methode wird, basierend auf der Arbeit von Saez et al. ausgewählt, der hervorhebt, dass SMOTE-IPF vorzugsweise Dateninstanzen entfernt, die für den Klassifikator schwer zu erlernen sind, wie verrauschte Dateninstanzen und Grenz-Dateninstanzen [SLSH15]. Die Laufzeiten von ENN und Tomek-Link betragen jeweils $O(n^2 \cdot m)$. ENN weist einen numerischen Hyperparameter auf, während die Methode Tomek-Link über keine Hyperparameter verfügt. Die vergleichbaren Anforderungen V2-V4 für die IPF-Methode wurden bereits in Abschnitt 4.3.1 erläutert.

Eine Unterabtastungsmethoden kann unabhängig von der Wahl der Überabtastungsmethode angewandt werden und sollte die Klassifikationsleistung des Klassifikators verbessern. Ursache dieser Verbesserung ist die Entfernung von Dateninstanzen durch die Anwendung einer Unterabtastungsmethode, welche als verrauscht gelten oder das Finden einer Entscheidungsgrenze erschweren [HM13]. Das alleinige Anwenden der Methoden SMOTE und ROS ohne Unterabtastung kann zu einer Überanpassung des Klassifikators beitragen [FGG+18]. Bei einer geringen Datenmenge wird auf die Anwendung einer Unterabtastungsmethode verzichtet, da diese Datensätze bereits zu wenige Dateninstanzen enthalten, um einen generalisierten Klassifikator zu erzeugen.

Ein Überblick über die ausgewählten Methoden entsprechend den jeweiligen Datencharakteristiken kann der Abbildung 4.2 entnommen werden.

4.3.2 Anwendungsreihenfolge der Vorverarbeitungsmethoden

Da insgesamt fünf verschiedene Vorverarbeitungsschritte durchgeführt werden können - Merkmalsauswahl, Dimensionsreduktion, Rauschunterdrückung, sowie Über- und Unterabtastung - ist die Reihenfolge, in der diese Methoden angewendet werden, von entscheidender Bedeutung. Garcia untersucht in seiner Arbeit [GLH16], welche Anwendungsreihenfolge der Vorverarbeitungsmethoden zu dem besten Ergebnis für einen bestimmten Datensatz führt. An diesem Ergebnis orientiert sich die Anwendungsreihenfolge der hier angewandten Datenvorverarbeitungsmethoden, welche Abbildung 4.3 entnommen werden kann. Bei den von Garica untersuchten Methoden fehlen jedoch die Vorverarbeitungsschritte der Dimensionsreduktion sowie der Unterabtastung.

Die Dimensionsreduktion wird in dieser Arbeit als dritter Schritt angewandt, da die Methoden im Vergleich zu den anderen Vorverarbeitungsmethoden eine lange Laufzeit aufweisen (vgl. Tabelle 4.1). Um die Laufzeit zu optimieren, sollten die Methoden daher auf einer möglichst kompakten Darstellung des Datensatzes angewandt werden. Deshalb ist es sinnvoll, die Dimensionsreduktion

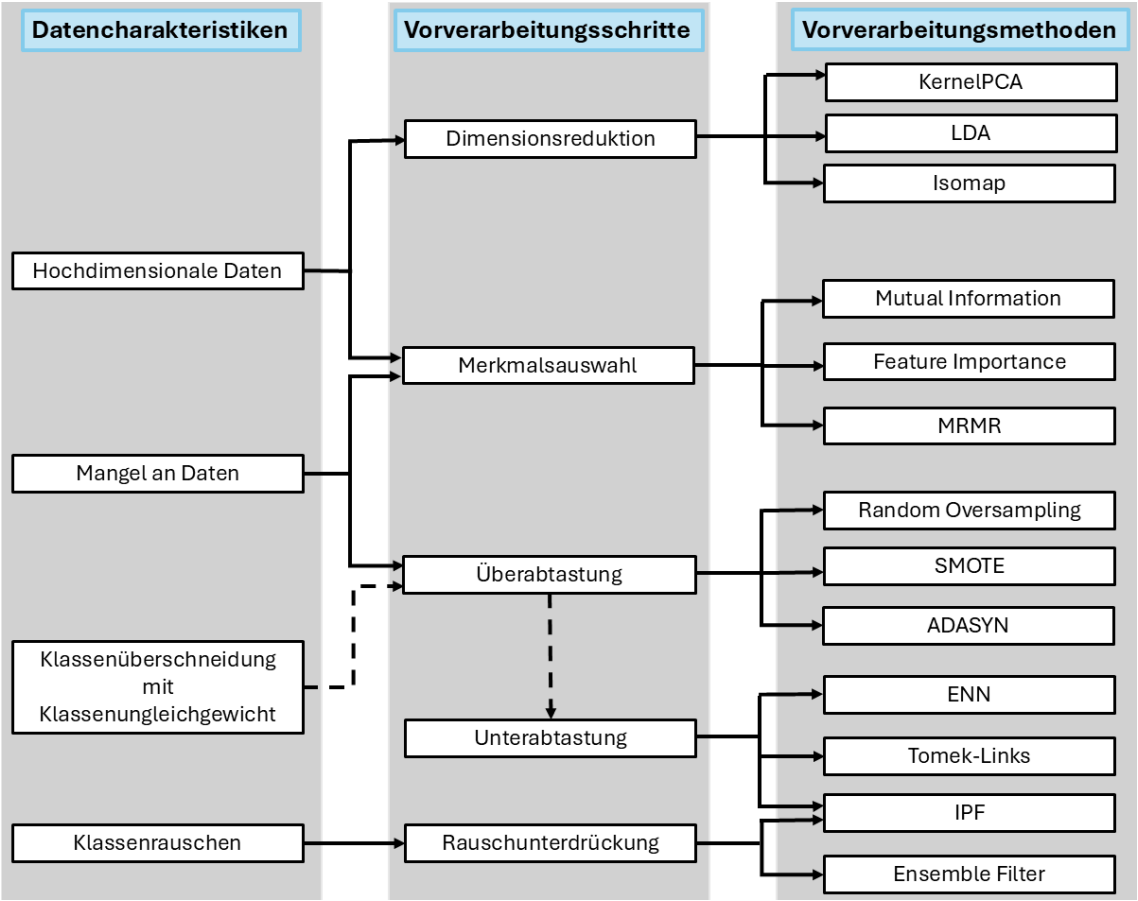

Abbildung 4.2: Die Abbildung bietet eine Übersicht über die ausgewählten Datencharakteristiken, die zugehörigen Vorverarbeitungsschritte sowie die jeweils ausgewählten Vorverarbeitungsmethoden für jeden Schritt der Vorverarbeitung. Die gestrichelte Linie bei der Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht bedeutet, dass lediglich bei Vorliegen dieser Datencharakteristik nach der Überabtastung eine Unterabtastung durchgeführt wird. Dies ist nicht der Fall, wenn ein Mangel an Daten vorliegt.

nach der Merkmalsauswahl anzusetzen, aber noch vor der Generierung neuer Dateninstanzen. Die Überabtastungsmethoden ADASYN und SMOTE verwenden zum Feststellen benachbarter Dateninstanzen den kNN-Algorithmus. Daher profitieren beiden Überabtastungsmethoden von einer reduzierten Darstellung des Datensatzes, da sich dadurch die Laufzeit des kNN-Algrithmus reduziert. Weiter kann sich durch das Anwenden der Dimensionsreduktion die Anordnung der Dateninstanzen im Merkmalsraum verändern [TSL00]. Dies kann zur Folge haben, dass sich der Bereich der Klassenüberscheidung verändert und infolge dessen der Bereich, in dem neue Dateninstanzen generiert werden sollten. Folgerichtig wird die Überabtastung als vierter Schritt in der Vorverarbeutungsreihenfolge angewandt.

Die Unterabtastungsmethoden werden schließlich am Ende der Vorverarbeitungspipeline eingesetzt. Diese Reihenfolge ergibt sich zwangsläufig, da Unterabtastung in Bezug auf Klassenüberschneidung mit Klassenungleichgewicht erst nach der Überabtastung stattfindet. Das Anwenden von Unterabtas-

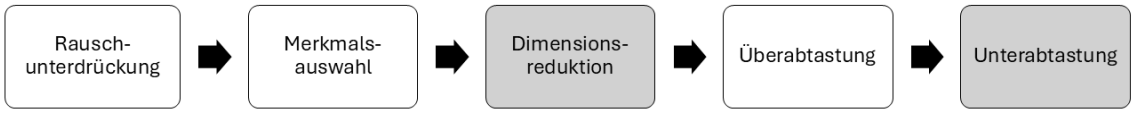

Abbildung 4.3: Die dargestellte Vorverarbeitungspipeline zeigt die Reihenfolge der Anwendung der ausgewählten Vorverarbeitungsschritte, basierend auf der Arbeit Garcías et al. [GLH16]. Die weiß hinterlegten Vorverarbeitungsschritte entsprechen den in Garcías et al. Arbeit behandelten Methoden, während die grau hinterlegten Schritte zusätzliche Erweiterungen darstellen, die in dieser Arbeit ergänzt werden.

tungsmethoden ohne eine vorherige Überabtastung führt dazu, dass die Unterabtastungsmethode gegebenenfalls alle Instanzen der Minderheitsklasse entfernt, da diese fälschlicherweise als Rauschen interpretiert werden [SLSH15].

5 Prototypische Implementierung

In diesem Kapitel wird die prototypische Implementierung des in Kapitel 4 erarbeiteten Konzepts vorgestellt. Diese Implementierung erfolgt mithilfe des automatisierten maschinellen Lernens Frameworks *AutoML-Ensemble*. Zunächst wird in Abschnitt 5.1 die Pipeline sowie die Architektur des Frameworks kurz erläutert, um die anschließenden Modifikationen besser nachvollziehen zu können. Danach wird in Abschnitt 5.2 der Implementierungsentwurf für das Framework vorgestellt. Dieser umfasst die verwendeten Vorverarbeitungsmethoden, Bibliotheken sowie Hyperparameter der Methoden. Abschließend wird in Abschnitt 5.3 auf die Details der prototypischen Implementierung eingegangen. Diese umfassen Erweiterungen sowie Anpassungen des Frameworks. Zum Schluss wird gezeigt, wie das erarbeitete Konzept in dem Framework angewandt wird.

5.1 Verwendetes AutoML-Framework

Um das in Kapitel 4 erarbeitete Konzept umzusetzen, wird das AutoML-Framework *AutoML-Ensemble* von Voggesberg et al. [VRM23] gewählt, das die automatisierte Erstellung und Optimierung eines Ensembles ermöglicht. Das Framework ist in der Programmiersprache Python (Version 3.8) implementiert und verwendet zum Einbinden maschineller Lernalgorithmen die Bibliothek Scikit-learn (1.0.2).

5.1.1 Pipeline

Im Folgenden wird die Pipeline von *AutoML-Ensemble* erläutert, welche insgesamt vier Schritte umfasst. Dabei werden zwei CASH-Optimierungsprobleme gelöst, für die jeweils ein zeitliches Budget gesetzt wird, welches die Dauer der Optimierung begrenzt. Die schematische Pipeline kann Abbildung 5.1 entnommen werden. Sie umfasst die folgenden Schritte:

1.) Datenspeicher

Der Datenspeicher speichert und modifiziert die Daten, damit diese für die nachfolgenden Schritte der *AutoML-Ensemble* Pipeline geeignet sind. Diese enthält zwei Vorverarbeitungsmethoden, welche auf die Datensätze angewendet werden. Zum Einen kommt das Labelencoding zum Einsatz, dies wird auf jeden Datensatz angewandt. Zum anderen wird das One-hot-encoding für Datensätze, die kategorische Merkmale enthalten, verwendet (vgl.Abschnitt 2.3.2).

2.) Klassifikatoroptimierung

Das erste Optimierungsproblem dient dazu, geeignete Klassifikatoren zu generieren. Das Tool verwendet zum Lösen des CASH-Problems Bayesian Optimization of Ensembles [LGS16].

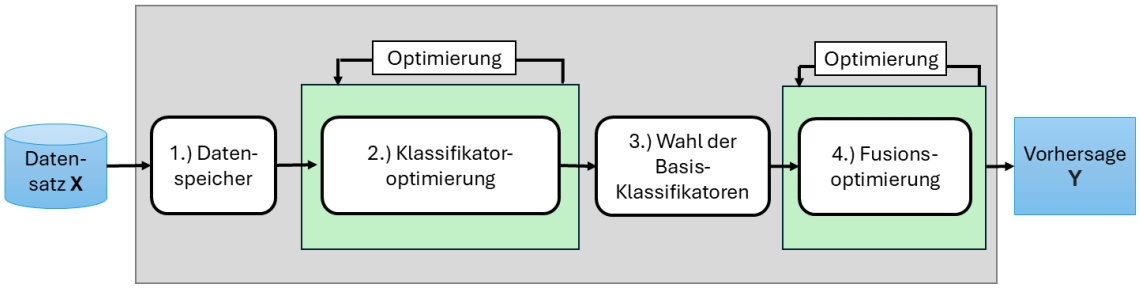

Abbildung 5.1: Schematische Darstellung der implementierten Pipeline in *AutoML-Ensemble*.

3.) Wahl der Basis-Klassifikatoren

Zum Erstellen eines Ensembles wird der Ansatz *Überproduzieren und Auswählen* verwendet, bei dem die Klassifikatoren anhand einer expliziten Diversitätsmetrik ausgewählt werden [VRM23]. Für die Auswahl der Basis-Klassifikatoren wird der Selektionsalgorithmus *Clustering* [GR01] angewandt (vgl. Abschnitt 2.4.2).

4.) Fusionsoptimierung

Ein wichtiger Schritt bei der Verwendung von Ensembles zur Vorhersage eines Klassifikationsproblems ist der Fusionsalgorithmus. Im zweiten CASH-Optimierungsproblem wird der optimale Fusionsalgorithmus ausgewählt.

5.1.2 Paketstruktur des Frameworks

Im Folgenden wird die Paketstruktur des *Auto-Ensemble Frameworks* erläutert, damit die prototypische Implementierung des Entwurfs in Abschnitt 5.3 besser nachvollziehbar ist. Insgesamt enthält das Framework sechs Pakete. Diese werden mit *inout, main, pipeline, optimization, pusion* und *utils* bezeichnet. Pakete, die Klassen enthalten, die im Verlauf der Arbeit angepasst werden, werden kurz erläutert. Darüber hinaus werden die Inhalte der Pakete nicht vollständig erläutert. Der Hauptfokus liegt auf den Klassen, welche verändert werden.

- *inout:*

  Das Paket enthält unter anderem Klassen zum Einlesen der Daten sowie zur Evaluierung und Speicherung der Performanz des Ensembles.

- *main:*

  Dieses Paket umfasst die Klasse *EnsembleLearner* und *DataStore* sowie das Modul *create_cs*. Die Klasse EnsembleLearner stellt die Hauptklasse des Frameworks dar und führt die Klassen der anderen Pakete zusammen. Das Modul *create_cs* enthält Methoden zur Generierung des Suchraums für die zwei zu lösenden Optimierungsprobleme. Die Klasse *DataStore* beinhaltete Methoden zur Transformation der Daten in das erforderliche Format des AutoML-Frameworks.

- *optimization:*

  Dieses Paket beinhaltet Klassen zur Lösung des CASH-Problems.

- *pipeline:*

  Das Pipeline-Paket enthält insgesamt vier Unterpakete. Diese beinhalten etwa die verschiedenen *Klassifikationsalgorithmen*, sowie unterschiedliche *Überproduzieren- und Auswählenalgorithmen*. Für diese Arbeit ist das Unterpaket *transformation* von Interesse, dieses enthält eine abstrakte Transformationsklasse, welche zur Implementierung der Vorverarbeitungsmethoden verwendet wird.

5.1.3 Integration des Konzeptes

Das Konzept aus Kapitel 4 wird in die Klassifikatoroptimierung des AutoML-Frameworks integriert. Dies wird durch die Anpassung der Vorverarbeitungsschritte und Vorverarbeitungsmethoden, welche im Framework enthalten sind, erreicht. Weiter muss der Suchraum der ersten Optimierungsphase angepasst werden, sodass nun Datenvorverarbeitungmethoden gemeinsam mit den Klassifikatoren optimiert werden. Eine schematische Darstellung der erweiterten Pipeline kann Abbildung 5.2 entnommen werden. Da das Framework bisher keine Vorverarbeitungsmethoden enthält, muss die richtige Anwendung der Vorverarbeitungsmethoden auf einem Datensatz gesichert werden. Es werden zwei Arten von Vorverarbeitungsmethoden angewandt. Zum einen gibt es Vorverarbeitungsmethoden wie etwa die Merkmalsauswahl oder die Dimensionsreduktion, welche auf den ganzen Datensatz angewendet werden, da diese die Darstellung der Merkmale verändern. Zum anderen gibt es Samplingmethoden, welche dem Datensatz Dateninstanzen hinzufügen oder aus diesem entfernen. Diese Methoden werden lediglich auf den Trainingsdatensatz eines Klassifikationsproblems angewandt. Vorverarbeitungsschritte, welche in die Kategorie des Samplings fallen, sind Rauschunterdrückunsgverfahren, Überabtastungs- und Unterabtastungsmethoden.

5.2 Implementierungsentwurf des Konzeptes

In diesem Abschnitt werden die integrierten Vorverarbeitungsmethoden erläutert. Da *AutoML-Ensemble* in Python Version 3.8 implementiert ist, wird zunächst geprüft, ob die ausgewählten Vorverarbeitungsmethoden in einer Python-Bibliothek vorhanden sind. Die meisten Methoden stammen aus den Bibliotheken *scikit-learn*[1] und *imbalanced-learn*[2], die beide gut dokumentiert sind.

Die Bibliotheken *Scikit-learn* ist eine bewährte Bibliothek in den Bereichen des maschinellen Lernens und wird bereits im *AutoML-Ensemble* Framework verwendet. Die Bibliothek *imbalance-learn* bietet viel erprobte Vorverarbeitungsmethoden, die unter anderem dem Problem der Klassenüberschneidung mit Klassenungleichgewicht entgegenwirken. Im Nachfolgenden wird kurz erläutert, auf welcher Implementierung eine ausgewählte Vorverarbeitungsmethode basiert. In Tabelle 5.1 wird für die ausgewählten Vorverarbeitungsmethoden die jeweilige Anzahl der Hyperparameter angegeben. Die Hyperparameter spielen eine zentrale Rolle, da sie einerseits den Suchraum des Optimierungsproblems aufspannen. Andererseits lässt sich durch die Wahl der Hyperparameter beeinflussen, wie stark die jeweiligen Vorverarbeitungsmethoden den Datensatz modifizieren oder

[1] https://scikit-learn.org/stable/index.html
[2] https://imbalanced-learn.org/stable/index.html

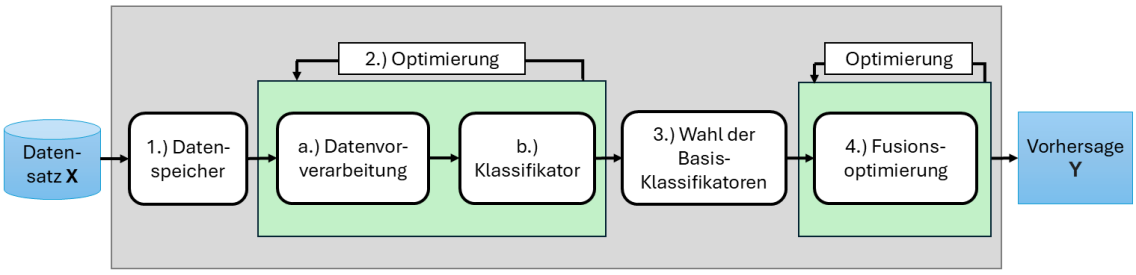

Abbildung 5.2: Schematische Darstellung der implementierten Pipeline in *AutoML-Ensemble* mit Integration des entwickelten Konzeptes. Die Datenvorverarbeitungsmethoden werden gemeinsam mit den Klassifikatoren im 2.) Schritt optimiert.

welche Muster in den Daten erkannt werden. Dies ist besonders wichtig, da das Ziel darin besteht, durch die Vorverarbeitungsmethoden implizite Diversität im Ensemble zu generieren und durch die Wahl der Hyperparameter unterschiedliche Darstellungen der Daten generiert werden können. Werden die Hyperparameter einer Vorverarbeitungsmethode bereits in der Erstveröffentlichung der Methode verwendet, so wird sich an diesen orientiert, um Initialwerte für den Konfigurationsraum des CASH-Optimierungsproblems zusetzten. Andernfalls werden für Hyperparameter, welche in der Erstveröffentlichung der Methode nicht vorkommen, die Standardwerte der Bibliothek übernommen. Eine Abweichung von dieser Strategie erfolgt, wenn die Konfiguration bestimmter Hyperparameter negative Auswirkungen auf andere Datencharakteristiken hat. Die Tabelle 5.1 enthält eine Übersicht der wichtigsten Informationen zur Implementierung der verschiedenen Vorverarbeitungsmethoden.

5.2.1 Rauschunterdrückungs

Für die beiden ausgewählten Rauschunterdrückungsverfahren existieren keine Python-Implementierungen, so dass die Methoden manuell auf Basis des R-Pakets NoiseFilterR[6] implementiert werden. Zwei der Hyperparameter von IPF werden aus der Arbeit über IPF-SMOTE [SLSH15] übernommen. Die Anfangswerte der anderen drei Parameter werden so angepasst, dass diese auch für einen Datensatz, welcher die Datencharakteristiken Klassenungleichgewicht mit Klassenüberschneidung enthält, geeignet sind. So wird der Gefahr begegnet, dass viele Dateninstanzen, die der Minderheitsklasse angehören, im Bereich der Klassenüberschneidung entfernt werden [SLSH15]. Beim Ensemble-Filter werden die Parameter übernommen, die auch in der Erstveröffentlichung definiert werden.

[3] https://rdrr.io/cran/NoiseFiltersR/src/R/EF.R
[4] https://rdrr.io/cran/NoiseFiltersR/src/R/IPF.R
[5] https://pypi.org/project/mrmr-selection/
[6] https://rdrr.io/cran/NoiseFiltersR

Vorverarbeitungs-schritt	Methode	Hyperparameter	Implementiert durch:
Rauschunter-drückung	Ensmeble-Filter	3 (1/2)	basierend auf R-Implementierung [3]
	IPF	5 (1/4)	basierend auf R-Implementierung [4]
Merkmals-auswahlmethoden	Mutual Information	1 (0/1)	scikit-learn
	Feature Importance	1 (0/1)	scikit-learn
	MRMR	2 (1/1)	mrmr-selection [5]
Dimensions-reduktionsmethoden	LDA	5 (2/3)	scikit-learn
	Kernel PCA	5 (1/4)	scikit-learn
	Isomap	2 (0/2)	scikit-learn
Überabtastung	SMOTE	3(1/2)	imbalance-learn
	ADASYN	2 (0/2)	imbalance-learn
	ROS	2 (1/1)	imbalance-learn
Unterabtastung	Tomek-Link	3(1/2)	imbalance-learn
	ENN	3(2/1)	imbalance-learn

Tabelle 5.1: Auflistung der angewandten Vorverarbeitungsschritte und verwendeten Vorverarbeitungsmethoden. Es werden die verwendete Bibliothek sowie die Anzahl der Hyperparameter aufgelistet. Die Hyperparameter sind wie folgt angegeben: Gesamt (kategorische/ numerische)

5.2.2 Merkmalsauswahl

Die Merkmalsauswahlmethoden Feature Importance und Mutual Information werden mittels scikit-learn implementiert. Die Methode MRMR wird durch die Python-Bibliothek *mrmr-selection* integriert, die auf der Arbeit von Zhao [ZAW19] basiert. In dieser Arbeit wurde die Methode überarbeitet, sodass MRMR effizient auf große Datensätze angewendet werden kann.

