

Technische Hochschule Deggendorf

Fakultät Angewandte Informatik

(Bachelor-Studiengang Angewandte Informatik)

Ensemble Learning

Seminararbeit für den Kurs

Grundlagen des wissenschaftlichen Arbeitens 2

an der Technischen Hochschule Deggendorf

vorgelegt von:

Sage, Sandro

00805804

am: 27. Dezember 2022

Prüfer*in:

Prof. Dr. Kristina Wanieck

Abstrakt

BLDC-Motoren bieten gegenüber herkömmlichen *DC-Motoren* viele sinnvolle Vorteile und Neurungen mit sich. Allerdings ist die Ansteuerung dieser etwas komplizierter und aufwendiger um das gleiche Ergebnis eines drehenden Ausgangsschafts zu erreichen. Für Die Ansteuerung werden sogenannte *ESCs*, oder einfach gesagt, Motortreiber benötigt in Form von Halbbrücken aus *MOSFETs* und deren repsketiven Treibern.

In dieser Arbeit wird auf die Vorteile dieser Motoren und die grundlegende Theorie der Ansteuerung eingegangen. Dabei wird in einfache Schaltung aufgestellt, welche mit einem *Integrated Circuit (IC)* erweitert wird um einen einfachen, aber funktionellen Treiber für *BLDC-Motoren* zu erhalten.

Inhaltsverzeichnis

Listen	iii
Abbildungsverzeichnis	iii
Akronyme	iii
Glossar	iii
1 Einleitung	1
2 Bagging	1
3 Stacking	3
4 Boosting	5
5 Kombinierungsansätze	5
5.1 Averaging	7
5.2 Voting	8
6 Anwendungsbeispiele von Ensemble Learning	8
Literaturverzeichnis	10

Abbildungsverzeichnis

1	Vorgehensweise der <i>Bagging</i> -Methode mittels <i>Bootstrap sampling</i> . Das Training der <i>individual learners</i> wird durch die parallele Architektur des System dargestellt.	2
2	Vergleich der K-nearest neighbors-Klassifizierungsfehler mit (\bar{e}_B) und ohne (\bar{e}_S) Bagging. Es fällt sofort auf, dass sich der Klassifizierungsfehler kaum bzw. gar nicht verringert. Dieser Vergleich wurde anhand sechs verschiedener Datensätzen erstellt [2, vgl.S.14].	3
3	K-Fold Cross Validation mit 5 folds.	4

Akronyme

CART	Classification and regression trees. 2
ESC	Electronic Speed Controller. i
IC	Integrated Circuit. i
KI	Künstliche Intelligenz. 1
KNN	K-nearest neighbors. 3
ML	Machine Learning. 1
MOSFET	Metall-Oxid-Halbleiter-Feldeffekttransistor. i

Glossar

BLDC-Motor	Bürstenloser Gleichstrommotor. i
DC-Motor	Gleichstrommotor. i

1 Einleitung

In der heutigen Zeit nimmt die Popularität und Anwendungsmöglichkeit künstlicher Intelligenz immer stärker zu [1, S.1]. Dies liegt einerseits an der Explosion erstellter und öffentlich zugänglicher Daten, aber auch andererseits an der vorgeschrittenen Forschung im Bereich *Machine Learning (ML)*, welcher als Teilbereich der *Künstliche Intelligenz (KI)* fungiert. Zudem tragen die erweiterten „computing capabilities“ [1, S.1] zu neuen Möglichkeiten der Applikation von *ML* bei. Ein komplexeres und fortgeschrittenes *ML*-Thema ist das *Ensemble Learning*. Im folgenden soll dieses komplexe Thema hinsichtlich der allgemeinen Definition, der verschiedenen Methodenansätze und der Aggregationsmöglichkeiten dargestellt und erläutert werden. Unter *Ensemble Learning* versteht man ein „multiple classifier system“ [5, S.182] oder auch „committee-based learning“ [4, S.15] genannt. Hierzu werden individuelle Lernalgorithmen zunächst auf bestimmte Art und Weise trainiert und folgend mit Hilfe einer bestimmten Kombinationstechnik aggregiert. Es gibt zwei verschiedene Arten, den homogenen und heterogenen Ensembles, zwischen denen unterschieden werden kann. Homogen bedeutet, dass alle individuellen „base learners“ [5, S.182] auf den gleichen Lernalgorithmen basieren, wodurch hingegen heterogene Ensembles auf die Verwendung unterschiedlicher *base learners* und somit Lernalgorithmen aufbauen. Wird ein heterogenes Ensemble betrachtet, so spricht man jedoch nicht mehr von *base learners*, sondern von „component learners“ [4, S.15]. Ein Vorteil bei der Anwendung von *Ensemble Learning*-Methoden ist die Verbesserung der Verallgemeinerungsfähigkeit, besser bekannt unter der „generalization ability“ [5, S.15], wobei damit die Fähigkeit gemeint ist, wie gut der Lernalgorithmus nicht nur die Trainingsdaten, sondern auch noch nie zuvor gesehene Testdaten generalisieren kann.