5.2.3 Dimensionsreduktion

Für die drei gewählten Dimensionsreduktionstechniken stehen scikit-learn Implementierungen zur Verfügung. Die Werte der Hyperparameter von LDA, Kernel PCA und Isomap orientieren sich an den scikit-learn Standardwerten, da die Werte der meisten dieser Hyperparameter in den Erstveröffentlichungen nicht definiert sind.

5.2.4 Überabtastung

Die Überabtastungsmethoden werden mittels der Bibliothek *imbalance-learn* implementiert. Im Gegensatz zu den Algorithmen aus der Erstveröffentlichung bieten die implementierten Methoden für Random Oversampling (ROS) und SMOTE die Möglichkeit, gezielt festzulegen, für welche Klassen neue Dateninstanzen generiert werden sollen.

5.2.5 Unterabtastung

Da IPF sowohl als Rauschunterdrückungsverfahren als auch als Unterabtastungsmethode eingesetzt wird, werden die Details in Tabelle 5.1 nicht erneut für IPF als Unterabtastungsmethoden gelistet. Die Methoden ENN und Tomek-Links werden mittels der Bibliothek *imbalance learn* implementiert.

5.3 Prototypische Implementierung in AutoML-Ensemble

Die Vorverarbeitungsmethoden werden entsprechend dem Framework in der Python-Version 3.8 implementiert. Das Framework wird dabei um die Pakete *imbalance-learn* und *mrmr-selection* erweitert. Im Folgenden werden die durchgeführten Erweiterungen und Anpassungen des Frameworks genauer erläutert. Anschließend wird gezeigt, wie das Framework aufgerufen wird, um die Vorverarbeitungspipeline zu aktivieren.

5.3.1 Erweiterung

Die Struktur der hinzugefügten Vorverarbeitungsmethoden orientiert sich an der generischen Struktur der scikit-learn Vorverarbeitungsmethoden sowie der des Frameworks. Das bedeutet, dass jede hinzugefügte Funktion die Struktur entsprechend der des BasePreprocessor aus Abbildung 5.3 aufweist. Für die sieben integrierten Samplingmethoden wird dem *transformation*-Pakte ein neues Paket *sampling* hinzugefügt. Die restlichen Vorverarbeitungsmethoden sind im Paket *feature_engineering* enthalten.

5.3.2 Anpassung

Damit die Vorverarbeitungsmethoden entsprechend der erarbeiteten Reihenfolge in Abschnitt 4.3.2 angewendet werden können, musste der Suchraum der Klassifikatoroptimierung angepasst werden. Während der ursprüngliche Suchraum lediglich Methoden für das *['Encoding']* enthält, wird dieser jetzt modifiziert zu *['Rauschunterdrückung', 'Merkmalsauswahl', 'Dimensionsreduktion', 'Überabtastung', 'Unterabtastung']*.

Die Methoden zur Merkmalsauswahl und Dimensionsreduktion sind im Paket *feature_engineering* implementiert. Diese Methoden werden auf alle Teilmengen des Datensatzes angewandt, da sie die Anzahl der verfügbaren Merkmale reduzieren. Dies führt dazu, dass der Klassifikator auf

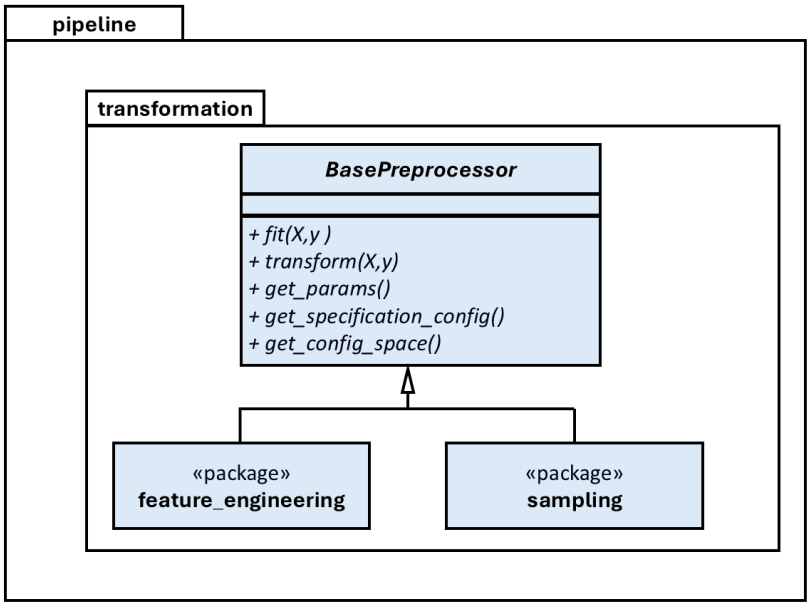

Abbildung 5.3: Das Diagramm zeigt eine vereinfachte Darstellung des Pipeline-Pakets des *AutoML-Ensemble* Frameworks. BasePreprocessor zeigt die generische Klasse, deren Struktur von den Vorverarbeitungsmethoden übernommen wird. Die hinzugefügten Vorverarbeitungsmethoden sind in den Paketen featur_engineering und sampling enthalten.

einer reduzierten Menge an Merkmalen trainiert wird. Folglich dürfen Merkmale, die nicht zum Trainieren des Klassifikators verwendet werden, auch nicht im Validierungs- und Testdatensatz für diesen Klassifikator vorhanden sein.

Durch das Anwenden von Samplingmethoden soll der Klassifikator die Struktur der zugrundeliegenden Daten besser erlernen. Zu diesem Zweck werden Dateninstanzen dem Trainingsdatensatz hinzugefügt oder entfernt. Diese Methoden werden jedoch nicht auf den Validierungs- und Testdatensatz angewandt, da dieser zur Bewertung der Klassifikationsgenauigkeit herangezogen werden und keinen Einfluss auf die Erstellung der Klassifikatoren haben. Da im bisherigen Framework keine Samplingmethoden enthalten waren, wird deren Anwendung auf einen Datensatz angepasst.

Als letzter Punkt wird die Ausgabe des Frameworks erweitert, sodass für die Basis-Klassifikatoren ausgegeben wird, welche Vorverarbeitungsmethoden angewandt wurden.

5.3.3 Anwendung

In Listing 5.1 ist ein beispielhafter Aufruf des *AutoML-Ensemble* Frameworks zu sehen. In *Zeile 1* wird das Framework importiert. Um *AutoML-Ensemble* auf einen Datensatz anzuwenden, muss zunächst ein Datensatz geladen werden, wie in den *Zeilen 2–6* gezeigt wird. Anschließend wird in den *Zeilen 8–11*, die Datencharakteristiken des vorliegenden Datensatzes angegeben. In *Zeile 13* wird die Ensemble-Klasse initialisiert. Für die Initialisierung müssen die Variablen `ens_size`, `budget_m` und `budget_f` angegeben werden. Die Variable `ens_size` definiert die gewünschte Größe des Ensembles, die Variablen `budget_m` und `budget_f` geben das zeitliche Budget für die Klassifikatoroptimierung

Listing 5.1 Anwendungsbeispiel des Frameworks AutoMLEnsemble Frameworks mit angewandter Datenvorverarbeitung.

```
1  import automlopen as ao
2  from openml import tasks, datasets
3
4  dataset = datasets.get_dataset(41, download_data=True, download_qualities=False,
5                                 download_features_meta_data=False)
6  X, y, _, _ = dataset.get_data(dataset.default_target_attribute)
7
8  dc = {  'small_data': True,
9          'class_noise': False,
10         'high_dimensional_data': False,
11         'class_overlap':True}
12
13 el = ao.EnsembleLearner(ens_size=2, budget_m=120, budget_f=120,
14                         preprocessing=True, data_characteristics=dc)
15 el.fit_evaluate(X, y)
```

sowie für die Fusionsoptimierung an. Wird das Ensemble ohne weitere gesetzte Variablen erstellt, so werden keine Vorverarbeitungsmethoden angewendet. Um Vorverarbeitungsmethoden anzuwenden, müssen zwei zusätzliche Variablen gesetzt werden. Zum einen muss `preprocessing` auf `True` sein, zum anderen muss der Variable `data_charactertistics` ein Dictionary übergeben werden, das angibt, welche Datencharakteristiken vorliegen. Dies ist das Dictionary welches in den *Zeilen 8–11* definiert ist. Zuletzt wird die Methode `fit_evaluate(X,y)` aufgerufen, um das Ensemble zu trainieren und zu evaluieren.

In Listing 5.2 kann die Ausgabe des Frameworks für die in **??** gegebene Konfiguration entnommen werden. Diese zeigt, dass entsprechend den Datencharakteristiken die Vorverarbeitungsschritte *Merkmalsauswahl (FD), Überabtastung (OVS)* und *Unterabtastung (US)* angewandt werden. Weiter wird die erreichte accuracy und Diversität des Ensembles sowie der Basis-Klassifikatoren angegeben

Listing 5.2 Ausgabe für das Anwendungsbeispiel Listing 5.1.

```
Evaluation Results
==========================================================================================
Algorithm      Fil          FS                  DR       OVS    US            accuracy
------------------------------------------------------------------------------------------
Combiner
COS                                                                                0.758
------------------------------------------------------------------------------------------
Classifiers
RF             NOFIL        MUTUAL_INFORMATION  NODR     ROS    tomek_links   0.758
RF             NOFIL        MUTUAL_INFORMATION  NODR     ROS    tomek_links   0.303
------------------------------------------------------------------------------------------

Ensemble Diversity
=====================================================
Algorithm Pair      doublefault
-----------------------------------------------------
COS                 0.212
-----------------------------------------------------
RF + RF             0.212
-----------------------------------------------------
```

6 Evaluation

In diesem Kapitel wird das in Kapitel 4 entwickelte und in Kapitel 5 umgesetzte Konzept zur gezielten Anwendung von Datenverarbeitungsmethoden zur Erhöhung der Diversität und Genauigkeit in einem Klassifikationsensemble evaluiert. Zunächst wird in Abschnitt 6.1 der Versuchsaufbau beschrieben, einschließlich der verwendeten Datensätze, der definierten Evaluationsparameter für das Framework, sowie der eingesetzten Hardware und Software. Im Darauffolgenden Abschnitt 6.2 werden die Baselinemethoden eingeführt, diese werden zur Bewertung des Konzepts herangezogen. Anschließend erfolgt in Abschnitt 6.3 die Evaluierung des Konzepts anhand von Performanzmetriken. Hierbei wird zunächst die Leistung des Ensembles gegenüber dem besten Basis-Klassifikator innerhalb des Ensembles verglichen. Im nächsten Schritt folgt die Gegenüberstellung zu zwei weiteren Baselinemethoden. In Abschnitt 6.4 wird der Einfluss des Konzepts auf die Diversität des Ensembles untersucht, wobei erneut ein Vergleich zu den Baselinemethoden durchgeführt wird. Danach folgt in Abschnitt 6.5.2 eine Analyse über den Einfluss der Datencharakteristiken auf die Klassifikationsgenauigkeit sowie eine Betrachtung über die Auswahl der Vorverarbeitungsmethoden. Im Letzten Abschnitt 6.6 werden die wichtigsten Erkenntnisse der Evaluation zusammengefasst.

6.1 Versuchsaufbau

In diesem Abschnitt wird der verwendete Versuchsaufbau für die Evaluation beschrieben. Zunächst werden in Abschnitt 6.1.1 die verwendeten Datensätze vorgestellt. Darauf folgt in Abschnitt 6.1.2 die Beschreibung der gesetzten Evaluationsparameter für das AutoML-Framework. In Abschnitt 6.1.3 und Abschnitt 6.1.4 werden die eingesetzte Software sowie Hardware beschrieben.

6.1.1 Datensätze

Zur Evaluierung des Konzepts werden insgesamt 23 Datensätze verwendet, die jeweils zwischen einer und vier der folgenden Datencharakteristiken aufweisen: *Geringe Datenmenge (GD)*, *Hochdimensionale Daten (HD)*, *Klassenrauschen (KA)* und *Klassenüberschneidung mit Klassenungleichgewicht (KK)*. Es werden alle möglichen Kombinationen an Datencharakteristiken evaluiert. Dazu werden die Datensätze in vier Gruppen eingeteilt, abhängig von der Anzahl der enthaltenen Datencharakteristiken des Datensatzes. Die Einteilung kann Tabelle 6.1 entnommen werden, die ID bezeichnet hierbei einen bestimmten Datensatz.

Tabelle 6.2 listet die verwendeten Datensätze, die dazugehörigen Datencharakteristiken sowie die Anzahl der Dateninstanzen, Merkmale und Klassen auf. Die Datensätze Synth1-Synth8 (IDs: 104, 203, 206, 208, 302, 305, 306, 402) sind synthetische Datensätze, die mit der Funktion `make_classification()` der Python Bibliothek *scikit-learn* generiert wurden. Die restlichen

Gruppe	IDs	Anzahl Datencharakteristiken
1	101-105	1
2	201-209	2
3	301-307	3
4	401-402	4

Tabelle 6.1: Die Tabelle gibt eine Übersicht über die verschiedenen Gruppen, die jeweils enthaltenen IDs in einer Gruppe, sowie die Anzahl der Datencharakteristiken, die in einer Gruppe betrachtet werden.

Datensätze stammen von den Webseiten OpenML[1] sowie UCI[2] und sind Echtwelt-Datensätze. Bei der Auswahl von Datensätzen mit der Datencharakteristik eine geringe Menge an Daten wird darauf geachtet, dass diese weniger als 1000 Dateninstanzen aufweisen.

Das Kriterium bei den Hochdimensionalen Datensätzen ist das Verhältnis der Anzahl von Merkmalen zur Anzahl an Dateninstanzen. Beispielsweise können Hochdimensionale Datensätze viele Dateninstanzen (> 60 000) und viele Merkmale (>350) oder mehr Merkmale als Dateninstanzen aufweisen.

Das Wählen von Echtwelt-Datensätzen, die Klassenüberschneidung mit Klassenungleichgewicht aufweisen, wird für die Datensätze *page-block* und *glass* basierend auf Arbeiten vorgenommen, die Algorithmen zum Bewältigen dieser Datencharakteristik evaluieren. Es wird mindestens einer der Datensätze in den Arbeiten [DWX+24; PCA+21; WHY+20] verwendet. Die Suche nach Datensätzen, welche neben der Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht noch die Datencharakteristik Hochdimensionale Daten aufweisen, ist schwierig. Die Datensätze, welche mindestens diese beiden Datencharakteristiken enthalten, sind *har* und *eating*. Diese werden gewählt, da sie in OpenML für den Aufgabentypen Subgroup Discovery eingesetzt werden. Die Wahl begründet sich darin, dass Subgruppen in Datensätzen eine potenzielle Ursache für eine Klassenüberschneidung darstellen [FGG+18]. Es existieren verschiedene Metriken zur Bewertung des Ausmaßes der Klassenüberschneidung mit Klassenungleichgewicht. Allerdings gibt es keine anerkannten Methoden, die in der wissenschaftlichen Praxis als allgemeiner Standard gelten [SAJ+22]. Daher wurde darauf verzichtet, das Ausmaß der Klassenüberschneidung in den ausgewählten Datensätzen zu quantifizieren. Darüber hinaus stehen keine standardisierten Datensätze zur Verfügung, die zur Evaluierung von Methoden in diesem Kontext verwendet werden

[1] https://www.openml.org/
[2] https://archive.ics.uci.edu/
[3] https://www.openml.org/d/6332
[4] https://www.openml.org/d/184
[5] https://www.openml.org/d/40926
[6] https://www.openml.org/d/30
[7] https://archive.ics.uci.edu/dataset/732/darwin
[8] https://www.openml.org/d/41
[9] https://www.openml.org/d/1478
[10] https://www.openml.org/d/1233

ID	Datensatz	Datencharakteristiken				Anzahl		
		GD	HD	KA	KK	Instanzen	Merkmale	Klassen
Gruppe 1								
101	cylinderbands[3]	x				540	40	2
102	kropt[4]			x		2 8056	7	18
103	CIFAR_10_small[5]		x			20 000	3 073	10
104	Synth1				x	8 000	20	4
105	page-block[6]				x	5 473	11	5
Gruppe 2								
201	cylinderbands[3]	x		x		540	40	2
202	DARWIN[7]	x	x			174	451	2
203	Synth2	x			x	800	20	4
204	glass[8]	x			x	214	10	6
205	CIFAR_10_smalls[5]		x	x		20 000	3 073	10
206	Synth3		x		x	60 000	350	4
207	har[9]		x		x	10299	562	6
208	Synth4			x	x	8 000	20	4
209	page-block[6]			x	x	5 473	11	5
Gruppe 3								
301	DARWIN[7]	x	x	x		174	451	2
302	Synth5	x	x		x	800	350	4
303	eating[10]	x	x		x	945	6 374	7
304	glass[8]	x		x	x	214	10	6
305	Synth6	x		x	x	800	20	4
306	Synth7		x	x	x	60 000	350	4
307	har[9]		x	x	x	1 0299	562	6
Gruppe 4								
401	eating[10]	x	x	x	x	945	6 374	7
402	Synth8	x	x	x	x	800	350	4

Tabelle 6.2: Auflistung der verwendeten Datensätze zur Evaluierung des Konzepts. Für jeden Datensatz wird die ID, der Name, die enthaltenen Datencharakteristiken sowie die Anzahl der Merkmale, Instanzen und Klassen angegeben. Datensätze, welche im Namen *Synth* enthalten, sind synthetische Datensätze. Die restlichen Datensätze sind Echtwelt Datensätze. Die Abkürzungen der Datencharakteristiken sind wie folgt: *Geringe Menge an Daten (GD)*, *Hochdimensionale Daten (HD)*, *Klassenrauschen (KA)* und *Klassenüberschneidung mit Klassenungleichgewicht (KK)*

[SAJ+22]. Aus diesem Grund werden zusätzlich synthetische Datensätze generiert, bei denen sich der Grad der Klassenüberschneidung kontrollieren lässt.

Zum Erstellen der synthetischen Datensätze wird die Funktion `make_classification()` der Bibliothek *scikit-learn* verwendet. Diese Funktion ermöglicht es die Klassenseparierbarkeit, mittels der Variablen `class_sep` zu kontrollieren. Der Wert der Variablen wird auf 0.75 gesetzt, was einen mittleren Grad der Klassenüberschneidung darstellt. Darüber hinaus bietet die Funktion die Option, Klassenrauschen durch die Variable `flip_y` hinzuzufügen. Diese führt ein zufälliges Vertauschen aller Klassenlabels gemäß einem festgelegten Anteil durch.

Zur Evaluierung der Datencharakteristik Klassenrauschen wird den Echtwelt-Datensätze Rauschen hinzugefügt. Diese wird nach demselben Schema wie bei `make_classifiaction()` erzeugt, um eine konsistente Evaluation des Klassenrauschens zu gewährleisten. Datensätze, welche Rauschen enthalten, wird 15% Klassenrauschen hinzugefügt. Dieser Wert orientiert sich an dem Rauschwert, welcher in verschiedenen Arbeiten verwendet wird [DGSH15; ZW04].
Alle Datensätze, bis auf den Datensatz mit der ID 102, sind doppelt enthalten - einmal mit und einmal ohne Rauschen. Dem Datensatz 102 wurde als Datencharakteristik Klassenrauschen hinzugefügt.

Zur Evaluation des Konzepts werden die Datensätze in drei Teile aufgeteilt. Dies ist notwendig zur Ausführung des AutoML-Frameworks. Die Einteilung ist wie folgt:

- 70% Trainingsdaten – werden verwendet zum Training der Klassifikatoren während Klassifikatoroptimierung (vgl. Abschnitt 5.1).
- 15 % Validierungsdaten – dienen zur Evaluierung der trainierten Basis-Klassifikatoren und zum Training des Fusionsalgorithmus.
- 15 % Testdaten – dienen zur Evaluierung der Fusionsalgorithmen.

Die Aufteilung der Datensätze erfolgt durch stratifizierte Zufallsstichproben, sodass die einzelnen Teildatensätze die gleiche Klassenverteilung aufweisen wie der Gesamtdatensatz.

6.1.2 Evaluationsparameter

In diesem Abschnitt werden die notwendigen Parameter für die Evaluierung des Konzeptes vorgestellt, darunter die Auswahl der Ensemblegrößen, die Wahl der Performanz-Metriken sowie die Diversität-Metrik und die verwendeten Zeitbudgets der Optimierungsphase.

Für jeden Datensatz werden fünf verschiedene Ensemblegrößen evaluiert: 2, 5, 10, 15 und 20. Vor der Ausführung des Frameworks müssen zwei Zeitbudgets festgelegt werden: das Zeitbudget für die Modellerstellung, sowie das für die Fusionsoptimierung. Das Zeitbudget für die Modellerstellung legt fest, wie viel Zeit das Framework zur Generierung und Evaluierung der Basis-Klassifikatoren verwendet. Für Datensätze, die weniger als 350 Merkmale oder weniger als 900 Dateninstanzen aufweisen, beträgt das Zeitbudget für die Modellerstellung 40 Minuten. Für Datensätze, die mehr als 350 Merkmale und über 900 Dateninstanzen aufweisen, liegt das Zeitbudget für die Modellerstellung bei zwei Stunden. Die Erhöhung des Zeitbudgets für diese Datensätze ist

notwendig, da die Anwendung der Vorverarbeitungsmethoden auf größere Datensätze einen höheren Zeitaufwand mit sich bringt (vgl. Abschnitt 4.3).

Das Fusionsbudget definiert das Zeitlimit für die Bewertung und Optimierung der verschiedenen Fusionsalgorithmen. Da die Fusionsoptimierung weniger rechenintensiv ist als die Modellerstellung, ist das zeitliche Budget hier geringer. Für die Datensätze, die weniger als 900 Dateninstanzen oder weniger als 350 Merkmale aufweisen, liegt das Fusionsbudget in dieser Arbeit bei 20 Minuten. Bei den Datensätzen mit mehr als 350 Merkmalen und 900 Dateninstanzen beträgt das Fusionsbudget 60 Minuten.

Als Verlustfunktion wird *1-Balanced Accuracy* gewählt. Diese misst den durchschnittlichen Anteil an falsch klassifizierten Instanzen, gewichtet nach den Klassen, und ist insbesondere für unausgewogene Klassenverteilungen geeignet. Die Balanced Accuracy hat den Vorteil, dass die Evaluierung nicht zugunsten der Mehrheitsklasse verzerrt wird. Zur Ausführung der Selektionsfunktion wird eine Diversitätsmetrik benötigt. Zu diesem Zweck wird die normalisierte Doppelfehlerrate gewählt (vgl. Abschnitt 2.4.2), da der Autor der verwendeten Selektionsmethode die Doppelfehlerrate als Diversitätsmetrik empfiehlt [GR01]. Die normalisierte Doppelfehlerrate wird wie folgt berechnet: *1-Doppelfehlerrate*. Es wurde die normalisierte Doppelfehlerrate gewählt, damit bei der Evaluierung der Diversität der Wert 1.0 eine hohe Diversität widerspiegelt, während ein Wert von 0 auf eine geringe Diversität hinweist.

Zur Performanzevaluierung werden die Metriken Accuracy, Balanced Accuracy sowie F1-Makro herangezogen. Makro-Metriken gewichten alle Klassen gleich unabhängig von ihrer Häufigkeit in den Daten. Dies ist insbesonders nützlich, um eine unzureichende Klassifizierung der Minderheitsklassen zu erkennen. Zur Ermittlung der Diversität innerhalb des Ensembles wurde die normalisierte Doppelfehlerrate verwendet. Insgesamt wird jeder Datensatz dreimal evaluiert, um so die Abhängigkeit der Ergebnisse vom Zufall zu reduzieren und um robustere und zuverlässigere Ergebnisse zu erhalten.

Zusammenfassend sind die Evaluierungsparameter des Frameworks wie folgt:

- **Ensemblegröße:** [2,5,10,15,20]
- **Budget für Modelloptimierung:** 40 Min. / 2 Std.
- **Budget für Fusionsoptimierung:** 20 Min. / 1 Std.
- **Verlustfunktion:** 1-Balanced Accuracy
- **Diversitätsmetrik:** Normalisiert Doppelfehlerrate
- **Performanzmetriken:** Accuracy, Balanced Accuracy, F1-Makro

6.1.3 Software

Das Framework ist in der Programmiersprache Python Version 3.8 implementiert und nutzt insgesamt zehn Python-Bibliotheken. Die jeweiligen Versionen und die Verwendungszwecke der Bibliotheken können Tabelle 6.3 entnommen werden.

Bibliothek	Version	Verwendungszweck
scikit-learn[11]	1.3.1	U.a. anwenden maschineller Lernalgorithmen und Vorverarbeitungsmethoden.
numpy[12]	1.24.4	Durchführen numerischer Berechnungen.
pandas[13]	2.0.3	Datenmanipulation.
scipy[14]	1.10.1	U.a. zur Berechnung der Selektion.
ConfigSpace[15]	0.5.0	Aufbau des Konfigurationsraums für die Optimierungsprobleme.
smac[16]	1.4.0	Bayesian Optimization für Optimierungsprobleme.
imbalanced-learn[17]	0.12.4	Anwenden von Samplingmethoden, welche für Klassen-überschneidung mit Klassenungleichgewicht geeignet sind.
openml[1]	0.15.0	Laden von Datensätzen aus dem OpenML Repositorium.
ucimlrepo[2]	0.0.7	Laden von Datensätzen aus dem UCI Repositorium.
mrmr-selection[18]	0.2.8	Anwenden der Merkmalsauswahlmethode MRMR.

Tabelle 6.3: Angabe der verwendeten Python-Bibliotheken des AutoML-Frameworks, so wie deren Verwendungszweck.

6.1.4 Hardware

Die Evaluierung wird auf insgesamt fünf verschiedenen virtuellen Maschinen durchgeführt. Die virtuellen Maschinen eins und zwei sind mit dem Prozessor AMD EPYC-Milan ausgestattet. Dieser arbeitet mit einer Taktfrequenz von 2.50 GHz. Beide Maschinen verfügen über 8 Kerne und 16 GB RAM. Das verwendete Betriebssystem ist Ubuntu 22.04.1 LTS. Die virtuellen Maschinen drei und vier verwenden den Prozessor Intel(R) Xeon(R) CPU mit einer Taktfrequenz von 2.20 GHz. Die Maschinen besitzen jeweils 16 Kerne und 33 GB RAM. Als Betriebssystem wird Ubuntu 22.04.5 LTS verwendet. Die fünfte virtuelle Maschine verwendet ebenfalls den Prozessor Intel(R) Xeon(R) CPU, mit einer Taktfrequenz von 2.20 GHz. Die Maschine weist 8 Kerne und 16 GB RAM auf. Das Betriebssystem ist Ubuntu 22.04.5 LTS.

Für jede Messung werden 8 Kerne verwendet. Das bedeutet, dass auf den virtuellen Maschinen drei und vier jeweils zwei Läufe parallel evaluiert werden.

[11] https://scikit-learn.org
[12] https://numpy.org
[13] https://pandas.pydata.org
[14] https://scipy.org
[15] https://github.com/automl/ConfigSpace
[16] https://github.com/automl/SMAC3
[17] https://imbalanced-learn.org/
[18] https://github.com/smazzanti/mrmr

6.2 Verwendete Baselinemethoden

Zur Bewertung des vorgestellten Ansatzes aus Kapitel 4 werden drei weitere Ansätze als Baseline herangezogen und mit dem erarbeiteten Konzept verglichen. Der in Kapitel 4 beschriebene Ansatz wird im Folgenden als *optimierte Pipeline* bezeichnet. Während die drei Baseline-Methoden unter den Bezeichnungen *bester Basis-Klassifikator*, *statische Pipeline* und *implizite Ensembles* analysiert werden. Bei dem Vergleich der optimierten Pipeline mit dem besten Basis-Klassifikator wird für alle 23 Datensätze die Performanz der optimierten Pipeline mit der Performanz des besten Basis-Klassifikators aus dem jeweiligen Ensemble verglichen. Die Details der anderen beiden Baselinemethoden werden nachfolgend in Abschnitt 6.2.1 und Abschnitt 6.2.2 erläutert.

6.2.1 Statische Pipeline

Der Ansatz, welcher als *statische Pipeline* bezeichnet wird, umfasst im Gegensatz zur *optimierten Pipeline* für alle Basis-Klassifikator dieselbe Datenvorverarbeitungspipeline, siehe Abbildung 6.1. Die Datenvorverarbeitungsmethoden werden direkt auf den Datensatz angewandt bevor das Training des Ensembles stattfindet. Das bedeute, dass während der Klassifikatoroptimierung keine Datenvorverarbeitungsmethoden mit optimiert werden, sondern lediglich die Modellgenerierung optimiert wird (vgl. Abschnitt 5.1). Die *statische Pipeline* wird basierend auf der Datenvorverarbeitungspipeline des Basis-Klassifikator mit der besten Performanz erstellt. Das bedeutet, dass zuerst für jeden Datensatz die *optimierte Pipeline* ausgeführt wird. Anschließend wird aus diesem Evaluationslauf die Datenvorverarbeitungspipeline des besten Basis-Klassifikators ausgewählt. Die Vorverarbeitungsmethoden dieses Klassifikators bilden dann die statische Pipeline für diesen Datensatz. Die Vorverarbeitungsmethoden werden mit den zugehörigen optimierten Hyperparametern übernommen.

6.2.2 Implizite Ensembles

Bei diesem Vergleich wird das Ensemble der *optimierten Pipeline* mit impliziten Ensembles verglichen. Als implizite Ensembles werden in dieser Arbeit Ensembles bezeichnet, welche *implizite Diversität* zur Erstellung des Ensembles verwenden. Das optimierte Ensemble wird dabei gegen *AdaBoost* und *Random Forest* getestet (vgl. Abschnitt 3.1). Die Ensembles der impliziten Diversitätsmethoden werden ebenfalls mit dem *AutoML-Ensemble Framework* erstellt. Zur Erstellung der impliziten Ensembles wird lediglich die Klassifikatoroptimierung des Frameworks verwendet, um die Hyperparameter von Random Forest und AdaBoost zu optimieren. Bei den impliziten Ensembles wird auf eine Datenvorverarbeitung verzichtet. Dies liegt einerseits daran, dass die impliziten Ensembles bereits Strategien zur Generierung impliziter Diversität (vgl. Kapitel 4) enthalten. Zum anderen ist der Random Forest Klassifikator bereits für bestimmte Datencharakteristiken, wie z.B. für Datensätze mit einer geringen Menge an Daten, geeignet [Pol06]. Außerdem arbeitet Random Forest effizient auf hochdimensionalen und verrauschten Daten [Pol06; SR18].

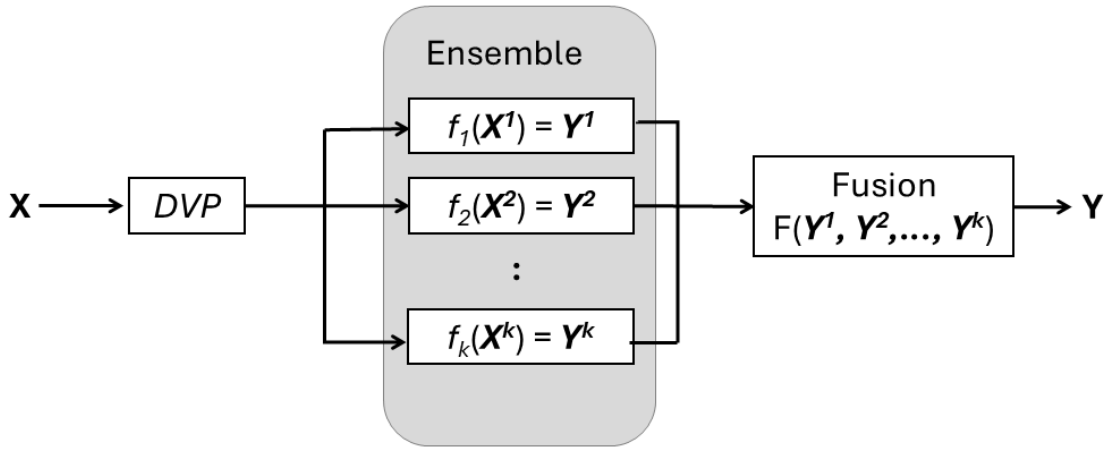

Abbildung 6.1: Schematische Darstellung der implementierten *statischen Pipeline* im *AutoML-Ensemble* Framework. Im Gegensatz zur *optimierten Pipeline* (vgl. Abbildung 5.1) werden die Datenvorverarbeitungsmethoden direkt auf den Datensatz angewendet, bevor das Training des Ensembles stattfindet.

6.3 Evaluation und Diskussion der Performanz

In diesem Abschnitt wird die Performanz der optimierten Pipeline mit der Performanz der drei Baselinemethoden verglichen. Zunächst erfolgt in Abschnitt 6.3.1 der Vergleich zwischen der optimierten Pipeline und dem besten Basis-Klassifikator. Daraufhin wird in Abschnitt 6.3.2 die Performanz der optimierten Pipeline derjenigen der statischen Pipeline gegenübergestellt. Abschließend erfolgt in Abschnitt 6.3.3 der Vergleich der Performanz zu den impliziten Ensembles.

6.3.1 Vergleich von Ensembles mit optimierter Pipeline und besten Basis-Klassifikator

In diesem Abschnitt wird die Performanz des Ensembles der optimierten Pipeline mit der des besten Basis-Klassifikators aus dem jeweiligen Ensemble für alle 23 Datensätze verglichen. Die Analyse erfolgt anhand der Balanced Accuracy, wobei die Ergebnisse für die Metriken Accuracy und F1-Makro mit Ausnahme von zwei Abweichungen analog zu den Ergebnissen der Balanced Accuracy sind. Die Ergebnisse dieses Vergleiches sind in Abbildung 6.2 und Tabelle 6.4 dargestellt und werden in diesem Abschnitt detailliert betrachtet und diskutiert. Die Ensembles und Basis-Klassifikatoren in den Abbildungen sind anhand ihrer ID aufsteigend sortiert. Für jeden Balken des Diagramms ist die ID des Datensatzes angegeben. Die blauen Balken repräsentieren das Ergebnis der optimierten Pipeline, während die orangefarbenen Balken das Ergebnis des besten Basis-Klassifikators eines Datensatzes darstellen. Die Ergebnisse zeigen, dass für 11 der 23 Datensätze das Ensemble eine höhere Balanced Accuracy aufweist als der beste Basis-Klassifikator. Die durchschnittliche Balanced Accuracy beträgt für das Ensemble 0.699 % und für die besten Basis-Klassifikatoren 0.71 %. Beide Ansätze erzielen auf dem Datensatz 207 die beste Klassifikationsleistung, mit einer Performanz von 0.979 % für das Ensemble. Die Perfomanz des besten Basis-Klassifikators beträgt 0.974 %. Die niedrigste Performanz wird auf dem Datensatz 103, mit einer Balanced Accuracy von 0.255 % für das Ensemble und 0.229 % für den besten Basis-Klassifikator beobachtet. Insgesamt

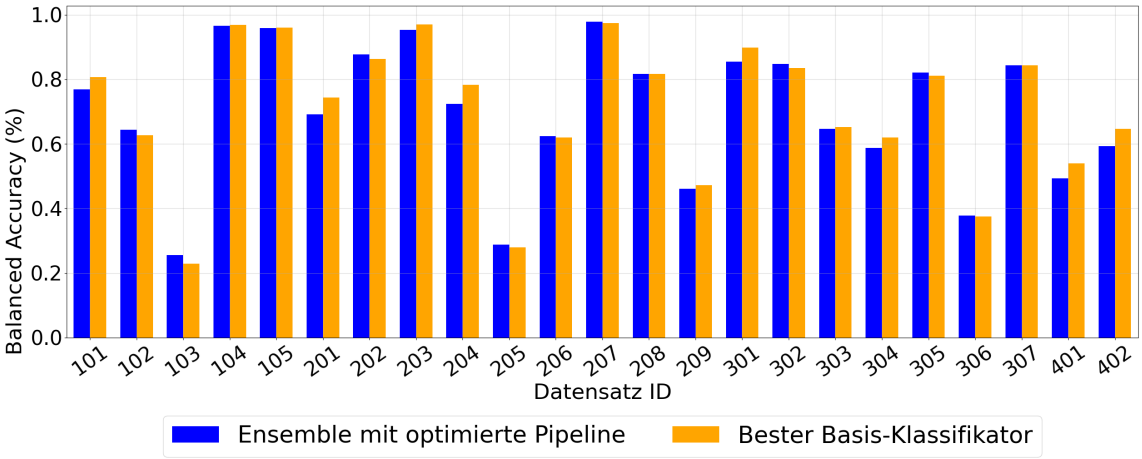

Abbildung 6.2: Vergleich der Leistung zwischen dem Ensemble mit der optimierten Pipeline und dem besten Basis-Klassifikator, für alle 23 Datensätze, anhand der Performanzmetrik Balanced Accuracy.

erreichten vier Datensätze für beide Ansätze eine Performanz von über 0.95 %. Alle Datensätze enthalten mindestens die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht. Im Gegensatz dazu liegt die erreichte Performanz für drei Datensätze bei weniger als 40 %, wobei diese Datensätze durch ihre hohe Dimensionalität gekennzeichnet sind.

Für neun der 23 Datensätze beträgt der Performanzunterschied weniger als 0.01 %-Punkte. Diese Datensätze weisen jeweils mindestens die Datencharakteristik Hochdimensionale Daten oder Klassenüberscheidung mit Klassenunterschiedsgewicht auf. Der Abbildung 6.3 können die Performanzunterschiede für alle Datensätze entnommen werden. Balken, welche blau und positiv sind, zeigen, dass das Ensemble besser abgeschnitten hat. Negative orange Balken bedeuten, dass der beste Basis-Klassifikator besser ist. Bei fünf Datensätzen erreicht der beste Basis-Klassifikator eine Performanz, die um mindestens 0.04%-Punkte der des Ensemble überlegen ist. Die optimierte Pipeline zeigt ihre geringste Überlegenheit bei dem Datensatz 208, mit einem Performanzunterschied von nur 0.001 %-Punkte. Mit 0.059 %-Punkten Performanzunterschied für den Datensatz 204 zeigt der beste Basis-Klassifikator hier die höchste Überlegenheit gegenüber dem Ensemble.

Im Folgenden wird analysiert, wie häufig das Ensemble dem besten Basis-Klassifikator überlegen ist, aufgeschlüsselt nach den vier betrachteten Datencharakteristiken. Dabei werden Datensätze, die mehrere Datencharakteristiken aufweisen, mehrfach berücksichtigt, sodass jede Datencharakteristik separat analysiert wird. Für die Datencharakteristik Hochdimensionale Daten ist das Ensemble für 66.7 % der Datensätze überlegen. Bei der Datencharakteristik Klassenrauschen liegt die Überlegenheit des Ensembles bei 41.7 %, während für Klassenüberschneidung mit Klassenungleichgewicht ein Wert von 43.8 % erreicht wird. Die geringste Überlegenheit weist das Ensemble bei der Datencharakteristik geringe Menge an Daten mit 25 % auf. Daraus kann gefolgert werden, dass das Ensemble insbesondere beim Vorliegen von hochdimensionalen Daten eine bessere Performanz erzielt als der beste Basis-Klassifikator. Im Gegensatz dazu ist die Wahrscheinlichkeit am höchsten, dass der beste Basis-Klassifikator eine bessere Performanz erzielt, wenn ein Datensatz nur eine geringen Datenmenge aufweist.

ID	Ensemble			Bester Basis-Klassifikator		
	Accuracy	**Balanced Accuracy**	**F1-Makro**	**Accuracy**	**Balanced Accuracy**	**F1-Makro**
			Gruppe 1			
101	0.761	0.769	0.758	**0.802**	**0.807**	**0.8**
102	**0.741**	**0.644**	**0.661**	0.697	0.628	0.644
103	**0.256**	**0.255**	**0.232**	0.229	0.229	0.209
104	0.97	0.966	0.966	**0.971**	**0.968**	**0.967**
105	0.938	0.958	0.804	**0.939**	**0.96**	**0.805**
			Gruppe 2			
201	0.687	0.691	0.67	**0.749**	**0.744**	**0.744**
202	**0.877**	**0.877**	**0.876**	0.864	0.864	0.864
203	0.953	0.954	0.94	**0.967**	**0.97**	**0.956**
204	0.717	0.724	0.708	**0.758**	**0.783**	**0.758**
205	**0.288**	**0.288**	**0.266**	0.279	0.279	0.249
206	**0.716**	**0.624**	**0.618**	0.687	0.619	0.615
207	**0.978**	**0.979**	**0.979**	0.973	0.974	0.974
208	0.846	**0.817**	0.827	**0.848**	0.816	**0.828**
209	**0.842**	0.462	**0.516**	0.813	**0.472**	0.506
			Gruppe 3			
301	0.852	0.854	0.85	**0.898**	**0.899**	**0.898**
302	**0.886**	**0.848**	**0.859**	0.831	0.835	0.812
303	0.646	0.646	0.621	**0.65**	**0.652**	**0.642**
304	0.646	0.588	0.558	**0.657**	**0.619**	**0.579**
305	**0.861**	**0.821**	**0.832**	0.847	0.811	0.816
306	**0.473**	**0.378**	**0.357**	0.47	0.376	0.351
307	**0.845**	**0.844**	**0.845**	0.844	**0.844**	0.844
			Gruppe 4			
401	0.488	0.493	0.469	**0.538**	**0.54**	**0.509**
402	0.672	0.593	0.599	**0.686**	**0.647**	**0.629**
Ø	0.736	0.699	0.687	**0.739**	**0.71**	**0.696**

Tabelle 6.4: Vergleich der Ergebnisse des Ensembles mit der optimierten Pipeline gegenüber den Ergebnissen des besten Basis-Klassifikators aus einem Ensemble. Die Werte sind auf drei Nachkommastellen gerundet, der dick hinterlegte Lauf für einen Datensatz erzielte das beste Ergebnis. Unterstrichene Ergebnisse weisen dieselbe Performanz für beide Ansätze auf. Die letzte Zeile stellt die durchschnittliche Performanz eines Ansatzes dar.

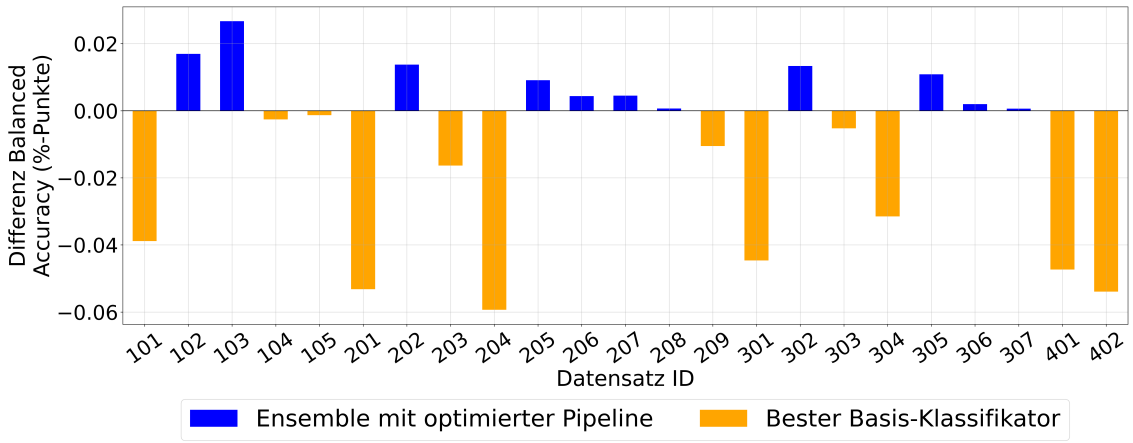

Abbildung 6.3: Aufzeigen des Performanzunterschieds anhand der Balanced Accuracy für alle Datensätze. Balken, welche blau und positiv sind, zeigen, dass das Ensemble besser abgeschnitten hat, negative orange Balken bedeuten, dass der beste Basis-Klassifikator besser ist.

Von den insgesamt zwölf Datensätzen, für die der beste Basis-Klassifikator dem Ensemble überlegen ist, weisen neun Datensätze die Datencharakteristik eine geringe Menge an Daten auf. Die Abbildung 6.4 zeigt die Performanzdifferenz des Ensembles zum besten Basis-Klassifikator für alle Datensätze, die die Datencharakteristik eine geringe Menge an Daten aufweisen. In der Darstellung repräsentieren orange Balken Datensätze, bei denen der beste Basis-Klassifikator eine höhere Performanz als das Ensemble erreicht hat, während blaue Balken eine bessere Performanz des Ensembles zeigen. Im Mittel erreichte das Ensemble eine Balanced Accuracy von 0.74 %, während die der besten Basis-Klassifikatoren 0.764 % beträgt. Für fünf dieser Datensätze ist die erreichte Performanz des besten Basis-Klassifikators um 0.04 %-Punkte überlegen. Bei den drei Datensätzen, für die das Ensemble eine bessere Klassifikationsleistung zeigt, beträgt der Performanzunterschied knapp über 0.01 %. Alle drei Datensätze enthalten als weitere Datencharakteristik Hochdimensionale Daten.

Der Grund für die bessere Performanz der Basis-Klassifikatoren gegenüber der von Ensembles bei einer geringen Menge an Daten kann auf die Art der Aufteilung der Daten in dem *AutoML-Ensemble* Frameworks zurückgeführt werden. In Abschnitt 6.1.1 wurde die Aufteilung des Datensatzes in Trainings-, Validierungs- und Testsets vorgestellt, wobei 70 % der Daten für das Training und jeweils 15% für die Validierung und das Testen vorgesehen sind. Liegt eine geringe Menge an Daten vor, so weist ein Datensatz weniger als 1000 Dateninstanzen auf. Dadurch reduziert sich die Anzahl der Trainingsinstanzen für die Basis-Klassifikatoren auf unter 700 Dateninstanzen, während für die Fusionsalgorithmen maximal 150 Dateninstanzen für das Training zur Verfügung stehen. Dieser Datenmangel führt dazu, dass die Klassifikatoren aus der Klassifikator- und Fusionsoptimierungsphase zur Überanpassung neigen. Besonders kritisch ist die Überanpassung bei der Fusionsoptimierung, da hier wesentlich weniger Daten vorliegen als für das Training der Basis-Klassifikatoren (vgl. Abschnitt 6.1.1). Infolgedessen fällt die Überanpassung des Fusionsalgorithmus deutlich stärker aus als die der Basis-Klassifikatoren, was zu der vergleichsweise geringeren Performanz des Ensembles gegenüber dem Basis-Klassifikator führt.