2 Bagging

Da die Verallgemeinerungsfähigkeit der Ensembles von der Unabhängigkeit der individuellen *base learners* abhängt und eine strikte Unabhängigkeit leider nicht gewährleistet werden kann, wird versucht, die Lernalgorithmen so unterschiedlich wie möglich voneinander zu erstellen. Dies kann dadurch bereitgestellt werden, indem jeder *base learner* auf einer bestimmten Partition des ganzen, originalen Datensatzes trainiert wird. Dabei muss beachtet werden, dass wenn nicht ausreichend Daten für das Training zur Verfügung steht, die Partitionen so erstellt werden, dass sie sich in gewissen Samples

überlappen [5, vgl. S.189]. Eine Ensemble Methode die diesen Ansatz verfolgt, ist *Bagging*[2]. *Bagging* ist eine repräsentative Methode von parallelem *Ensemble Learning*, die auf dem sogenannten *Bootstrap sampling* basiert. Der Prozess funktioniert folgendermaßen: Gibt es einen Datensatz $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, wobei y_i der Klasse bzw. dem Label des Datensamples entspricht, mit m -Samples, dann wird „randomly“ [5, S.190] ein Sample aus dem Datensatz kopiert. Dieser Vorgang wird mit „replacement“ [2, S.1] ausgeführt, dass heißt, dass eben nur das Sample kopiert wird und sich danach noch immer im originalen Datensatz befindet. Führt man diesen Prozess T -mal durch. Eine Übersicht des Verfahren bietet Abbildung 1. Der Aspekt des *replacements* wird durch die orangenen Samples veranschaulicht, da eigentlich nur ein oranges Samples in D enthalten ist, jedoch durch die *bootstrap sampling*-Methode das orange Sample mehr als einmal in den Bootstrapdatensätzen $\{D_{B1}, D_{B2}, \dots, D_{BT}\}$ existieren kann (siehe D_{B2}, D_{BT}).

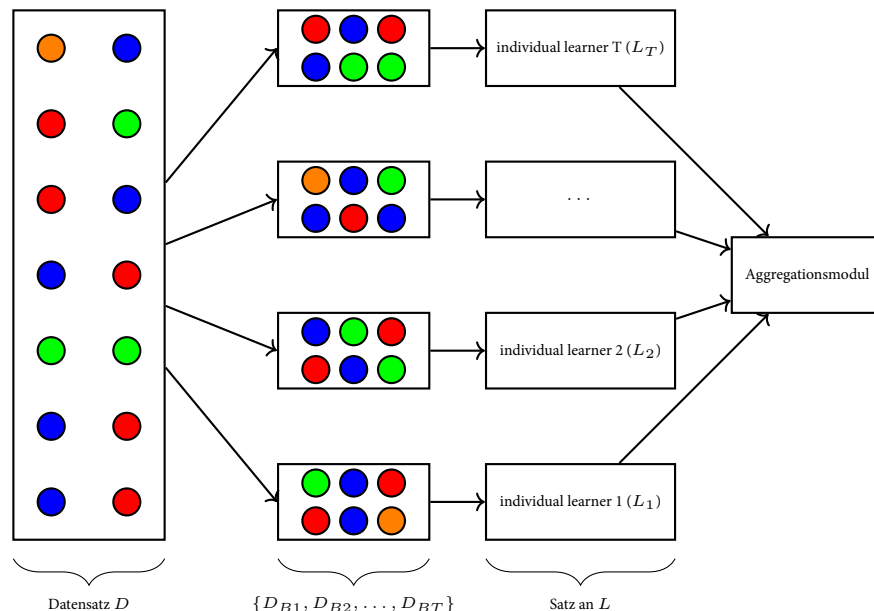


Abbildung 1: Vorgehensweise der *Bagging*-Methode mittels *Bootstrap sampling*. Das Training der *individual learners* wird durch die parallele Architektur des Systems dargestellt.