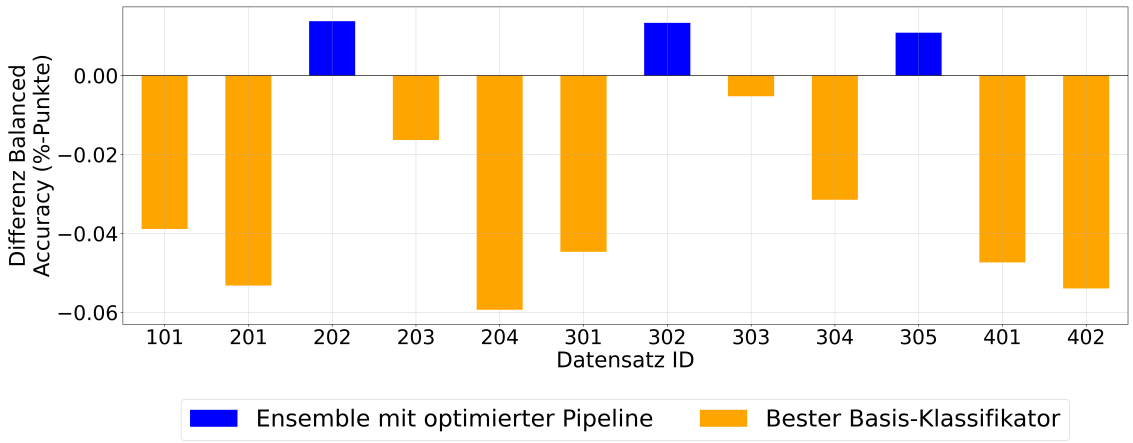

Abbildung 6.4: Aufzeigen des Performanzunterschieds anhand der Balanced Accuracy für alle Datensätze, welche mindestens die Datencharakteristik eine geringe Datenmenge enthalten. Blaue Balken zeigen, dass das Ensemble besser abschneidet. Orangefarbene Balken bedeuten, dass der beste Basis-Klassifikator besser ist.

Die restlichen drei Datensätze der zwölf Datensätze, bei denen der beste Basis-Klassifikator besser ist, enthalten die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht. Zwei dieser Datensätze, die Sätze 104 und 105 gehören der Gruppe 1 an. Dies bedeutet, dass diese Datensätze ausschließlich die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht enthalten. Der Performanzunterschied des Basis-Klassifikators zum Ensemble für die Datensätze 104 und 105 beträgt 0.002 %-Punkte. Wird die Größe der Ensembles für beide Datensätze betrachtet, fällt auf, dass diese verhältnismäßig kleine Ensemblegrößen aufweisen. Die durchschnittliche Ensemblegröße für alle Läufe der optimierten Pipeline beträgt 7.5. Die durchschnittliche Ensemblegröße des Datensatzes 104 beträgt vier, die des Datensatzes 105 beträgt zwei. Eine mögliche Schlussfolgerung ist, dass die Problematik der Klassenüberschneidung mit Klassenungleichgewicht für diese beiden Datensätze mittels einer Vorverarbeitungspipeline relativ eindeutig zu beheben ist, wodurch die Überlegenheit eines Ensembles gegenüber eines einzelnen Klassifikators verloren geht.

Die Kombination der Datencharakteristiken geringe Mengen an Daten und Klassenüberschneidung mit Klassenungleichgewicht erweist sich als problematisch (Datensätze 203, 204, 302, 303, 304, 305, 401, 402, vgl. Abbildung 6.4). In sechs von acht Fällen, in denen diese Datencharakteristiken gemeinsam vorliegen, übertrifft der beste Basis-Klassifikator die Performanz des Ensembles. Die durchschnittliche Performanz des Ensembles beträgt 0.708 %, die der besten Basis-Klassifikatoren liegt bei 0.732 %. Eine mögliche Erklärung für diesen Performanzunterschied könnte in der Wechselwirkung der beiden Datencharakteristiken liegen. Während Hochdimensionale Daten die Effekte der Klassenüberschneidung mit Klassenungleichgewicht mit zunehmender Dimensionalität abschwächen (vgl. Abschnitt 2.2), könnte eine geringe Datenmenge einen gegenteiligen Effekt bewirken.

Ist die Datencharakteristik Klassenrauschen in einem Datensatz enthalten, liegt die Wahrscheinlichkeit bei 58.33 %, dass der beste Basis-Klassifikator besser als das Ensemble ist. Eine nähere Betrachtung der entsprechenden Läufe verdeutlicht, dass alle betroffenen Datensätze mindestens eine weitere Datencharakteristik enthalten. Bei einer detaillierten Analyse der Ergebnisse der 2. Gruppe,

für Datensätze welches Klassenrauschen enthalten und für die der beste Basis-Klassifikator besser abschneidet (201, 209), fällt auf, dass für diese Datensätze bereits der beste Basis-Klassifikator ohne Rauschen eine bessere Performanz erzielt hat (101, 105) (vgl. Abbildung 6.3). Daraus kann abgeleitet werden, dass das Ensemble bei Klassenrauschen eine bessere Leistung zeigt als ein einzelner Klassifikator. Klassenrauschen hingegen gemeinsam mit anderen komplexen Datencharakteristiken, wie einer geringen Menge an Daten, die Problematik dieser verstärkt.

Tritt die Datencharakteristik Hochdimensionale Daten auf, ist der beste Basis-Klassifikator nur in 0.33 % der Fälle und damit selten besser als das Ensemble. Die im Mittel erreichte Balanced Accuracy des Ensembles liegt bei 0.64 %, die der besten-Basis-Klassifikatoren bei 0.646 %. Die Datencharakteristiken kommen insgesamt in zwölf Datensätzen vor, wobei in vier Fällen der beste Basis-Klassifikator besser ist. Allerdings sind in diesen Fällen mindestens die Datencharakteristiken geringe Mengen an Daten enthalten.

6.3.2 Vergleich zwischen optimierter und statischer Pipeline

In diesem Abschnitt wird die Performanz der optimierten Pipeline mit der Performanz des Ensembles der statischen Pipeline, unter Verwendung der Balanced Accuracy, verglichen. Die Ergebnisse dieses Vergleiches sind in Abbildung 6.5 sowie in Tabelle 6.5 dargestellt und werden im Folgenden analysiert und diskutiert. In der Abbildung 6.5 ist die Performanz der optimierten Pipeline durch blaue Balken dargestellt und die der statischen Pipeline durch rote Balken. Die Datensätze sind anhand ihrer ID aufsteigend sortiert. Die Betrachtung der Ergebnisse zeigt, dass für 14 von 23 Datensätzen die optimierte Pipeline eine höhere Balanced Accuracy erreicht hat, als das Ensemble der statischen Pipeline. Die durchschnittliche Balanced Accuracy der optimierten Pipeline liegt bei 0.669 %, während die der statischen Pipeline 0.691 % beträgt. Die höchste Klassifikationsgenauigkeit erzielt die optimierte Pipeline auf dem Datensatz 207 mit 0.979 %. Im Gegensatz dazu erzielt die statische Pipeline auf dem Datensatz 104 ihre beste Performanz mit 0.985 %. Die niedrigste Performanz wird für den Datensatz 103 erzielt, hier beträgt die Balanced Accuracy der optimierten Pipeline 0.255 % und die der statischen Pipeline 0.287 %.

Der Abbildung 6.6 können die Performanzunterschiede zwischen den beiden Ansätzen entnommen werden. Bei zehn der 23 Datensätzen beträgt der Performanzunterschied unter 0.01 %-Punkten, wobei alle Datensätze mindestens die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht aufweisen. Für drei Datensätze ist die optimierte Pipeline um mindestens 0.075%-Punkte der statischen Pipeline überlegen. Im Gegensatz dazu erreicht die statische Pipeline bei zwei Datensätzen eine um mindestens 0.05 %-Punkte bessere Performanz. Der kleinste Performanzunterschied ist bei dem Datensatz 204 vorhanden und beträgt lediglich 0.001 %-Punkte, während der größte Performanzunterschied bei dem Datensatz 305 mit 0.101 %-Punkte verzeichnet wird.

Im Folgenden werden die Datensätze nach der Anzahl ihrer Datencharakteristiken gruppiert. Für jede Gruppe wird der prozentuale Anteil berechnet, bei dem die optimierte Pipeline eine bessere oder äquivalente Leistung wie die statische Pipeline erzielt. Für die Gruppe 1 lag der Anteil bei 60 %, für die Gruppe 2 bei 66.7 % und für die Gruppe 3 und Gruppe 4, bei 71.4 % und 50 %. Anhand der Gruppierung kann abgeleitet werden, dass mit der zunehmenden Anzahl an vorhandenen Charakteristiken (Gruppe 1–3) in einem Datensatz die optimierte Pipeline tendenziell zu einer besseren Ensemble Performanz führt als die statische. Das Ergebnis der 4. Gruppe weicht von

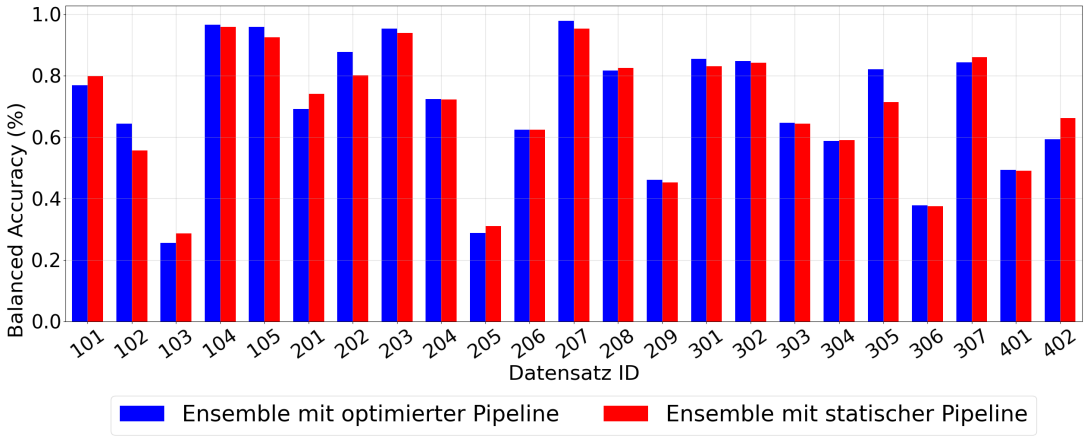

Abbildung 6.5: Vergleich der Performanz des Ensembles mit der optimierten Vorverarbeitungspipeline und der Performanz des Ensembles mit der statischen Vorverarbeitungspipeline.

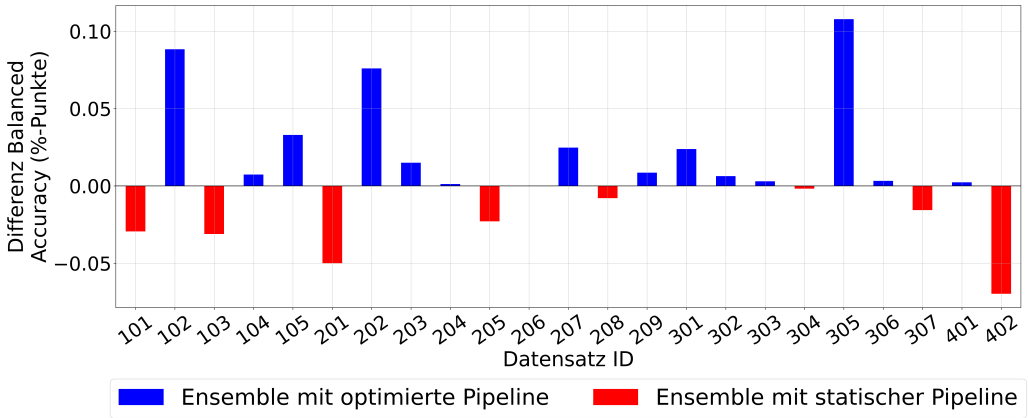

Abbildung 6.6: Aufzeigen des Performanzunterschieds anhand der Balanced Accuracy für alle Datensätze. Balken, welche blau und positiv sind, zeigen, dass das Ensemble mit der optimierten Pipeline besser abschneidet. Rote Balken bedeuten, dass das Ensemble mit der statischen Pipeline besser ist.

diesem Trend ab, allerdings umfasst diese Gruppe lediglich zwei Datensätze. Aus dem Ergebnis der Gruppierungen kann abgeleitet werden, dass mit steigender Anzahl an Datencharakteristiken die Wahrscheinlichkeit sinkt, dass eine einzelne Vorverarbeitungspipeline die Eigenschaft der Daten ausreichend erfasst und dieser entgegenwirkt.

Zunächst wird die durchschnittliche Performanz der Gruppe 1 betrachtet, diese beträgt für die optimierte Pipeline 0.718 % und für die statische Pipeline 0.704 %. Wird die Performanzdifferenz der Gruppe 1 näher betrachtet, fällt auf, das die Performanz der statischen Pipeline für die Datencharakteristiken geringe Menge an Daten (101) und Hochdimensionale Daten (103) deutlich besser als die der optimierten Pipeline ist. Für den Datensatz 101 liegt der Performanzunterschied bei 0.029 %-Punkten, für den Datensatz 103 bei 0.032 %-Punkten. Dieser Unterschied lässt sich auf die Anzahl der erstellten Klassifikatoren während der Klassifikatoroptimierung zurückführen, siehe Abbildung 6.7. Die Abbildung 6.7 zeigt auf der y-Achse die erreichte Balanced Accuracy

ID	Ensemble mit optimierter Pipeline				Ensemble mit statischer Pipeline			
	Accuracy	Balanced Accuracy	F1-Makro	Diversität	Accuracy	Balanced Accuracy	F1-Makro	Diversität
Gruppe 1								
101	0.761	0.769	0.758	0.847	**0.786**	**0.798**	**0.785**	**0.87**
102	**0.741**	**0.644**	**0.661**	**0.771**	0.715	0.556	0.58	0.745
103	0.256	0.255	0.232	0.313	**0.286**	**0.287**	**0.261**	**0.362**
104	**0.97**	**0.966**	**0.966**	**0.978**	0.961	0.958	0.957	0.971
105	0.938	**0.958**	0.804	**0.985**	**0.962**	0.925	**0.846**	0.971
Gruppe 2								
201	0.687	0.691	0.67	0.834	**0.753**	**0.741**	**0.743**	**0.839**
202	**0.877**	**0.877**	**0.876**	**0.917**	0.802	0.801	0.798	0.882
203	**0.953**	**0.954**	**0.94**	**0.966**	0.942	0.939	0.925	0.952
204	**0.717**	**0.724**	**0.708**	0.783	**0.717**	0.723	0.705	**0.812**
205	0.288	0.288	0.266	0.361	**0.311**	**0.311**	**0.294**	**0.373**
206	0.716	**0.624**	**0.618**	**0.786**	**0.729**	**0.624**	0.614	0.777
207	**0.978**	**0.979**	**0.979**	**0.976**	0.951	0.954	0.954	0.967
208	0.846	0.817	0.827	0.856	**0.859**	**0.825**	**0.839**	**0.868**
209	**0.842**	**0.462**	**0.516**	**0.86**	0.836	0.453	0.505	0.849
Gruppe 3								
301	**0.852**	**0.854**	**0.85**	**0.91**	0.827	0.831	0.825	0.842
302	**0.886**	**0.848**	**0.859**	**0.836**	0.883	0.842	0.858	0.574
303	**0.646**	**0.646**	0.621	0.654	0.643	0.643	**0.634**	**0.697**
304	**0.646**	0.588	**0.558**	**0.688**	0.636	**0.59**	0.557	0.679
305	**0.861**	**0.821**	**0.832**	**0.878**	0.767	0.714	0.715	0.821
306	0.473	**0.378**	**0.357**	**0.601**	**0.489**	0.375	0.335	0.556
307	0.845	0.844	0.845	0.858	**0.86**	**0.86**	**0.861**	**0.863**
Gruppe 4								
401	0.488	**0.493**	**0.469**	0.564	**0.493**	0.491	0.466	**0.58**
402	0.672	0.593	0.599	0.759	**0.742**	**0.663**	**0.673**	**0.775**
Ø	0.736	**0.699**	**0.687**	**0.782**	**0.737**	0.691	0.684	0.766

Tabelle 6.5: Vergleich der Ergebnisse des Ensembles mit der optimierten Pipeline gegenüber den Ergebnissen der Ensembles der statischen Pipeline. Die Diversität basiert auf der normalisierten Doppelfehlerrate. Die Werte sind auf drei Nachkommastellen gerundet, der dick hinterlegte Lauf für einen Datensatz erzielte das beste Ergebnis. Unterstrichene Ergebnisse haben dieselbe Performanz für beide Ansätze. Die letzte Zeile stellt die durchschnittliche Performanz eines Ansatzes dar.

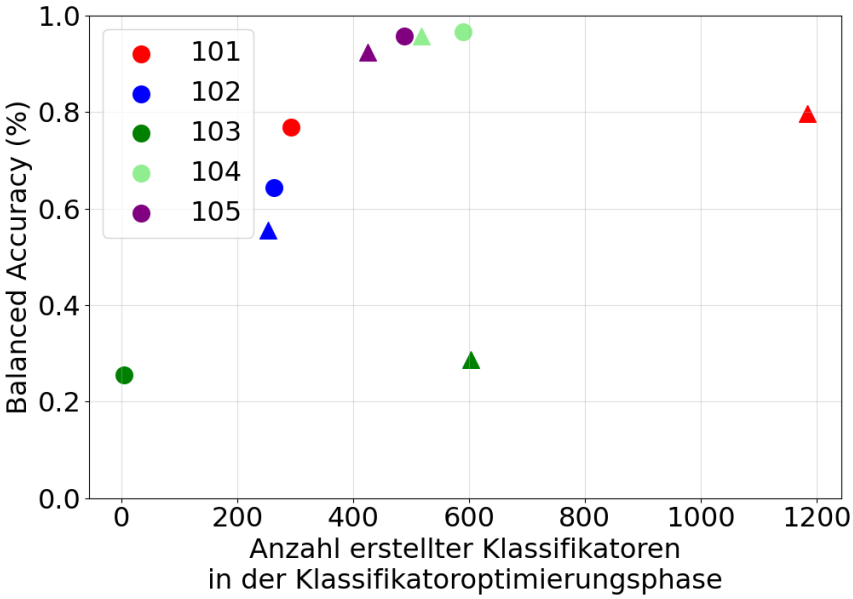

Abbildung 6.7: Darstellung des Zusammenhangs der Anzahl an generierten Klassifikatoren während der Klassifikatoroptimierungsphase und der erreichten Balanced Accuracy eines Durchlaufs. Die Durchläufe der optimierten Pipeline sind als Punkte dargestellt, während die Durchläufe der statischen Pipeline durch Dreiecke gekennzeichnet sind.

für einen Datensatz und auf der x-Achse die Anzahl der generierten Modelle während der Klassifikatoroptimierung an. Die Läufe der optimierten Pipeline sind als Punkt dargestellt, während die Läufe der statischen Pipeline durch ein Dreieck gekennzeichnet sind. Aus der Abbildung kann entnommen werden, dass für die Datensätze 101 und 103 bei der optimierten Pipeline deutlich weniger Modelle als bei der statischen Pipeline generiert wurden. Es werden mindestens 600 Modelle weniger erstellt. Bei den übrigen drei Datensätzen ist die Anzahl der erstellten Modelle für beide Pipelines relativ gleich, in diesem Fall ist die Performanz der optimierten Pipeline besser als die der statischen Pipeline.

Die durchschnittliche Performanz der optimierten Pipeline für die Gruppe 2 beträgt 0.712 %, die der statischen Pipeline 0.707 %. In dieser Gruppe übertrifft die statische Pipeline in drei Fällen (201, 205, 208) die optimierte Pipeline. Auffällig ist, dass die Datensätze 101 und 103 erneut in der Gruppe 2 vertreten sind. Dieses Mal mit der zusätzlichen Datencharakteristik Klassenrauschen. Die IDs der Datensätze von 101 und 103 sind in der Gruppe 2 als Datensätze 201 und 205 enthalten.

Die statische Pipeline erreichte für den Datensatz 201 eine Balanced Accuracy von 0.741 %, während die optimierte Pipeline 0.691 % beträgt. Zur Analyse der Performanzunterschiede zwischen der optimierten und der statischen Pipeline wird die Anzahl der generierten Modelle während der Klassifikatoroptimierung untersucht. Es zeigt sich, dass die Überlegenheit der statischen Pipeline gegenüber der optimierten Pipeline mit zunehmender Differenz in der Anzahl der während der Klassifikatoroptimierung erzeugten Klassifikatoren wächst.

Die bessere Performanz der statischen Pipeline für Datensatz 205 ist ebenfalls auf die Anzahl der generierten Klassifikatoren während der Klassifikatoroptimierung zurückzuführen. Die optimierte Pipeline erzielt eine Performanz von 0.228 % und die Performanz der statischen Pipeline liegt bei 0.311 %. Bei der optimierten Pipeline werden während der Klassifikatoroptimierung lediglich vier Klassifikatoren erstellt, die zugleich das Ensemble bilden. Im Gegensatz dazu erstellt die Klassifikatoroptimierung der statischen Pipeline im Schnitt 272 Klassifikatoren, wodurch eine größere Auswahl an Klassifikatoren für die Bildung eines Ensembles zur Verfügung steht. Unter diesen 272 Klassifikatoren befinden sich solche, die eine höhere Klassifikationsleistung erzielen als das Ensemble der optimierten Pipeline. Der Grund für die geringe Anzahl an generierten Klassifikatoren bei der optimierten Pipeline ist darauf zurückzuführen, dass die Laufzeit von Vorverarbeitungsmethoden sowie Klassifikatoren mit zunehmender Größe des Datensatzes ansteigen (vgl. Tabelle 4.1). Aufgrund des begrenzten Zeitbudgets für die Klassifikatoroptimierung können bei der optimierten Pipeline daher weniger Hyperparameterkonfigurationen für die verschiedenen Vorverarbeitungsmethoden und Klassifikatoren erstellt und evaluiert werden.

Eine Analyse des Datensatzes 208 zeigt, dass die statische Pipeline im Schnitt eine um weniger als 0.01 %-Punkte bessere Performanz erzielt als die optimierte Pipeline. Die Klassifikationsleistung der optimierten Pipeline liegt bei 0.817 %, die der statischen bei 0.825 %. Hierbei ist auffällig, dass die durchschnittliche Ensemblegröße für den Datensatz 2.5 beträgt. Im Durchschnitt über alle Datensätze weisen die Ensembles der optimierten Pipeline eine Ensemblegröße von 7.5 auf. Die der statischen Pipeline liegt im Durchschnitt bei 11.1 Basis-Klassifikatoren. Dies deutet darauf hin, dass für diesen Datensatz ein einzelner Klassifikator ausreichend ist, um die zugrundeliegenden Muster im Datensatz zu erfassen (vgl. Abschnitt 6.3.1). Dadurch geht der Vorteil der verschiedenen Vorverarbeitungspipelines bei der optimierten Pipeline verloren.

In der Gruppe 3 ist die optimierte Pipeline in fünf von sieben Fällen der statischen Pipeline überlegen. Die durchschnittliche Performanz der optimierten Pipeline für die Gruppe 3 liegt bei 0.711 %, die der statischen Pipeline bei 0.694 %. Bei der Analyse der Ergebnisse kann ein Zusammenhang zwischen der Performanz des besten Basis-Klassifikators und der Performanz der statischen Pipeline festgestellt werden. Die Abbildung 6.8 zeigt für die Gruppe 3 und Gruppe 4 einen Vergleich der Performanz der Pipelines und des besten Basis-Klassifikators für jeden Durchlauf. Zur Erinnerung, jeder Datensatz wird insgesamt dreimal für jeden Ansatz evaluiert. Zum einem um so die Abhängigkeit der Ergebnisse vom Zufall zu reduzieren. Zum anderen, um robustere und zuverlässigere Ergebnisse zu erhalten (vgl. Abschnitt 6.1.2). Die blauen Balken zeigen dabei den Performanzunterschied der optimierten Pipeline gegenüber dem besten Basis-Klassifikator an. Hierbei bedeutet ein positiver Wert, dass das optimierte Ensemble eine höhere Performanz erreicht. Der rote Balken stellt den Performanzunterschied der optimierten Pipeline gegenüber der statischen Pipeline dar. Ein positiver Wert bedeutet, dass das Ensemble der optimierten Pipeline eine höhere Klassifikationsgenauigkeit erreicht. Diese Darstellung verdeutlicht, dass in Fällen, in denen der beste Basis-Klassifikator der optimierten Pipeline deutlich überlegen ist, auch die statische Pipeline der optimierten Pipeline überlegen ist. Die Performanz des besten Basis-Klassifikators gilt als deutlich überlegen, wenn diese um mindestens 0.01 %-Punkte höher ist. Daraus kann abgeleitet werden, dass der beste Basis-Klassifikator bereits eine Vorverarbeitungspipeline gefunden hat, die die relevanten Muster in den Daten hinreichend erfasst und damit die Überlegenheit der optimierten Pipeline verloren geht.

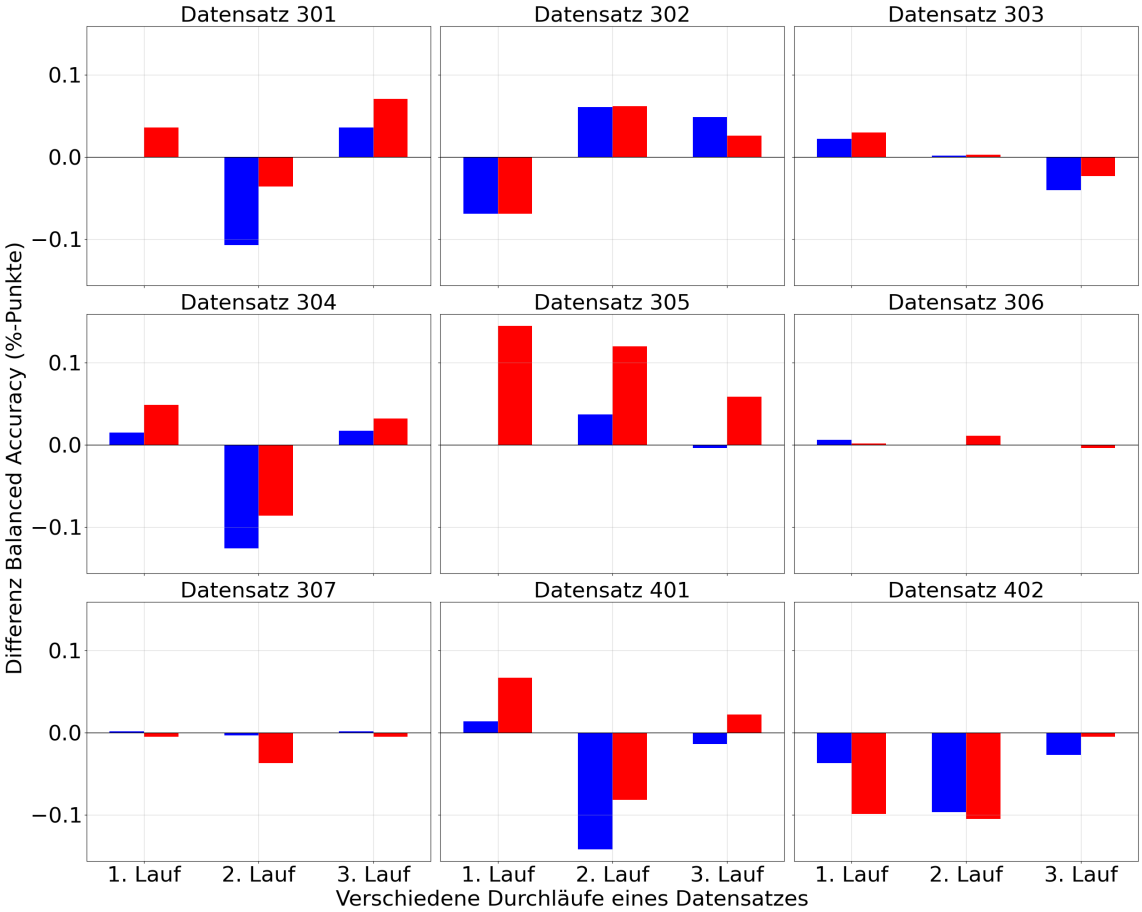

Abbildung 6.8: Darstellung der Performanzunterschiede der verschiedenen Durchläufe für die Datensätze der Gruppen 3 und 4. Die verschiedenen Durchläufe werden in dem Diagramm als 1.Lauf, 2. Lauf und 3. Lauf betitelt. Der blaue Balken zeigt den Unterschied der Performanz zwischen dem optimierten Ensemble und dem besten Basis-Klassifikator. Der rote Balken zeigt den Performanzunterschied zwischen der optimierten und der statischen Vorverarbeitungspipeline.

Insgesamt kann für Datensätze, welche mehr als zwei Datencharakteristiken aufweisen, festgehalten werden, dass für Durchläufe, in denen der beste Basis-Klassifikator eine höhere Balanced Accuracy, als das optimierte Ensemble aufweist, auch die statische Pipeline eine bessere Performanz aufweist. Die Gemeinsamkeit der statischen Pipeline und des besten Basis-Klassifikators liegt in der Vorverarbeitungspipeline. Dies deutet darauf hin, dass die Wahl der Vorverarbeitungsmethoden mit zunehmender Anzahl an Datencharakteristiken einen höheren Einfluss auf die Klassifikationsleistung hat als die Anzahl der Klassifikatoren.

In der Gruppe 4, die zwei Datensätze umfasst, zeigt die optimierte Pipeline für den Datensatz 401 mit 0.469 % eine um 0.002 %-Punkte bessere Performanz. Wohingegen bei dem Datensatz 402 die statische Pipeline mit 0.663 % überlegen ist. Für den Datensatz 402 erreichte die optimierte Pipeline eine Performanz von 0.593 %. Ein Vergleich mit Abbildung 6.8 zeigt auch hier einen Zusammenhang zwischen dem besten Basis-Klassifikator und der Performanz der statischen Pipeline. Für die Fälle, in denen der beste Basis-Klassifikator eine deutlich bessere Performanz aufweist als die optimierte

Pipeline, erzielt auch die statische Pipeline eine bessere Performanz als die optimierte Pipeline. Es ist hervorzuheben, dass die optimierte Pipeline für den Datensatz 401 überlegen ist, obwohl die statische Pipeline deutlich mehr Klassifikatoren während der Klassifikatorptimierungsphase erstellt hat als die optimierte Pipeline. Die statische Pipeline generiert im Schnitt 5950 Klassifikatoren, die optimierte Pipeline hingegen nur 109. Die zeigt, dass die Optimierung der Vorverarbeitungsmethoden trotz der erhöhten Laufzeit einen entscheidenden Beitrag zur Leistungssteigerung eines Ensembles leisten kann, wenn einem Datensatz mehrere komplexe Datencharakteristiken zugrunde liegen.

Zusammenfassend kann festgehalten werden, dass die Anzahl der erstellten Klassifikatoren während der Klassifikatoroptimierung einen starken Einfluss auf die Performanz des Ensembles ausübt. In den meisten Fällen gilt: Je mehr Klassifikatoren während der Klassifikatoroptimierung generiert werden, desto besser ist die Klassifikationsleistung des erstellten Ensembles. Dies zeigt sich insbesondere bei Datensätzen mit weniger als drei Datencharakteristiken. Eine mögliche Erklärung hierfür ist, dass mit zunehmender Anzahl an Datencharakteristiken in einem Datensatz die Anwendung einer einzigen Vorverarbeitungspipeline nicht mehr ausreichend ist, um die zugrundeliegenden Muster in den Daten vollständig zu erfassen. In solchen Fällen geht der Vorteil einer höheren Anzahl an generierten Klassifikatoren während der Klassifikatoroptimierung verloren. Des Weiteren scheint die Performanz der statischen Pipeline stark von der des besten Basis-Klassifikators ab Gruppe 3 abhängig zu sein. Wenn ein einzelner Basis-Klassifikator die Daten besser klassifiziert als das Ensemble, deutet dies darauf hin, dass eine Vorverarbeitungspipeline gefunden worden ist, welche die zugrundeliegenden Muster in den Daten effektiv identifiziert und modelliert. Dies führt dazu, dass die statische Pipeline eine höhere Vorhersagegenauigkeit erreicht als die optimierte Pipeline.

6.3.3 Vergleich zwischen der optimierten Pipeline und den impliziten Ensembles

Als dritter Vergleich wird die Performanz der optimierten Pipeline mit der Performanz der impliziten Ensembles verglichen. Die Ergebnisse sind in Tabelle 6.6 und in der Abbildung 6.9 zusammengefasst. Insgesamt trifft die optimierte Pipeline in einer großen Mehrheit der Datensätze genauere Vorhersagen im Vergleich zu den impliziten Ensembles. Dabei gilt, dass das Ensemble mit der optimierten Pipeline die beste Performanz zeigt, gefolgt von Random Forest, während AdaBoost die geringste Performanz aufweist.

Zunächst wird die Performanz des Ensembles mit der optimierten Pipeline mit der von Random Forest verglichen. Die optimierte Pipeline erreicht für 18 von 23 Datensätzen eine höhere Klassifikationsleistung. Im Durchschnitt über alle Datensätze erreicht die optimierte Pipeline eine Accuracy von 0.736 % und eine Balanced Accuracy von 0.699 %. Die Accuracy des Random Forest beträgt gemittelt über alle Datensätze 0.664 % und die Balanced Accuracy 0.606 %. Bis auf eine Ausnahme gilt, dass der Ansatz, welcher die höhere Balanced Accuracy aufweist, auch die höhere Accuracy erreicht. Daher erfolgt die Analyse der Ergebnisse anhand der Balanced Accuracy, wobei die Ergebnisse für die Metrik Accuracy analog sind. Die optimierte Pipeline erreicht für zehn Datensätze eine Balanced Accuracy von über 0.85%, der Random Forest Algorithmus erreicht diese Leistung hingegen nur für sieben Datensätze. Die optimierte Pipeline weist für fünf Datensätze eine höhere Balanced Accuracy von über 0.25 %-Punkten auf, alle fünf Datensätze sind hochdimensional.

Für die Läufe mit den Datensätzen 101, 201, 204, 301 und 306 weist der Random Forest Algorithmus eine höhere Performanz auf. Bis auf den Datensatz 306 weisen alle Datensätze die Charakteristik einer geringen Menge an Daten auf (vgl. Abschnitt 6.3.1). Die bessere Performanz des Random

Forest Algorithmus für diese Datensätze wird damit begründet, dass die optimierte Pipeline zur Überanpassung während der Klassifikator- und insbesondere der Fusionsoptimierung neigt, wenn eine geringe Datenmenge vorliegt. Der Datensatz 306 enthält die drei Datencharakteristiken Hochdimensionale Daten, Klassenrauschen und Klassenüberschneidung mit Klassenungleichgewicht. Auf diesen Datensatz werden insgesamt fünf Vorverarbeitungsmethoden angewandt, entsprechend den zugrundeliegenden Datencharakteristiken des Datensatzes (vgl. Abschnitt 4.3). Aufgrund der Anwendung von fünf Vorverarbeitungsmethoden in Kombination mit der hohen Dimensionalität des Datensatzes ist die Laufzeit für die Erstellung eines Klassifikators einschließlich der zugehörigen Datenvorverarbeitungsmethoden erheblich erhöht (vgl. Tabelle 4.1). Daher werden während der Klassifikatoroptimierung der optimierten Pipeline lediglich zwei Klassifikatoren erstellt, was zu einer geringen Leistung der optimierten Pipeline von 0.378 % führte.

Zusammenfassend kann festgehalten werden, dass die optimierte Pipeline dem Random Forest überlegen ist, solange die optimierte Pipeline nicht aufgrund einer geringen Datenmenge überangepasst ist. Daraus kann geschlussfolgert werden, dass die optimierte Pipeline besser komplexe Datencharakteristiken handhabt als der Random Forest. Es ist hervorzuheben, dass für die Datensätze 103, 205 und 206 kein Random Forest Klassifikator innerhalb des gegebenen Modellbudgets von zwei Stunden erstellt wird. Alle drei Datensätze sind hochdimensional. Dies ist überraschend, da Random Forest sich eigentlich dadurch auszeichnet, effizient bei der Verarbeitung großer Datenmengen zu sein [Pol06; SR18]. Dies kann möglicherweise dadurch erklärt werden, dass die drei Datensätze besonders hochdimensional sind. Die Datensätze 103 und 205 weißen 3 037 Merkmale auf, der Datensatz 306 enthält 60 000 Dateninstanzen und 350 Merkmale.

Ein Vergleich der Performanz von AdaBoost mit der des Ensembles der optimierten Pipeline zeigt, dass letzteres bei 21 von 23 Datensätzen eine bessere Performanz aufweist. Die durchschnittliche Accuracy von AdaBoost über alle Datensätze hinweg liegt bei 0.601 %. Die Balanced Accuracy beträgt 0.544 %. Damit weist AdaBoost im Vergleich zu dem Ensemble der optimierten Pipeline im Schnitt eine geringere Performanz von 0.135 %-Punkten für die Accuracy und von 0.155 %-Punkten für die Balanced Accuracy auf. Das Ensemble der optimierten Pipeline erzielt für sieben Datensätze eine höhere Balanced Accuracy von 0.2 %-Punkten als AdaBoost. Diese Datensätze enthalten mindestens die Datencharakteristik Klassenrauschen oder Klassenüberschneidung mit Klassenungleichgewicht. Die schlechte Performanz von AdaBoost ist darauf zurückzuführen, dass AdaBoost auf verrauschten Datensätzen eine schlechte Klassifikationsleistung aufweist [Pol06]. Die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht kann auch als eine Art von Klassenrauschen angesehen werden [FGG+18].

Die Datensätze 103 und 306 repräsentieren die Ausnahme, bei der AdaBoost die Performanz des optimierten Ensembles übertrifft. Die Performanz des optimierten Ensembles für den Datensatz 103 beträgt 0.255 %, die von AdaBoost 0.293 %. Für den Datensatz 306 beträgt die Klassifikationsleistung des Ensembles der optimierten Pipeline 0.378 %, die von AdaBoost 0.438 %. Diese geringe Performanz lässt sich darauf zurückführen, dass beide Datensätze hochdimensional sind und die optimierte Pipeline in der Klassifikatoroptimierung maximal vier Klassifikatoren generiert. Dadurch erzielt AdaBoost eine bessere Performanz als die optimierte Pipeline.

Der Vergleich der Performanz von AdaBoost mit der des Ensembles mit optimierter Pipeline zeigt, dass das Ensemble mit der optimierten Pipeline komplexe Datencharakteristiken besser handhabt als AdaBoost.

DS	Ensemble mit Optimierter Pipeline			Random Forest			AdaBoost		
	Accuracy	Balanced Accuracy	Diversity	Accuracy	Balanced Accuracy	Diversity	Accuracy	Balanced Accuracy	Diversity
Gruppe 1									
101	0.761	0.769	0.847	**0.819**	**0.817**	**0.863**	0.724	0.715	0.769
102	**0.741**	**0.644**	**0.771**	0.651	0.509	0.65	0.254	0.135	0.176
103	0.256	0.255	**0.313**	-	-	-	**0.293**	**0.293**	0.231
104	**0.97**	**0.966**	**0.978**	0.885	0.862	0.892	0.609	0.661	0.405
105	0.938	**0.958**	**0.985**	**0.974**	0.881	0.975	0.915	0.782	0.558
Gruppe 2									
201	0.687	0.691	**0.834**	**0.741**	**0.72**	0.824	0.671	0.655	0.774
202	**0.877**	**0.877**	**0.917**	**0.877**	0.876	0.884	0.815	0.815	0.833
203	**0.953**	**0.954**	**0.966**	0.786	0.672	0.795	0.686	0.598	0.55
204	0.717	0.724	0.783	**0.778**	**0.78**	**0.8**	0.586	0.496	0.484
205	**0.288**	**0.288**	**0.361**	-	-	-	0.277	0.277	0.223
206	**0.716**	**0.624**	**0.786**	-	-	-	0.585	0.516	0.506
207	**0.978**	**0.979**	0.976	0.973	0.974	**0.977**	0.787	0.787	0.511
208	**0.846**	**0.817**	**0.856**	0.797	0.766	0.789	0.596	0.557	0.428
209	**0.842**	**0.462**	**0.86**	0.853	0.439	0.855	0.827	0.399	0.542
Gruppe 3									
301	0.852	0.854	**0.91**	**0.864**	**0.862**	0.887	0.79	0.792	0.823
302	**0.886**	**0.848**	**0.836**	0.697	0.546	0.665	0.556	0.479	0.485
303	**0.646**	**0.646**	**0.654**	0.596	0.59	0.542	0.472	0.469	0.369
304	**0.646**	**0.588**	**0.688**	0.657	0.578	0.687	0.525	0.469	0.425
305	**0.861**	**0.821**	**0.878**	0.694	0.629	0.732	0.639	0.577	0.491
306	0.473	0.378	0.601	**0.692**	**0.611**	**0.732**	0.535	0.438	0.503
307	**0.845**	**0.844**	**0.858**	0.84	0.838	0.834	0.804	0.802	0.369
Gruppe 4									
401	**0.488**	**0.493**	**0.564**	0.477	0.474	0.483	0.394	0.392	0.355
402	**0.672**	**0.593**	**0.759**	0.631	0.508	0.604	0.492	0.401	0.484
Ø	**0.736**	**0.699**	**0.782**	0.664	0.606	0.673	0.601	0.544	0.491

Tabelle 6.6: Vergleich der Ergebnisse des Ensembles mit der optimierten Pipeline gegenüber den Ergebnissen der Ensembles mit impliziter Diversität (Random Foresst und AdaBoost). Die Diversität basiert auf der normalisierten Doppelfehlerrate. Die Werte sind auf drei Nachkommastellen gerundet, der dick hinterlegte Durchlauf für einen Datensatz erzielte das beste Ergebnis.

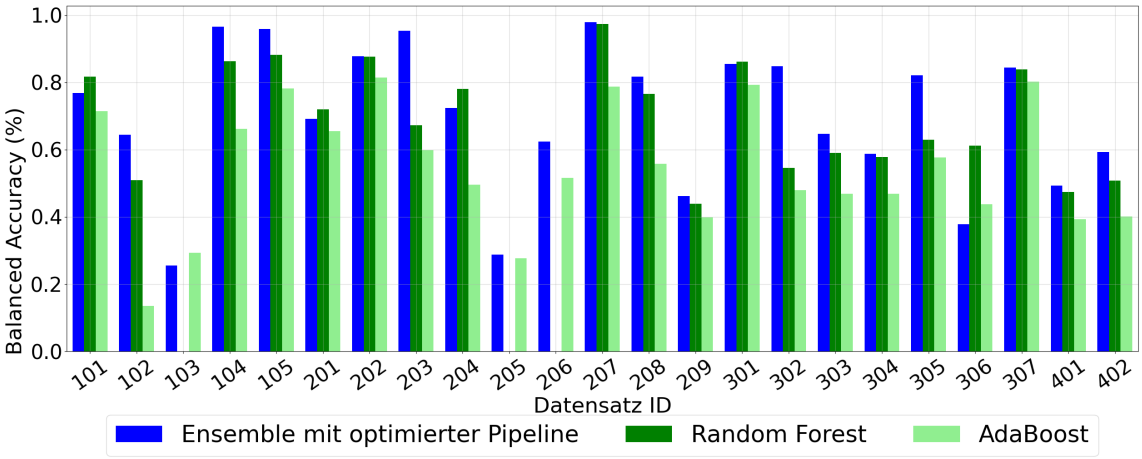

Abbildung 6.9: Vergleich der Performanz, anhand der Balanced Accuracy, für die Ensembles der optimierten Pipeline gegenüber den Ensembles mit impliziter Diversität. Die Ensembles mit impliziter Diversität sind Random Forest und AdaBoost.

6.4 Evaluation und Diskussion der Diversität

In diesem Abschnitt wird die Diversität der optimierten Pipeline mit der Diversität von zwei der Baselinemethoden anhand der normalisierten Doppelfehlerrate verglichen (vgl. Abschnitt 6.1). Zunächst wird in Abschnitt 6.4.1 die Diversität der Ensembles mit der optimierten Pipeline mit der Diversität der Ensembles mit der statischen Pipeline verglichen. In Abschnitt 6.4.2 erfolgt ein Vergleich der Diversität der Ensembles mit der optimierten Pipeline mit der Diversität der Ensembles, die eine implizierte Diversität aufweisen.

6.4.1 Vergleich zwischen optimierter und statischer Pipeline

In diesem Kapitel wird für die 23 Datensätze die Diversität der Ensembles der optimierten Pipeline mit der Diversität der Ensembles der statischen Pipeline verglichen. Die Ergebnisse dieses Vergleiches sind in Abbildung 6.10 sowie Tabelle 6.5 dargestellt und werden in diesem Abschnitt analysiert und diskutiert. Für 13 von 23 Datensätzen erzielte das Ensemble der optimierten Pipeline eine höhere Diversität als das Ensemble der statischen Pipeline. Die durchschnittliche Diversität über alle Datensätze beträgt für die Ensembles der optimierten Pipeline 0.782 % und für die Ensembles der statischen Pipeline 0.766%. Somit ist die Diversität der Ensembles der optimierten Pipeline um 0.016 %-Punkte höher, siehe Abbildung 6.11. In der Abbildung 6.11 repräsentieren helle Farben den Performanzunterschied der Balanced Accuracy für einen Datensatz, während dunkle Farben den Unterschied der erreichten Diversität darstellen. Ein blauer Balken zeigt an, dass die Performanz bzw. die Diversität des Ensembles mit der optimierten Pipeline höher ist, während rote Balken auf eine höhere Performanz bzw. Diversität des Ensembles mit der statischen Pipeline hinweisen. Das Ensemble mit der optimierten Pipeline erreicht für fünf Datensätze eine höhere Diversität von mindestens 0.035 %-Punkten. Die Datensätze weisen mindestens die Datencharakteristik einer geringen Menge an Daten oder Hochdimensionale Daten auf. Im Gegensatz dazu erreicht das

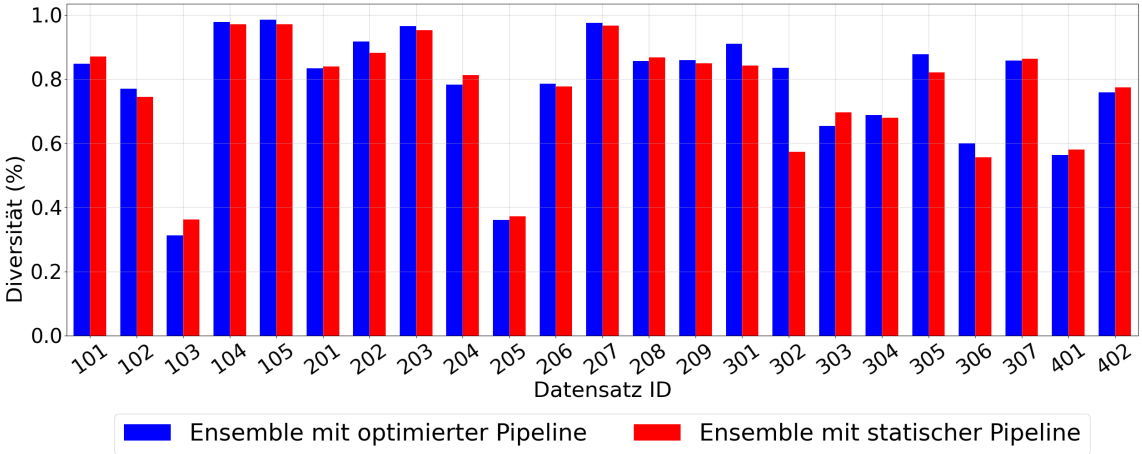

Abbildung 6.10: Vergleich der Diversität zwischen den Ensembles der optimierten Pipeline und den Ensembles der statischen Pipeline, anhand der normalisierten Doppelfehlerrate.