Eine Bedingung für die Funktionsweise der *Bagging*-Methode ist die Instabilität der Lernalgorithmen, wobei Breiman diese Voraussetzung studiert hat und es sich herausstellte, dass z.B. Neuronale Netze, *Classification and regression trees* (CART) eine gewisse

Instabilität aufweisen währenddessen der *K-nearest neighbors* (KNN)-Algorithmus stabil bleibt. Gibt es *base learners*, die unempfindlich gegenüber „perturbation“ [4, S.51] in den Trainingssamples sind und somit gewissermaßen gleich sind, dann wird eine Kombination dieser *base learners* keine Verbesserung in der Verallgemeinerung gewährleisten. Diese *base learners* nennt man dann auch „stable learners“ [4, S.51]. Breiman testete dies anhand sechs verschiedener Datensätze und stellt den Beweis hierfür in Abbildung 2 dar. Ein Vorteil des *Bagging* Ensemble Learning's ist die Parallelität, die es ermöglicht, durch Verwendung von „multi-core computing processors“ [5, S.48] oder parallelen Computerarchitekturen, die Trainingszeit zu minimieren.

Data Set	\bar{e}_S	\bar{e}_B
waveform	26.1	26.1
heart	5.1	5.1
breast cancer	4.4	4.4
ionosphere	36.5	36.5
diabetes	29.3	29.3
glass	30.1	30.1

Abbildung 2: Vergleich der K-nearest neighbors-Klassifizierungsfehler mit (\bar{e}_B) und ohne (\bar{e}_S) Bagging. Es fällt sofort auf, dass sich der Klassifizierungsfehler kaum bzw. gar nicht verringert. Dieser Vergleich wurde anhand sechs verschiedener Datensätzen erstellt [2, vgl.S.14].

3 Stacking

Die zweite Methode des *Ensemble Learning's* ist das sogenannte *Stacking*. Wenn hier ausreichend viel Daten vorhanden sind, so können unter der Verwendung eines *meta-learners* mehrere individuelle Lerner kombiniert werden. Es wird auch von „[C]ombining by learning“ [5, S.196] gesprochen, da das Aggregationsmodul selbst durch einen individuellen Lernen aufgebaut wird. Die individuellen Lerner werden *first-level learners* und die Lerner, die die Kombination ausüben, *second-level learners* oder *meta learners* genannt. Im ersten Schritt des *Stacking's* werden die *first-level learners* auf Basis des originalen, gesamten Datensatzes trainiert und im nächsten Schritt wird ein neuer Datensatz,

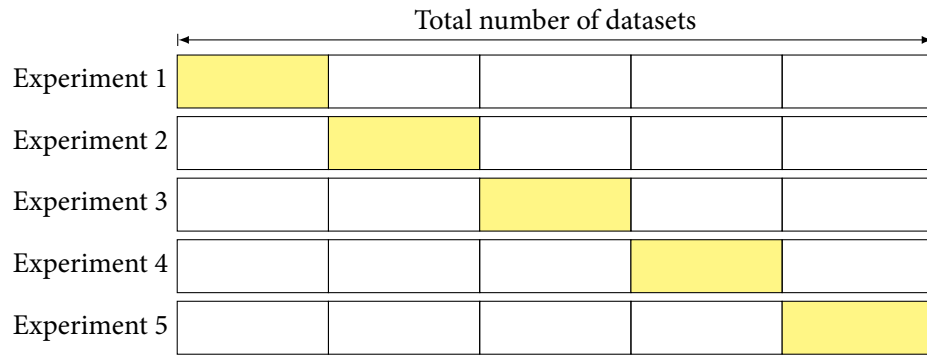