Ensemble mit der statischen Pipeline bei zwei Datensätzen (103, 303) eine um mindestens 0.035 %-Punkten höhere Diversität. Beide Datensätze zeichnen sich durch ihre hohe Dimensionalität aus.

Für 12 der 13 Datensätze (102, 104, 105, 202, 203, 206, 207, 209, 301, 302, 305, 306), bei denen die optimierte Pipeline eine höhere Diversität erreicht hat als die statische Pipeline, erzielen die Ensembles der optimierten Pipeline ebenfalls eine höhere Klassifikationsgenauigkeit. siehe Abbildung 6.11. Dieses Ergebnis stimmt mit der Annahme von Kuncheva überein, dass diverse Ensembles zu einer besseren Performanz führen [Kun04].

Eine Ausnahme von dieser Annahme bilden jedoch die Datensätze 204, 303 und 401, bei denen das optimierte Ensemble eine höhere Balanced Accuracy, jedoch eine geringere Diversität als die statische Pipeline aufweist. Für alle drei Datensätze liegt der Performanzunterschied für die Balanced Accuracy bei unter 0.005 %. Bei dem Datensatz 304 tritt das gegenteilige Ereignis ein, hier weist die statische Pipeline eine bessere Performanz und eine geringere Diversität als die optimierte Pipeline auf. Der Performanzunterschied der beiden Pipelines liegt erneut bei unter 0.005 %-Punkten. Dieser geringe Unterschied in der Performanz könnte dazu führen, dass das Ergebnis von der Annahme Kunchevas abweicht.

6.4.2 Vergleich zwischen der optimierten Pipeline und den impliziten Ensembles

Die folgende Untersuchung vergleicht die Diversität der optimierten Pipeline mit der Diversität der Ensembles mit impliziter Diversität anhand der normalisierten Doppelfehlerrate. Die Ergebnisse sind in der Tabelle 6.6 und in der Abbildung 6.12 zusammengefasst. Insgesamt zeigt sich, dass die optimierte Pipeline im Vergleich zu den Ensembles mit impliziter Diversität eine höhere Diversität aufweist. Das Ensemble mit der optimierten Pipeline erreicht für 20 der 23 Datensätze die höchste Diversität. Für die restlichen drei Datensätze erlangt der Random Forest Klassifikator die höchste Diversität. AdaBoost weist für keinen Datensatz die höchste Diversität auf. Die durchschnittliche

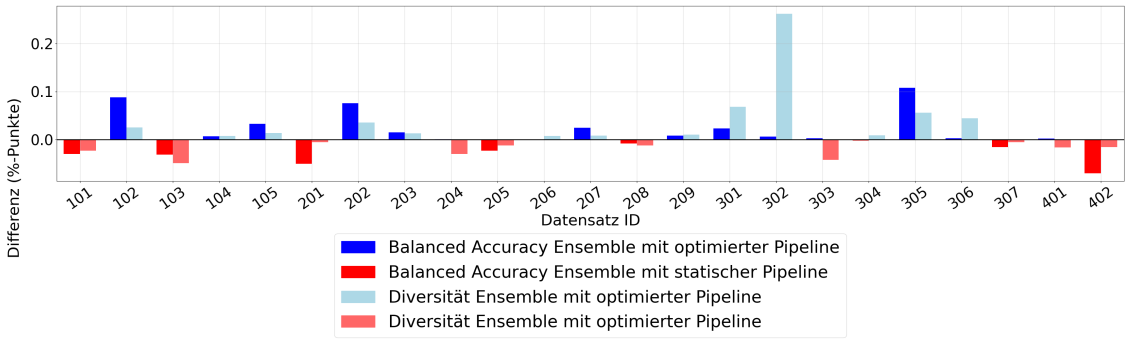

Abbildung 6.11: Darstellung der Performanz- und Diversitätsdifferenz für die Ensembles der optimierten und der statischen Pipelines. Die dunkle Farbe gibt hierbei die Performanzdifferenz der Balanced Accuracy an. Die helleren Farben geben die Diversitätsdifferenz an. Zeigen die Balken nach oben, so hat das Ensemble mit der optimierten Pipeline eine höhere Performanz oder Diversität erreicht. Wenn die Balken nach unten zeigen, so hat Random Forest eine höhere Performanz oder Diversität erreicht

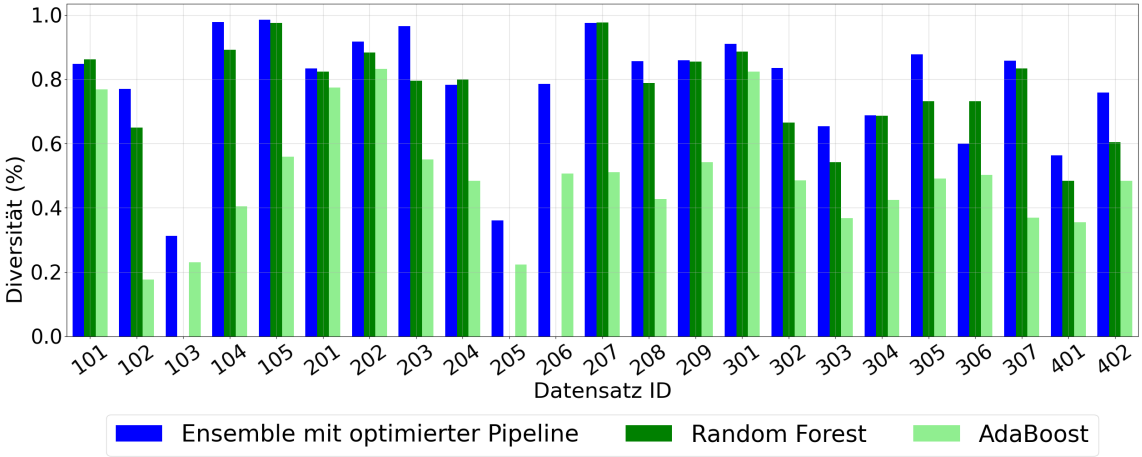

Abbildung 6.12: Vergleich der Diversität zwischen den Ensembles mit der optimierten Pipeline (blau), den Random Forest Ensembles (dunkelgrün) und AdaBoost (hellgrün).

Diversität, über alle Datensätze gemittelt, beträgt für das Ensemble mit der optimierten Pipeline 0.783 %. Für das Random Forest Ensemble liegt die durchschnittliche Diversität bei 0.673 % und für AdaBoost bei 0.491 %.

Das Ensemble mit der optimierten Pipeline erreicht für 20 Datensätze eine höhere Diversität als die Random Forest Ensembles. Für 17 der 20 Datensätze erreicht das Ensemble mit der optimierten Pipeline ebenfalls eine höhere Balanced Accuracy (vgl. Abschnitt 6.3.3). Die Random Forest Ensembles erreichen für vier Datensätze eine höhere Diversität als die optimierte Pipeline. Für drei der vier Datensätze (101, 204, 306) zeigt Random Forest sowohl eine höhere Diversität als auch eine bessere Performanz als die optimierte Pipeline. Das bedeutet, dass für 19 der 23 Datensätze, bei denen ein Ansatz die höhere Diversität aufweist, derselbe Ansatz eine höhere Balanced Accuracy erreicht. Dieses Ergebnis unterstützt die Annahme von Kuncheva, dass eine höhere Diversität eines

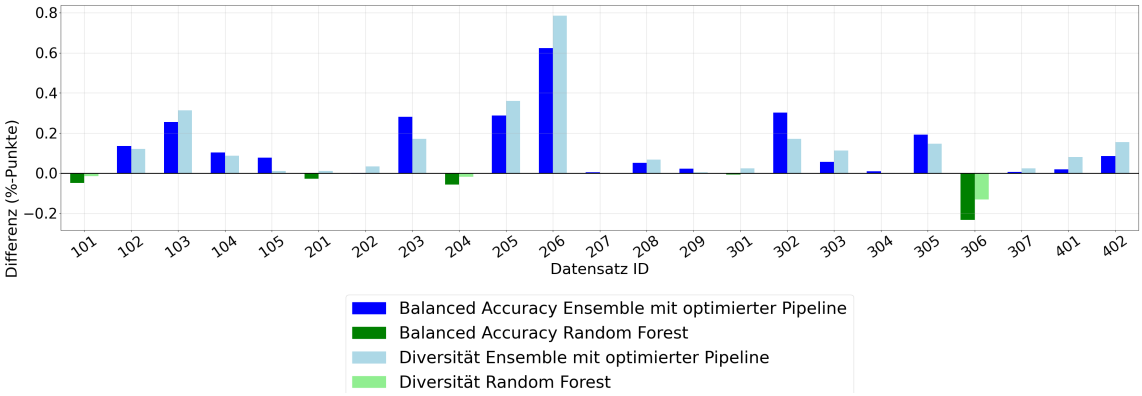

Abbildung 6.13: Darstellung der Performanz- und Diversitätsdifferenz zwischen den Ensembles mit optimierter Pipeline und den Random Forest Ensembles. Die dunkle Farbe gibt hierbei die Performanzdifferenz an und die hellere Farbe die Diversitätsdifferenz. Zeigen die Balken nach oben und sind blau, so schneidet die optimierte Pipeline besser ab.

Ensembles zu einer besseren Performanz führt [Kun04], siehe Abbildung 6.13. In der Abbildung 6.13 repräsentieren helle Farben den Performanzunterschied der Balanced Accuracy für einen Datensatz. Dunkle Farben hingegen stellen den Unterschied der erreichten Diversität dar. Ein blauer Balken zeigt an, dass die Performanz bzw. die Diversität des Ensembles mit der optimierten Pipeline höher ist. Grüne Balken hingegen weisen auf eine höhere Performanz bzw. Diversität des Random Forest Ensembles hin.

Ergebnisse, welche von diesem Trend abweichen, finden sich in den Datensätze 201, 207 und 301. Hier weist das eine Ensemble eine höhere Balanced Accuracy auf, während das andere Ensemble eine höhere Diversität erreichte. Für den Datensatz 207 erreicht die optimierte Pipeline eine höhere Balanced Accuracy von 0.005 %-Punkten. Das Random Forst Ensemble erreichte hingegen eine höhere Diversität, diese ist um 0.001 %-Punkte höher und somit vernachlässigbar. Bei den Datensätzen 201 und 301 erzielt das Random Forest Ensemble eine um mindestens 0.008 %-Punkte höhere Balanced Accuracy. Im Gegensatz dazu erreicht das Ensemble mit der optimierten Pipeline eine um mindestens 0.01%-Punkte höhere Diversität. Aus diesem Ergebnis kann geschlussfolgert werden, dass der erarbeitete Ansatz zur Erzeugung von impliziter Diversität (vgl. Abschnitt 4.1) mittels Datenvorverarbeitungsmethoden erfolgsversprechend ist.

Im Vergleich zu AdaBoost erreicht das optimierte Ensemble für alle Datensätze eine höhere Diversität, siehe Abbildung 6.14. Die Ergebnisse in Abbildung 6.14 sind analog zu den Ergebnissen in Abbildung 6.13 dargestellt, jedoch wird hier AdaBoost statt Random Forest als Vergleichsbaseline betrachtet. Die geringsten Unterschiede in der Diversität beider Ansätze treten bei den Datensätzen 101, 103, 201, 202, 301 auf. Hier beträgt die Differenz der Diversität unter 10 %-Punkte jedoch mindestens 5 %-Punkte. Diese Datensätze zeichnen sich durch eine geringe Menge an Daten und/ oder Hochdimensionale Daten aus. Daraus lässt sich ableiten, dass diese beiden Datencharakteristiken die Leistungsfähigkeit und Diversität von AdaBoost weniger negativ beeinflussen als die Datencharakteristiken Klassenrauschen und Klassenüberschneidung mit Klassenungleichgewicht. Diese Annahme wird durch die Ergebnisse der Datensätze 102, 104, 105, 203, 207, 208 und 307 gestützt, bei denen die optimierte Pipeline eine um mehr als 40 %-Punkte höhere Diversität als AdaBoost erreicht. Alle Datensätze enthalten die Datencharakteristik Klassenrauschen und/oder

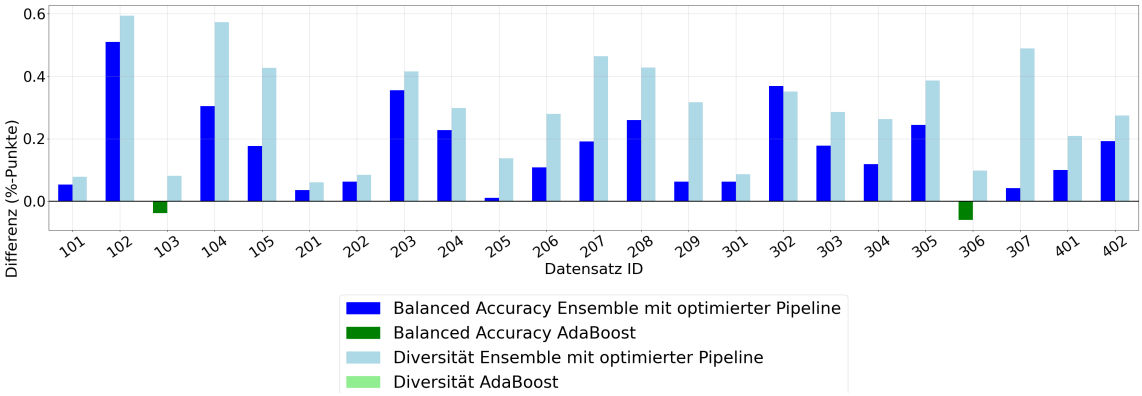

Abbildung 6.14: Darstellung der Performanz- und Diversitätsdifferenz für die Ensembles mit optimierter Pipeline und AdaBoost. Die dunkle Farbe gibt hierbei die Performancedifferenz an und die hellere Farbe die Diversitätsdifferenz. Zeigen die Balken nach oben und sind blau, so schneidet die optimierte Pipeline besser ab.

Klassenüberschneidung mit Klassenungleichgewicht. Diese Datencharakteristiken entsprechen Bedingungen, unter denen AdaBoost tendenziell eine schlechtere Performanz und somit eine geringe Diversität aufweist [FGE14].

6.5 Weitere Analysen

In Abschnitt 6.5.1 wird zunächst der Einfluss der Datencharakteristiken auf die Klassifikationsgenauigkeit und Diversität der Ensembles analysiert. Die betrachteten Ensembles sind mithilfe des erarbeiteten Konzepts erstellt. Anschließend erfolgt in Abschnitt 6.5.2 eine Analyse über die ausgewählten Vorverarbeitungsmethoden.

6.5.1 Einfluss der Datencharakteristiken auf die Klassifikationsgenauigkeit

Im Folgenden wird der Einfluss der unterschiedlichen Datencharakteristiken auf die Klassifikationsgenauigkeit sowie Diversität der Ensembles untersucht, welche auf Basis des erarbeiteten Konzepts erstellt werden. Für den Vergleich der Klassifikationsgenauigkeit wird die Balanced Accuracy herangezogen und für die Bewertung der Diversität wird die normalisierte Doppelfehlerrate verwendet. Die Analyse umfasst den Vergleich der durchschnittlichen Klassifikationsleistung über alle 23 Datensätze hinweg mit der durchschnittlichen erreichten Balanced Accuracy für die verschiedenen Datencharakteristiken. Hierbei wird für jede Datencharakteristik die durchschnittliche Balanced Accurcay aller Datensätze berechnet, die mindestens diese Datencharakteristik enthalten. Dies bedeutet, dass Datensätze, die mehrere Datencharakteristiken enthalten, in die Berechnung aller relevanten Charakteristiken einbezogen sind. Die Ergebnisse werden in Abbildung 6.15 und Tabelle 6.7 dargestellt. Bei der Abbildung 6.15 stellen die dunkelblauen Balken die Balanced Accuracy und die hellblauen Balken die Diversität dar. Die ersten beiden Balken geben die durchschnittliche Balanced Accuracy und Diversität über alle Datensätze hinweg an, während die übrigen Balken die Ergebnisse nach spezifischer Datencharakteristik zeigen.

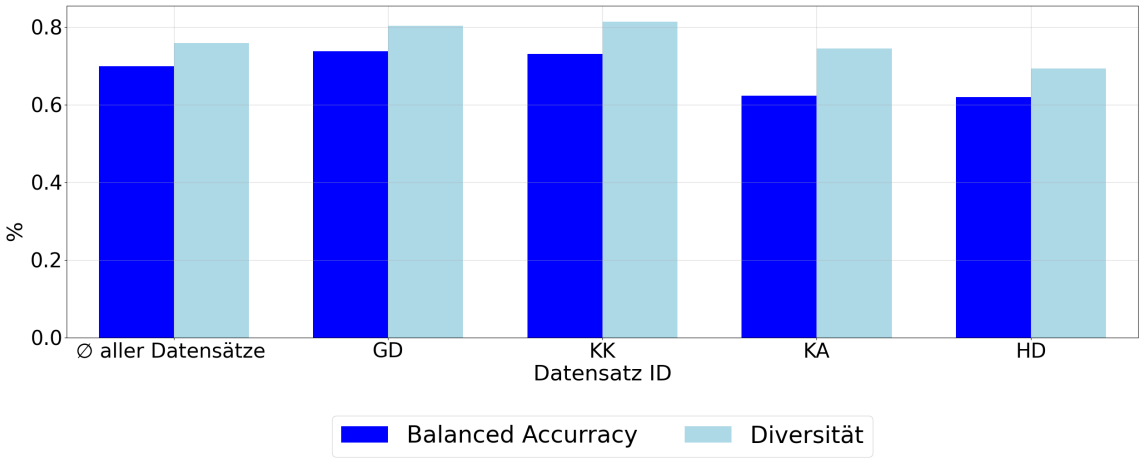

Abbildung 6.15: Darstellung der durchschnittlich erreichten Performanz und Diversität über alle Datensätze, sowie der erreichten durchschnittlichen Performanz und Diversität für die verschiedenen Datencharakteristiken. Die dargestellten Datencharakteristiken sind: *Geringe Datenmenge (GD)*, *Hochdimensionale Daten (HD)*, *Klassenrauschen (KA)* und *Klassenüberschneidung mit Klassenungleichgewicht (KK)*

Datencharakteristik	**Balanced Accuracy**	**Diversität**
Ø aller Datensätze	0.699	0.759
Geringe Menge an Daten	0.738	0.803
Klassenüberschneidung mit Klassenungleichgewicht	0.731	0.814
Klassenrauschen	0.623	0.745
Hochdimensionale Daten	0.620	0.693

Tabelle 6.7: Die Tabelle zeigt die durchschnittlich erreichte Performanz und Diversität über alle Datensätze, sowie die erreichte durchschnittliche Performanz und Diversität für die verschiedenen Datencharakteristiken.

Die durchschnittliche Performanz über alle Datensätze beträgt 0.699 %. Weist ein Datensatz mindestens eine der beiden Datencharakteristiken Hochdimensionale Daten oder Klassenrauschen auf, so fällt die durchschnittliche Balanced Accuracy für diese Datensätze unterdurchschnittlich aus. Die erreichte Performanz für Datensätze mit Klassenrauschen ist 0.623 % und für hochdimensionale Datesätze 0.62%. Es lässt sich ableiten, dass die beiden Datencharakteristiken für die vorliegenden Datensätze die Klassifikationsleistung des Ensembles am negativsten beeinflussen. Das Entfernen von Klassenrauschen gilt als eine der anspruchsvollsten Aufgaben im Bereich des maschinellen Lernens [VK06; ZW04]. Dies zeigt sich auch in den Ergebnissen der Evaluierungsdatensätze. Der Grund für die reduzierte Performanz bei hochdimensionalen Datensätzen lässt sich darauf zurückführen, dass zu wenige Klassifikatoren während der Klassifikatoroptimierung erstellt werden. Diese Datencharakteristik tritt in insgesamt zehn Datensätzen auf, von denen sieben Datensätze (103, 205, 206, 207, 306, 307, 401), aufgrund der Kombination von vorhandenen Dateninstanzen und Merkmalen als stark hochdimensional gelten. Für diese sieben Datensätze wird das Modellerstellungsbudget für die Klassifikatoroptimierung erhöht. Die durchschnittliche Balanced Accuracy

für diese Datensätze beträgt lediglich 0.55 %, während sie bei den drei weniger hochdimensionalen Datensätzen 0.773 % beträgt. Eine Analyse der erstellten Anzahl an Klassifikatoren während der Klassifikatoroptimierung zeigt, dass für die drei Datensätze, die weniger hochdimensional sind, im Schnitt 100 Klassifikatoren generiert werden, während für die restlichen sieben Datensätze lediglich 15 Klassifikatoren.

Im Gegensatz dazu zeigt sich, dass Datensätze mit den Datencharakteristiken geringe Menge an Daten oder Klassenüberschneidung mit Klassenungleichgewicht eine überdurchschnittliche Performanz aufweisen. Liegt eine geringe Menge an Daten vor, beträgt die durchschnittliche Balanced Accuracy 0.738 %. Für Datensätze, die eine Klassenüberschneidung mit Klassenungleichgewicht aufweisen, beträgt die durchschnittliche Balanced Accuracy 0.731 %. Obwohl das Vorliegen einer geringen Datenmenge in dem *AutoML-Ensemble* Framework zur Überanpassung des Ensembles führt (vgl. Abschnitt 6.3.1), beeinflusst diese Datencharakteristik die Performanz der Ensembles weniger negativ als Klassenrauschen oder Hochdimensionale Daten.

Die Datencharakteristik Klassenüberschneidung mit Klassenungleichgewicht gilt als eine der schädlichsten Datencharakteristiken [FGG+18; GMS08]. Sie mindert die Klassifikationsgenauigkeit für die vorliegenden Datensätze jedoch am geringsten. Diese Beobachtung unterstützt die Aussage, dass die Problematik der Klassenüberschneidung mit Klassenungleichgewicht mit steigender Dimensionalität abgeschwächt wird [SAJ+22].

Bei der Betrachtung der Diversität in Abhängigkeit von den Datencharakteristiken zeigt sich, dass diese eng mit der Klassifikationsleistung korreliert. Datensätze, welche Klassenüberschneidung mit Klassenungleichgewicht oder einer geringen Menge an Daten aufweisen, erreichen die höchste Diversität. Diese beträgt für Datensätze, welche Klassenüberschneidung mit Klassenungleichgewicht enthalten, 0.814 % und für Datensätze mit einer geringen Menge an Daten 0.803 %. Im Gegensatz dazu ist die Diversität bei Datensätzen mit Klassenrauschen oder bei hochdimensionalen Datensätzen geringer. Die Diversität für Datensätze mit Klassenrauschen beträgt 0.745 %, die für hochdimensionale Datensätze erreicht 0.693 %. Die Diversität der hochdimensionalen Datensätze ist um 0.05 %-Punkte geringer als die für Klassenrauschen. Das verdeutlicht, dass die geringe Anzahl an generierten Klassifikatoren während der Klassifikatoroptimierung die Vielfalt potenzieller Basis-Klassifikatoren begrenzt. Dadurch wird die Erstellung eines diversen Ensembles erschwert. Dieses Ergebnis unterstreicht den Zusammenhang der Diversität und Klassifikationsgenauigkeit von Ensembles und unterstützt die Annahme Kunchevas (vgl. Abschnitt 6.4).

6.5.2 Wahl der Vorverarbeitungsmethoden

In diesem Teil der Evaluation wird untersucht, ob sich bei der Auswahl der Vorverarbeitungsmethoden Auffälligkeiten zeigen. Zunächst wird die Auswahl der Vorverarbeitungsmethoden über alle Datensätze hinweg betrachtet. Das Ergebnis kann der Abbildung 6.16 entnommen werden. Diese zeigt für jeden Vorverarbeitungsschritt auf, mit welcher prozentualer Häufigkeit eine Methode für einen Vorverarbeitungsschritt ausgewählt wird.

Bei den Rauschunterdrückungsverfahren dominiert der Ensemble-Filter, dieser wurde bei 78% der Datensätze mit Klassenrauschen ausgewählt. Auffällig ist zudem, dass bei den Methoden zur Merkmalsauswahl, der Dimensionsreduktion sowie Überabtastung, die Methoden, welche die höchste Laufzeit aufweisen (MRMR, Isomap und ADASYN) am seltensten ausgewählt werden. Dies

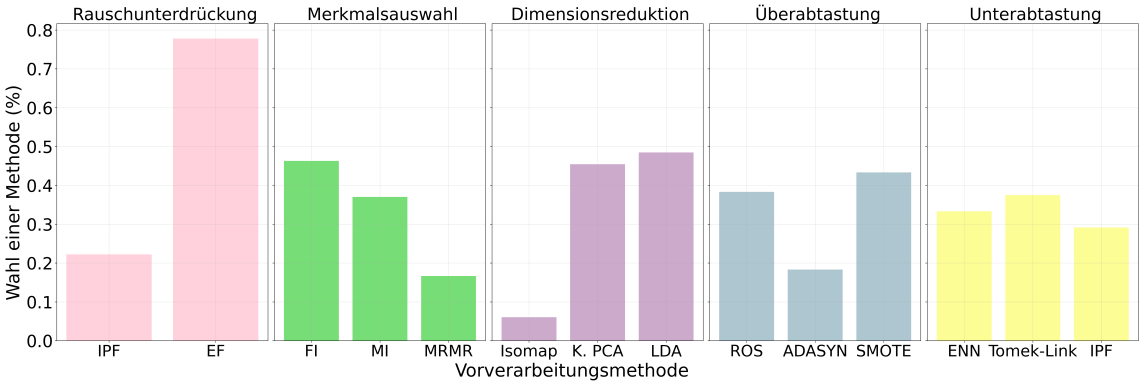

Abbildung 6.16: Vergleich der Häufigkeit, mit der jede Datenvorverarbeitungsmethode in ihrem Datenvorverarbeitungsschritt ausgewählt wird. Die Vorverarbeitungsmethoden sind wie folgt abgekürzt: Ensemble-Filter (EF), Feature Importance (FI), Mutual Information (MI), Kernel PCA (K. PCA), Random Oversampling (ROS).

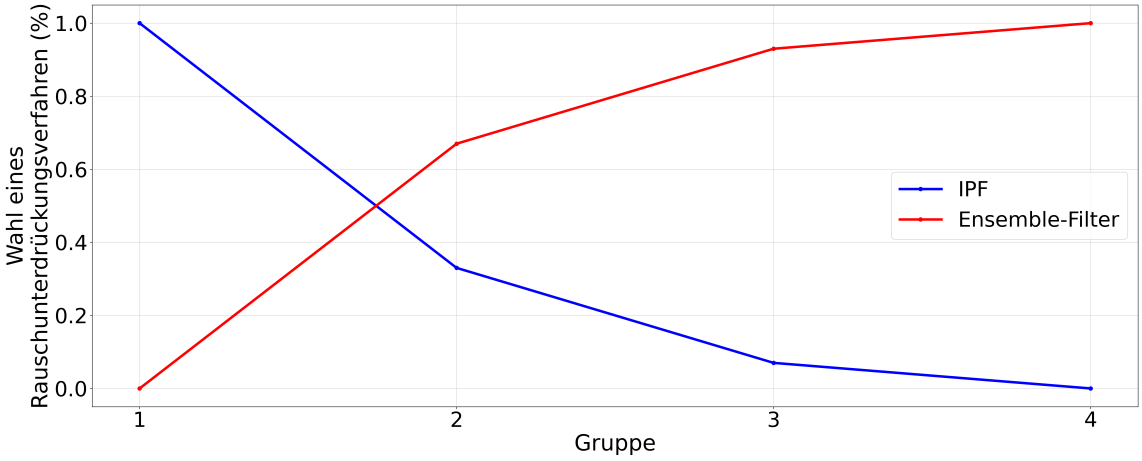

Abbildung 6.17: Häufigkeitsanteil, wie oft jedes Rauschunterdrückungsverfahren in einer Gruppe gewählt wurde.

deutet darauf hin, dass die Laufzeit für die Auswahl der Vorverarbeitungsmethoden von Bedeutung ist. Diese Annahme ist schlüssig, da Vorverarbeitungsmethoden und ihre Modelle verworfen werden, wenn diese eine zu lange Laufzeit aufweisen.

Eine Analyse der Verteilung der ausgewählten Vorverarbeitungsmethoden nach den Gruppen zeigt, dass mit zunehmender Anzahl an Datencharakteristika bei den Rauschunterdrückungsverfahren der Ensemble-Filter häufiger eingesetzt wird, siehe Abbildung 6.17. Dies lässt sich darauf zurückführen, dass der IPF-Filter bei Datensätzen mit Klassenüberschneidung und Klassenungleichgewicht nicht verwendet wird. Der Grund hierfür liegt darin, dass das Anwenden von IPF bei dieser Datencharakteristik zu einer Verschlechterung der Klassifikatorperformanz führt, wenn zuvor keine Überabtastung durchgeführt wird [SLSH15].

6.6 Zusammenfassung der Evaluierungsergebnisse

Im Nachfolgenden werden die wichtigsten Erkenntnisse der Evaluierung zusammenfassend dargestellt. Bei dem Vergleich der besten Basis-Klassifikatoren mit den Ensembles mit der optimierten Pipeline erzielt das Ensemble eine niedrigere Klassifikationsleistung für mehr Datensätze. Der Grund für die verminderte Performanz des Ensembles gegenüber dem besten Basis-Klassifikator wurde ermittelt (vgl.Abschnitt 6.3.1). Ebenso können Strategien zur Erhöhung der Klassifikationsleistung für das Ensemble aufgezeigt werden. Diese werden im Nachfolgenden aufgezeigt.

Bei einer größeren Anzahl von Datensätzen erzielen die Ensembles mit der optimierten Vorverarbeitungspipeline im Vergleich zu den Ensembles mit impliziter Diversität und der statischen Pipeline eine höhere Klassifikationsgenauigkeit sowie Diversität. Dies zeigt, dass das Ziel, durch die Anwendung von Vorverarbeitungsmethoden die Diversität innerhalb eines Ensembles zu erhöhen, erreicht wird (vgl. Kapitel 4). Da die Diversität von Ensembles eng im Zusammenhang mit deren Klassifikationsleistung steht, ist die Förderung von diversen Ensembles eine sinnvolle Strategie zur Verbesserung der Klassifikationsleistung.

Es konnten drei Faktoren herausgearbeitet werden, aufgrund derer das optimierte Ensemble in bestimmten Fällen schlechter abschneidet als die Baselinemethoden:

- Die Aufteilung der Daten in Trainings-, Validierungs- und Testmenge und die damit verbundene Funktionsweise des *AutoML-Ensemble* Frameworks (vgl. Abschnitt 6.1 und Abschnitt 6.3).
- Die geringe Anzahl der erstellten Klassifikatoren während der Klassifikatoroptimierung.
- Die zu geringe Komplexität des Datensatzes. Werden die Muster eines Datensatzes ausreichend von einem einzelnen Klassifikator erfasst, schneidet dieser im Vergleich zum Ensemble besser ab (vgl. Abschnitt 6.3.1).

Den ersten beiden Herausforderungen kann entgegengewirkt werden. Das Problem der Datenaufteilung kann bei kleinen Datensätzen zu einer Überanpassung der Klassifikation und der Fusionsmethode führen. Durch eine alternative Aufteilung und Verwendung des Datensatzes könnten der Fusionsoptimierung mehr Daten zur Verfügung gestellt werden. Dadurch würde die Wahrscheinlichkeit einer Überanpassung durch die Fusionsmethoden verringert werden. In einer neuen Version des *AutoML-Ensemble* Framework wird bereits eine neue Aufteilung der Daten unterstützt, sodass der Fusionsoptimierung mehr Dateninstanzen zum Trainieren zur Verfügung steht. Da die neue Datenaufteilung bereits im *AutoML-Ensemble* Framework implementiert ist, wurde die optimierte Pipeline mit dieser Aufteilung exemplarisch für einen Durchlauf evaluiert. In Abbildung 6.18 sind die Ergebnisse dargestellt. Die Abbildung zeigt die Performanzdifferenz der verwendeten Version des AutoML-Frameworks gegenüber der neueren Version des AutoML-Frameworks. Die Balken, die lila sind zeigen, dass die neue Version des AutoML-Frameworks eine höhere Performanz für einen Datensatz erreicht hat. Während blaue Balken zeigen, dass die alte Version eine höhere Performanz für den vorliegenden Datensatz erreicht. Für die exemplarische Evaluierung ist eine Verbesserung der Performanz des Ensembles mit der neueren Version des Frameworks zu sehen. Besonders für Datensätze, welche eine geringe Menge an Daten aufweisen. Das zweite Problem beschreibt die begrenzte Anzahl an generierten Klassifikatoren während der Klassifikatorenoptimierung. Dies kann durch eine Erhöhung des Modellerstellungsbudgets behoben

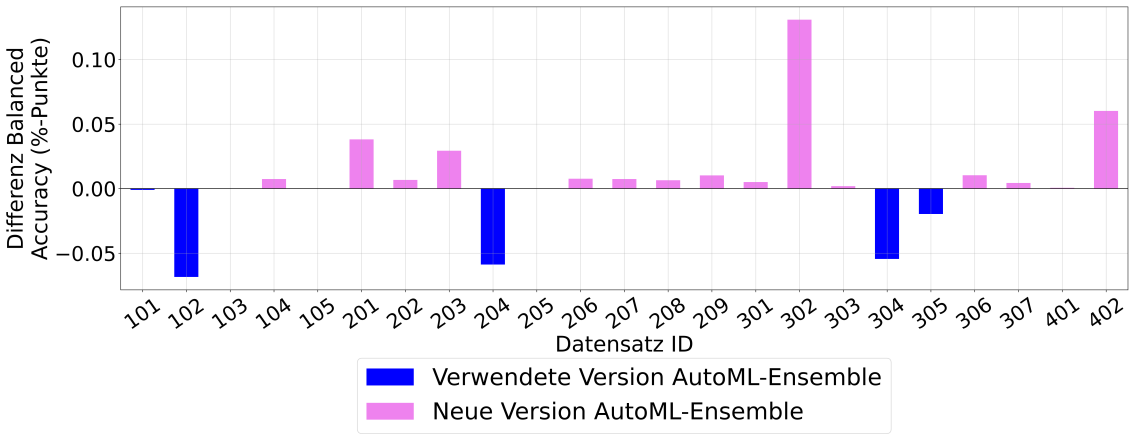

Abbildung 6.18: Performanzdifferenz zwischen der verwendeten und der neuen Version des AutoML-Frameworks: Lila Balken zeigen Datensätze, bei denen die neue Version eine bessere Performanz erzielt hat, während blaue Balken Datensätze darstellen, bei denen die verwendete Version überlegen war.

werden. Mit diesen beiden Anpassungen sollte sich sowohl die Performanz als auch die Diversität der optimierten Pipeline verbessern. Dadurch kann die Überlegenheit dieses Ansatzes gegenüber den anderen Baselinemethoden weiter gefestigt werden.

Abschließend kann festgehalten werden, dass das vorgestellte Konzept sich hervorragend für den Einsatz in AutoML-Frameworks eignet. Im Gegensatz zu einer statischen Pipeline ermöglicht das erarbeitete Konzept eine automatische Konfiguration der Vorverarbeitungspipelines, welche spezifisch die Schwierigkeiten der Datencharakteristiken eines Datensatzes adressiert. Der Anwender muss daher keine Einstellungen mehr vornehmen.

7 Zusammenfassung

Maschinelles Lernen wird mittlerweile in verschiedenen Bereichen eingesetzt. Unter anderem findet es Anwendung in der Medizin, zur Unterstützung von Diagnoseerstellung, sowie in der Industrie zur Erkennung fehlerhafter Produkte im Produktionsprozess. Die für das maschinelle Lernen eingesetzten Algorithmen basieren häufig auf Annahmen über die Daten, die in realen Szenarien oft nicht erfüllt werden. So wird typischerweise angenommen, dass die Daten keinen fehlerhaften Wert aufweisen und dass ausreichend Daten zur Verfügung stehen, um die Muster in einem Datensatz zu erkennen. Liegen solche Herausforderungen in einem Datensatz vor, können einzelne Klassifikatoren oft keine zufriedenstellende Leistung erzielen.

Ein Ansatz, dieser Herausforderung entgegenzuwirken, besteht in der Verwendung von Ensembles. Diese sind dafür bekannt, komplexe Datencharakteristiken handhaben zu können. Ein wichtiges Konzept bei Ensembles stellt die Diversität dar. Für diese wird angenommen, dass eine Erhöhung der Diversität mit einer Erhöhung der Klassifikationsgenauigkeit korreliert.

In dieser Arbeit wurde ein Konzept entwickelt, das durch die gezielte Anwendung von Vorverarbeitungsmethoden in Abhängigkeit der Datencharakteristiken eines Datensatzes die Diversität und die Klassifikationsgenauigkeit eines Ensembles steigert. Das Konzept wurde für die Datencharakteristiken Klassenrauschen, eine geringe Menge an Daten, Hochdimensionale Daten sowie Klassenüberschneidung mit Klassenungleichgewicht entwickelt. Dabei ist das Konzept flexibel einsetzbar und kann einzelne oder in Kombination auftretende Datencharakteristiken in einem Datensatz adressieren. Zu diesem Zweck wurden Anforderungen für die Auswahl geeigneter Vorverarbeitungsmethoden erarbeitet. Ein wichtiger Aspekt dabei ist, dass die prototypische Implementierung des Konzepts in einem AutoML-Framework erfolgt. Eine solche Implementierung hat Einfluss auf die Auswahl geeigneter Vorverarbeitungsmethoden. Die Integration des Konzepts in einem AutoML-Framework hat zudem den Vorteil, dass die Auswahl und Optimierung der Vorverarbeitungsmethoden und Klassifikatoren automatisiert durchgeführt werden. Das vorgestellte Konzept wurde prototypisch in der Software *AutoML-Ensemble* implementiert.

Zur Evaluierung des entwickelten Ansatzes wurden Evaluationsläufe für 23 Datensätze durchgeführt, die sowohl Echtwelt- als auch synthetische Datensätze umfassen. Die Datensätze weisen unterschiedliche Datencharakteristiken sowie die möglichen Kombinationen dieser auf. Zuerst wurde untersucht, ob die erstellten Ensembles eine höhere Klassifikationsleistung als die einzelnen Basis-Klassifikatoren erzielten. Dabei zeigte sich, dass für 43 % der Datensätze das Ensemble ein besseres Ergebnis erzielte als die besten Basis-Klassifikatoren. Zur weiteren Bewertung wurden zusätzliche Vergleiche zu Baselinemethoden durchgeführt. Hierbei wurde zum einen eine statische Vorverarbeitungspipeline und zum anderen etablierte Ensemblemethoden wie AdaBoost und Random Forest verwendet. Das erarbeitete Konzept erzielt in 60.9 % der Evaluierungsläufe eine bessere oder gleiche Leistung wie die statische Pipeline und übertraf AdaBoost in 91.3 % und Random Forest in 78.3 % der evaluierten Datensätze. Die Analyse der Ergebnisse identifizierte drei Hauptfaktoren, die die Performanz der optimierten Pipeline beeinträchtigen. Erstens: es wird

eine geringe Vorhersagegenauigkeit bei Datensätzen mit einer geringen Datenmenge erzielt. Dies ist darauf zurückzuführen, dass das verwendete *AutoML-Ensemble* Framework überangepasste Ensembles aufgrund der Einteilung des Datensatzes in Trainings-, Validierungs- und Testmengen erstellt. Zweitens: eine zu geringe Anzahl an generierten Klassifikatoren in der Klassifikatoroptimierungsphase wirkt sich negativ auf die Ensembleperformanz auf. Drittens: es gibt Datensätze, bei denen das Verwenden eines Ensembles nicht notwendig ist, da einzelne Klassifikatoren und die dazugehörige Datenvorverarbeitungspipeline die Muster des Datensatzes bereits ausreichend erfassen.

Die Analyse der Datencharakteristiken hat gezeigt, dass Hochdimensionale Daten und Klassenrauschen die Klassifikationsleistung von Ensembles für die vorliegenden Datensätze am stärksten negativ beeinflussen. Im Gegensatz dazu erwiesen sich die Datencharakteristiken der Klassenüberschneidung mit Klassenungleichgewicht und eine geringe Menge an Daten als weniger schädlich für die Performanz. Zudem wurde festgestellt, dass sich die Klassifikationsleistung mit der zunehmenden Anzahl an vorliegenden Datencharakteristiken verschlechtert.

Abschließend lässt sich festhalten, dass das erarbeitete Konzept im Durchschnitt zu einer Verbesserung der Performanz von Klassifikationsensembles beiträgt. Die Ergebnisse der durchgeführten Evaluation belegen, dass durch die gezielte Anwendung von Vorverarbeitungsmethoden sowohl die Erhöhung der Diversität als auch die Erhöhung der Klassifikationsgenauigkeit in Ensembles möglich ist. Dies unterstreicht, dass das Ziel - durch die gezielte Anwendung von Vorverarbeitungsmethoden die Diversität innerhalb eines Ensembles zu steigern - erreicht wurde.

Der Beitrag dieser Arbeit umfasst:

- Die Entwicklung eines Konzepts zur Steigerung der Diversität und Genauigkeit in Klassifikationsensembles durch die gezielte Anwendung von Vorverarbeitungsmethoden.
- Prototypische Implementierung in einem AutoML-Framework.
- Bestätigung der Annahme, dass ein positiver Zusammenhang zwischen Diversität und Genauigkeit in Klassifikations-Ensembles besteht.
- Eine umfassende Evaluation des erarbeiteten Ansatzes anhand von 23 Echtwelt- und synthetischen Datensätzen.
- Die Untersuchung des Einflusses der Hyperparameteroptimierung von Vorverarbeitungsmethoden auf die Klassifikationsgenauigkeit und Diversität von Ensembles.

8 Ausblick

Während der Evaluation des Konzeptes können zwei Kriterien identifiziert werden, unter denen die Ensembles der optimierten Pipeline nicht durchgehend eine bessere Leistung als die Ensembles der statischen Pipeline erzielen. Diese Schwachstellen gilt es durch gezielte Maßnahmen zu adressieren. Eine Ursache besteht in der Überanpassung des Ensembles bei einer geringen Datenmenge während der Klassifikator- und Fusionsoptimierungsphase. Eine verbesserte Aufteilung der Daten in diesen Phasen kann dieser Problematik entgegenwirken, wie in Abschnitt 6.6 bereits exemplarisch gezeigt wird.

Die zweite Ursache, welche die Leistung der optimierten Pipeline negativ beeinflusst, ist, wenn zu wenige Klassifikatoren während der Klassifikatoroptimierungsphase erstellt werden. Dies kann dazu führen, dass nicht genügend Klassifikatoren generiert werden, die eine hohe Klassifikationsleistung aufweisen. Zum Abschwächen dieser Problematik gibt es mehrere Ansatzmöglichkeiten.

Derzeit werden die Vorverarbeitungsmethoden gemeinsam mit den Klassifikatoren optimiert, wodurch die Laufzeit für das Training der Klassifikatoren erheblich erhöht wird. Ein Lösungsvorschlag hier ist, vor der eigentlichen Klassifikatioroptimierung bereits mehrere Vorverarbeitungspipelines für den jeweiligen Datensatz zu generieren. Aus diesen Vorverarbeitungspipelines können die Klassifikatoren dann wählen. Auf diese Weise entfällt die gemeinsame Optimierung der Vorverarbeitungsmethoden während der Klassifikatoroptimierung, so dass mehr Zeit darauf verwendet werden kann, geeignete Klassifikatoren zu generieren. Die Erstellung mehrerer Vorverarbeitungspipelines könnte beispielsweise durch einen eigenen Optimierer oder durch den Einsatz von Large Language Models wie Code-Llama [RGG+23] erfolgen.

Ein alternativer Ansatz zur Verkürzung der Laufzeit der Klassifikatoroptimierung ist das Einsetzen von Meta-Learning [Van19]. Dieses wird beispielsweise von dem Framework Auto-Sklearn [FKE+15] für einen Warm-Start der Modelloptimierung eingesetzt. Hierbei werden Hyperparameterkonfigurationen vorgeschlagen, die bei ähnlichen Klassifikationsproblemen bereits eine hohe Performanz gezeigt haben. Dies ermöglicht, die Modelloptimierung effizient zu initialisieren, da zu erwarten ist, dass diese Konfiguration bei einem ähnlichen Problem ebenfalls zu einer hohen Performanz führt. Durch den Warm-Start der Modelloptimierung werden schnellere Hyperparameterkonfigurationen für Klassifikatoren gefunden, welche zu einer guten Klassifikationsleistung führen.

Weitere mögliche Aspekte, die im Zusammenhang mit dem erarbeiteten Konzept untersucht werden sollten, sind die automatische Erkennung der Datencharakteristiken eines Datensatzes, sodass diese nicht mehr manuell durch den Nutzer festgelegt werden müssen. Dies entspräche der Idee von AutoML- Frameworks, maschinelles Lernen auch für Anwender ohne Expertenwissen im maschinellen Lernen zugänglich zu machen.

Ein interessanter Ansatz für zukünftige Arbeiten ist die Erweiterung des erarbeiteten Konzepts durch das Einbinden weiterer komplexer Datencharakteristiken sowie die Untersuchung ihrer Wechselwirkung. Insbesondere die Analyse des Einflusses von Datenvorverarbeitungsmethoden auf Datensätze, die mehrere Datencharakteristiken aufweisen, ist von Interesse. Hierbei sollte untersucht werden, wie verschiedene Vorverarbeitungsmethoden kombiniert und angepasst werden müssen, um die Klassifikationsleistung unter solchen Bedingungen zu optimieren. Ein weiterer Aspekt ist die Untersuchung der Anwendungsreihenfolge von den Datenvorverarbeitungsmethoden. Die in dieser Arbeit genutzte Reihenfolge basiert auf der Arbeit von Garcia et al. [GLH16], welche die optimale Anwendungsreihenfolge der Datenvorverarbeitungsmethoden lediglich für einen Datensatz untersucht. Daher wäre eine umfassende Evaluierung zur Anwendungsreihenfolge von Datenvorverarbeitungsmethoden wünschenswert.

Literaturverzeichnis

[Abu13] H. Abusamra. „A comparative study of feature selection and classification methods for gene expression data of glioma“. In: *Procedia Computer Science* 23 (2013), S. 5–14 (zitiert auf S. 23, 45–47).

[AKV19] S.-A. N. Alexandropoulos, S. B. Kotsiantis, M. N. Vrahatis. „Data preprocessing in predictive data mining“. In: *The Knowledge Engineering Review* 34 (2019), e1 (zitiert auf S. 26, 29).

[ASS21] F. Anowar, S. Sadaoui, B. Selim. „Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)“. In: *Computer Science Review* 40 (2021), S. 100378. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2021.100378. URL: https://www.sciencedirect.com/science/article/pii/S1574013721000186 (zitiert auf S. 52, 53).

[BCSV08] P. Brazdil, C. G. Carrier, C. Soares, R. Vilalta. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008 (zitiert auf S. 22).

[BF99] C. E. Brodley, M. A. Friedl. „Identifying mislabeled training data“. In: *Journal of artificial intelligence research* 11 (1999), S. 131–167 (zitiert auf S. 26, 50).

[BG07] I. Borg, P. J. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2007 (zitiert auf S. 32).

[BIYM12] S. Barua, M. M. Islam, X. Yao, K. Murase. „MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning“. In: *IEEE Transactions on knowledge and data engineering* 26.2 (2012), S. 405–425 (zitiert auf S. 54).

[BN06] C. M. Bishop, N. M. Nasrabadi. *Pattern recognition and machine learning*. Bd. 4. 4. Springer, 2006 (zitiert auf S. 22).

[BPM04] G. E. Batista, R. C. Prati, M. C. Monard. „A study of the behavior of several methods for balancing machine learning training data“. In: *ACM SIGKDD explorations newsletter* 6.1 (2004), S. 20–29 (zitiert auf S. 32, 54).

[Bre01] L. Breiman. „Random forests“. In: *Machine learning* 45 (2001), S. 5–32 (zitiert auf S. 30, 35, 42, 52, 53).

[Bre96] L. Breiman. „Bagging predictors“. In: *Machine learning* 24 (1996), S. 123–140 (zitiert auf S. 41).

[Bro04] G. Brown. „Diversity in neural network ensembles“. Diss. Citeseer, 2004 (zitiert auf S. 36).

[Bru23] A. Bruhn. *Computer Vision*. Vorlesung, Universität Stuttgart, Januar 2023. 2022/2023 (zitiert auf S. 31).

[Bur19] A. Burkov. *The Hundred-page Machine Learning Book*. Andriy Burkov, 2019. ISBN: 9781777005474. URL: https://books.google.de/books?id=Gc5WzwEACAAJ (zitiert auf S. 21–23, 25, 28, 36).

[BWHY05] G. Brown, J. Wyatt, R. Harris, X. Yao. „Diversity creation methods: a survey and categorisation“. In: *Information fusion* 6.1 (2005), S. 5–20 (zitiert auf S. 35, 37).

[CBHK02] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer. „SMOTE: synthetic minority over-sampling technique“. In: *Journal of artificial intelligence research* 16 (2002), S. 321–357 (zitiert auf S. 32, 54).

[CC00] T. F. Cox, M. A. Cox. *Multidimensional scaling*. CRC press, 2000 (zitiert auf S. 53).

[DDC18] S. Das, S. Datta, B. B. Chaudhuri. „Handling data irregularities in classification: Foundations, trends, and future challenges“. In: *Pattern Recognition* 81 (2018), S. 674–693 (zitiert auf S. 19, 25).

[DGSH15] J. Derrac, S. Garcia, L. Sanchez, F. Herrera. „Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework“. In: *J. Mult. Valued Logic Soft Comput* 17 (2015), S. 255–287 (zitiert auf S. 52, 72).

[DHG01] R. Duda, P. Hart, D. G.Stork. „Pattern Classification“. In: NJ: Wiley-Interscience, Jan. 2001. ISBN: 0-471-05669-3 (zitiert auf S. 52).

[Die00] T. G. Dietterich. „Ensemble methods in machine learning“. In: *International workshop on multiple classifier systems*. Springer. 2000, S. 1–15 (zitiert auf S. 19, 34).

[DKC+03] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, A. N. Poo. „Multi-category classification by soft-max combination of binary classifiers“. In: *Multiple Classifier Systems: 4th International Workshop, MCS 2003 Guildford, UK, June 11–13, 2003 Proceedings 4*. Springer. 2003, S. 125–134 (zitiert auf S. 36).

[Dun89] G. H. Dunteman. *Principal components analysis*. Bd. 69. Sage, 1989 (zitiert auf S. 31, 53).

[DWX+24] Q. Dai, L.-h. Wang, K.-l. Xu, T. Du, L.-f. Chen. „Class-overlap detection based on heterogeneous clustering ensemble for multi-class imbalance problem“. In: *Expert Systems with Applications* 255 (2024), S. 124558 (zitiert auf S. 70).

[EMS+20] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola. „Autogluon-tabular: Robust and accurate automl for structured data“. In: *arXiv preprint arXiv:2003.06505* (2020) (zitiert auf S. 44).

[FCW+21] C. Fan, M. Chen, X. Wang, J. Wang, B. Huang. „A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data“. In: *Frontiers in energy research* 9 (2021), S. 652801 (zitiert auf S. 26, 29, 31, 32).

[FF12] A. J. Ferreira, M. A. Figueiredo. „Boosting algorithms: A review of methods, theory, and applications“. In: *Ensemble machine learning: Methods and applications* (2012), S. 35–85 (zitiert auf S. 42).

[FGE14] K. Fawagreh, M. M. Gaber, E. Elyan. „Random forests: from early developments to recent advancements“. In: *Systems Science & Control Engineering: An Open Access Journal* 2.1 (2014), S. 602–609 (zitiert auf S. 42, 94).

[FGG+18] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, F. Herrera, A. Fernández, S. García, M. Galar, R. C. Prati et al. „Data intrinsic characteristics“. In: *Learning from Imbalanced Data Sets* (2018), S. 253–277 (zitiert auf S. 23, 25, 28, 32, 50, 54, 55, 70, 88, 96).

[FH89] E. Fix, J. L. Hodges. „Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties“. In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), S. 238–247. ISSN: 03067734, 17515823. URL: http://www.jstor.org/stable/1403797 (besucht am 16. 09. 2024) (zitiert auf S. 26, 28).

[Fis36] R. A. Fisher. „The use of multiple measurements in taxonomic problems“. In: *Annals of eugenics* 7.2 (1936), S. 179–188 (zitiert auf S. 26, 31, 53).

[FKE+15] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter. „Efficient and robust automated machine learning“. In: *Advances in neural information processing systems* 28 (2015) (zitiert auf S. 43, 103).

[FS97] Y. Freund, R. E. Schapire. „A decision-theoretic generalization of on-line learning and an application to boosting“. In: *Journal of computer and system sciences* 55.1 (1997), S. 119–139 (zitiert auf S. 36, 42).

[GEW06] P. Geurts, D. Ernst, L. Wehenkel. „Extremely randomized trees“. In: *Machine learning* 63 (2006), S. 3–42 (zitiert auf S. 42).

[GLH15] S. García, J. Luengo, F. Herrera. *Data preprocessing in data mining*. Bd. 72. Springer, 2015 (zitiert auf S. 23, 24, 26, 28).

[GLH16] S. García, J. Luengo, F. Herrera. „Tutorial on practical tips of the most influential data preprocessing algorithms in data mining“. In: *Knowledge-Based Systems* 98 (2016), S. 1–29 (zitiert auf S. 23, 26, 28, 29, 31, 52, 55, 57, 104).

[GMS08] V. García, R. A. Mollineda, J. S. Sánchez. „On the k-NN performance in a challenging scenario of imbalance and overlapping“. In: *Pattern Analysis and Applications* 11 (2008), S. 269–280 (zitiert auf S. 25, 50, 96).

[GR01] G. Giacinto, F. Roli. „An approach to the automatic design of multiple classifier systems“. In: *Pattern recognition letters* 22.1 (2001), S. 25–33 (zitiert auf S. 60, 73).

[HBGL08] H. He, Y. Bai, E. A. Garcia, S. Li. „ADASYN: Adaptive synthetic sampling approach for imbalanced learning“. In: *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee. 2008, S. 1322–1328 (zitiert auf S. 32, 33, 54).

[HM13] H. He, Y. Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. 1st. Wiley-IEEE Press, 2013. ISBN: 1118074629 (zitiert auf S. 28, 55).

[Ho95] T. K. Ho. „Random decision forests“. In: *Proceedings of 3rd international conference on document analysis and recognition*. Bd. 1. IEEE. 1995, S. 278–282 (zitiert auf S. 35, 42).

[HRM19] V. Hirsch, P. Reimann, B. Mitschang. „Data-driven fault diagnosis in end-of-line testing of complex products“. In: *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2019, S. 492–503 (zitiert auf S. 19, 50).

[HTFF09] T. Hastie, R. Tibshirani, J. H. Friedman, J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Bd. 2. Springer, 2009 (zitiert auf S. 23).

[HZC21] X. He, K. Zhao, X. Chu. „AutoML: A survey of the state-of-the-art“. In: *Knowledge-based systems* 212 (2021), S. 106622 (zitiert auf S. 38).

[JBB15] A. Jović, K. Brkić, N. Bogunović. „A review of feature selection methods with applications“. In: *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. Ieee. 2015, S. 1200–1205 (zitiert auf S. 29, 30).

[JM06] H. Jiawei, K. Micheline. *Data mining: concepts and techniques*. Morgan kaufmann, 2006 (zitiert auf S. 22).

[Kég17] B. Kégl. *How to Build a Data Science Pipeline*. Available at: https://www.kdnuggets.com/2017/07/build-data-science-pipeline.html. 2017. URL: https://www.kdnuggets.com/2017/07/build-data-science-pipeline.html (zitiert auf S. 21).

[KKP06] S. B. Kotsiantis, D. Kanellopoulos, P. E. Pintelas. „Data preprocessing for supervised leaning“. In: *International journal of computer science* 1.2 (2006), S. 111–117 (zitiert auf S. 22, 25).

[KL51] S. Kullback, R. A. Leibler. „On information and sufficiency“. In: *The annals of mathematical statistics* 22.1 (1951), S. 79–86 (zitiert auf S. 30).

[KR07] T. M. Khoshgoftaar, P. Rebours. „Improving software quality prediction by noise filtering techniques“. In: *Journal of Computer Science and Technology* 22 (2007), S. 387–396 (zitiert auf S. 26, 27, 50).

[KR92] K. Kira, L. A. Rendell. „A practical approach to feature selection“. In: *Machine learning proceedings 1992*. Elsevier, 1992, S. 249–256 (zitiert auf S. 52).

[Kun04] L. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004. ISBN: 9780471660255. URL: https://books.google.de/books?id=B4TZtp7X82oC (zitiert auf S. 19, 21, 22, 34–38, 41, 91, 93).

[KW03] L. I. Kuncheva, C. J. Whitaker. „Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy“. In: *Machine learning* 51 (2003), S. 181–207 (zitiert auf S. 36–38).

[KW17] M. Koziarski, M. Woźniak. „CCR: A combined cleaning and resampling algorithm for imbalanced data classification“. In: *International Journal of Applied Mathematics and Computer Science* 27.4 (2017), S. 727–736 (zitiert auf S. 54).

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324 (zitiert auf S. 38).

[LFG+13] V. López, A. Fernández, S. García, V. Palade, F. Herrera. „An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics“. In: *Information sciences* 250 (2013), S. 113–141 (zitiert auf S. 25, 27, 28).

[LGS16] J.-C. Lévesque, C. Gagné, R. Sabourin. „Bayesian hyperparameter optimization for ensemble learning“. In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. UAI'16. Jersey City, New Jersey, USA: AUAI Press, 2016, S. 437–446. ISBN: 9780996643115 (zitiert auf S. 59).

[LP20] E. LeDell, S. Poirier. „H2o automl: Scalable automatic machine learning“. In: *Proceedings of the AutoML Workshop at ICML*. Bd. 2020. ICML San Diego, CA, USA. 2020 (zitiert auf S. 43).

[MJAE22] C. E. Morr, M. Jammal, H. Ali-Hassan, W. EI-Hallak. *Machine Learning for Practical Decision Making*. Techn. Ber. Springer, 2022 (zitiert auf S. 26, 29).

[MLG+17] P. Morales, J. Luengo, L. P. Garcia, A. C. Lorena, A. C. de Carvalho, F. Herrera et al. „The noisefiltersr package: label noise preprocessing in R“. In: *The R Journal* 9.1 (2017), S. 219–228 (zitiert auf S. 50, 52).

[MM05] P. Melville, R. J. Mooney. „Creating diversity in ensembles using artificial data“. In: *Information Fusion* 6.1 (2005), S. 99–111 (zitiert auf S. 35, 43).

[MOAD21] I. D. Mienye, G. Obaido, K. Aruleba, O. A. Dada. „Enhanced prediction of chronic kidney disease using feature selection and boosted classifiers“. In: *International Conference on Intelligent Systems Design and Applications*. Springer. 2021, S. 527–537 (zitiert auf S. 23).

[MS21] I. D. Mienye, Y. Sun. „Effective feature selection for improved prediction of heart disease“. In: *Pan-African Artificial Intelligence and Smart Systems Conference*. Springer. 2021, S. 94–107 (zitiert auf S. 19, 23, 44, 46).

[MS22] I. D. Mienye, Y. Sun. „A survey of ensemble learning: Concepts, algorithms, applications, and prospects“. In: *IEEE Access* 10 (2022), S. 99129–99149 (zitiert auf S. 42).

[MSA+18] M. Mercier, M. S. Santos, P. H. Abreu, C. Soares, J. P. Soares, J. Santos. „Analysing the footprint of classifiers in overlapped and imbalanced contexts“. In: *International symposium on intelligent data analysis*. Springer. 2018, S. 200–212 (zitiert auf S. 25).

[Mur22] K. P. Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022 (zitiert auf S. 30, 52).

[MY19] P. Misra, A. S. Yadav. „Impact of preprocessing methods on healthcare predictions“. In: *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*. 2019 (zitiert auf S. 23, 44, 46).

[PCA+21] J. D. Pascual-Triana, D. Charte, M. Andrés Arroyo, A. Fernández, F. Herrera. „Revisiting data complexity metrics based on morphology for overlap and imbalance: snapshot, new overlap number of balls metrics and singular problems prospect“. In: *Knowledge and Information Systems* 63 (2021), S. 1961–1989 (zitiert auf S. 70).

[Pea92] K. Pearson. „On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling“. In: *Breakthroughs in Statistics: Methodology and Distribution*. Hrsg. von S. Kotz, N. L. Johnson. New York, NY: Springer New York, 1992, S. 11–28. ISBN: 978-1-4612-4380-9. DOI: 10.1007/978-1-4612-4380-9_2. URL: https://doi.org/10.1007/978-1-4612-4380-9_2 (zitiert auf S. 52).

[PLD05] H. Peng, F. Long, C. Ding. „Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy“. In: *IEEE Transactions on pattern analysis and machine intelligence* 27.8 (2005), S. 1226–1238 (zitiert auf S. 30, 53).

[Pol06] R. Polikar. „Ensemble based systems in decision making“. In: *IEEE Circuits and systems magazine* 6.3 (2006), S. 21–45 (zitiert auf S. 19, 34–37, 42, 50, 75, 88).

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830 (zitiert auf S. 54).

[Qui14] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014 (zitiert auf S. 26).

[Ras18] S. Raschka. *STAT479: Machine Learning Lecture Notes*. Department of Statistics, University of Wisconsin-Madison. Fall 2018. 2018. URL: http://stat.wisc.edu/sraschka/teaching/stat479-fs2018/ (zitiert auf S. 52).

[RGG+23] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez et al. „Code llama: Open foundation models for code“. In: *arXiv preprint arXiv:2308.12950* (2023) (zitiert auf S. 103).

[RLM+17] S. Ramírez-Gallego, I. Lastra, D. Martínez-Rego, V. Bolón-Canedo, J. M. Benítez, F. Herrera, A. Alonso-Betanzos. „Fast-mRMR: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data“. In: *International Journal of Intelligent Systems* 32.2 (2017), S. 134–152 (zitiert auf S. 53).

[RRL+20] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, T. Baker. „Analysis of dimensionality reduction techniques on big data“. In: *Ieee Access* 8 (2020), S. 54776–54788 (zitiert auf S. 23, 45, 46).

[RS00] S. T. Roweis, L. K. Saul. „Nonlinear dimensionality reduction by locally linear embedding“. In: *science* 290.5500 (2000), S. 2323–2326 (zitiert auf S. 53).

[SAJ+22] M. S. Santos, P. H. Abreu, N. Japkowicz, A. Fernández, C. Soares, S. Wilk, J. Santos. „On the joint-effect of class imbalance and overlap: a critical review“. In: *Artificial Intelligence Review* 55.8 (2022), S. 6207–6275 (zitiert auf S. 25, 27, 32, 33, 50, 70, 72, 96).

[Sch20] L. A. Schintler. „High Dimensional Data“. In: *Encyclopedia of Big Data*. Hrsg. von L. A. Schintler, C. L. McNeely. Cham: Springer International Publishing, 2020, S. 1–3. ISBN: 978-3-319-32001-4. DOI: 10.1007/978-3-319-32001-4_552-2. URL: https://doi.org/10.1007/978-3-319-32001-4_552-2 (zitiert auf S. 23, 54).

[Sch90] R. E. Schapire. „The strength of weak learnability“. In: *Machine learning* 5 (1990), S. 197–227 (zitiert auf S. 42).

[Sha48] C. E. Shannon. „A mathematical theory of communication“. In: *The Bell system technical journal* 27.3 (1948), S. 379–423 (zitiert auf S. 30).

[SIL07] Y. Saeys, I. Inza, P. Larranaga. „A review of feature selection techniques in bioinformatics“. In: *bioinformatics* 23.19 (2007), S. 2507–2517 (zitiert auf S. 29, 30).

[SLSH15] J. A. Sáez, J. Luengo, J. Stefanowski, F. Herrera. „SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering“. In: *Information Sciences* 291 (2015), S. 184–203 (zitiert auf S. 25, 27, 52, 55, 57, 62, 97).

[SR18] O. Sagi, L. Rokach. „Ensemble learning: A survey“. In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 8.4 (2018), e1249 (zitiert auf S. 42, 43, 75, 88).

[SSM97] B. Schölkopf, A. Smola, K.-R. Müller. „Kernel principal component analysis“. In: *International conference on artificial neural networks*. Springer. 1997, S. 583–588 (zitiert auf S. 53).

[SW11] H. Sun, S. Wang. „Measuring the component overlapping in the Gaussian mixture model“. In: *Data mining and knowledge discovery* 23 (2011), S. 479–502 (zitiert auf S. 25).

[SWB17] A. Senawi, H.-L. Wei, S. A. Billings. „A new maximum relevance-minimum multicollinearity (MRmMC) method for feature selection and ranking“. In: *Pattern Recognition* 67 (2017), S. 47–61 (zitiert auf S. 53).

[TL+10] N. D. Thang, Y.-K. Lee et al. „An improved maximum relevance and minimum redundancy feature selection algorithm based on normalized mutual information“. In: *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*. IEEE. 2010, S. 395–398 (zitiert auf S. 53).

[Tom76] I. Tomek. „Two Modifications of CNN“. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6.11 (1976), S. 769–772. DOI: 10.1109/TSMC.1976.4309452 (zitiert auf S. 28, 55).

[TSL00] J. B. Tenenbaum, V. d. Silva, J. C. Langford. „A global geometric framework for nonlinear dimensionality reduction“. In: *science* 290.5500 (2000), S. 2319–2323 (zitiert auf S. 31, 53, 56).

[Van19] J. Vanschoren. „Meta-learning“. In: *Automated machine learning: methods, systems, challenges* (2019), S. 35–61 (zitiert auf S. 103).

[VEP21] P. Vuttipittayamongkol, E. Elyan, A. Petrovski. „On the class overlap problem in imbalanced data classification“. In: *Knowledge-based systems* 212 (2021), S. 106631 (zitiert auf S. 25, 33).

[VK06] J. Van Hulse, T. M. Khoshgoftaar. „Class noise detection using frequent itemsets“. In: *Intelligent Data Analysis* 10.6 (2006), S. 487–507 (zitiert auf S. 95).

[VPV+09] L. Van Der Maaten, E. O. Postma, H. J. Van Den Herik et al. „Dimensionality reduction: A comparative review“. In: *Journal of Machine Learning Research* 10.66-71 (2009), S. 13 (zitiert auf S. 53).

[VRM23] J. Voggesberger, P. Reimann, B. Mitschang. „Towards the Automatic Creation of Optimized Classifier Ensembles.“ In: *ICEIS (1)*. 2023, S. 615–622 (zitiert auf S. 34, 39, 59, 60).

[WFH+05] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, M. Data. „Practical machine learning tools and techniques“. In: *Data mining*. Bd. 2. 4. Elsevier Amsterdam, The Netherlands. 2005, S. 403–413 (zitiert auf S. 52).

[WGC14] M. Woźniak, M. Grana, E. Corchado. „A survey of multiple classifier systems as hybrid systems“. In: *Information Fusion* 16 (2014), S. 3–17 (zitiert auf S. 35, 36).

[WHY+20] J. Wei, H. Huang, L. Yao, Y. Hu, Q. Fan, D. Huang. „NI-MWMOTE: An improving noise-immunity majority weighted minority oversampling technique for imbalanced classification problems“. In: *Expert Systems with Applications* 158 (2020), S. 113504 (zitiert auf S. 54, 70).

[Wil72] D. L. Wilson. „Asymptotic Properties of Nearest Neighbor Rules Using Edited Data“. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-2.3 (1972), S. 408–421. DOI: 10.1109/TSMC.1972.4309137 (zitiert auf S. 28, 50, 55).

[WJJ21] L. Wang, S. Jiang, S. Jiang. „A feature selection method via analysis of relevance, redundancy, and interaction“. In: *Expert Systems with Applications* 183 (2021), S. 115365 (zitiert auf S. 22).

[WM00] D. R. Wilson, T. R. Martinez. „Reduction techniques for instance-based learning algorithms“. In: *Machine learning* 38 (2000), S. 257–286 (zitiert auf S. 50).

[WW17] S. Wojciechowski, S. Wilk. „Difficulty factors and preprocessing in imbalanced data sets: an experimental study on artificial data“. In: *Foundations of Computing and Decision Sciences* 42.2 (2017), S. 149–176 (zitiert auf S. 25).

[ZAW19] Z. Zhao, R. Anand, M. Wang. „Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform“. In: *2019 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE. 2019, S. 442–452 (zitiert auf S. 53, 63).

[ZC01] G. Zenobi, P. Cunningham. „Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error“. In: *Machine Learning: ECML 2001: 12th European Conference on Machine Learning Freiburg, Germany, September 5–7, 2001 Proceedings 12*. Springer. 2001, S. 576–587 (zitiert auf S. 36).

[ZH21] M.-A. Zöller, M. F. Huber. „Benchmark and survey of automated machine learning frameworks“. In: *Journal of artificial intelligence research* 70 (2021), S. 409–472 (zitiert auf S. 38).

[Zho12] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012 (zitiert auf S. 34–36).

[Zho18] L. Zhou. *How to Build a Better Machine Learning Pipeline*. Available at: https://www.datanami.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline/. 2018. URL: https://www.datanami.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline/ (zitiert auf S. 21).

[ZW04] X. Zhu, X. Wu. „Class noise vs. attribute noise: A quantitative study“. In: *Artificial intelligence review* 22 (2004), S. 177–210 (zitiert auf S. 23, 72, 95).

Alle URLs wurden zuletzt am 20. 11. 2024 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift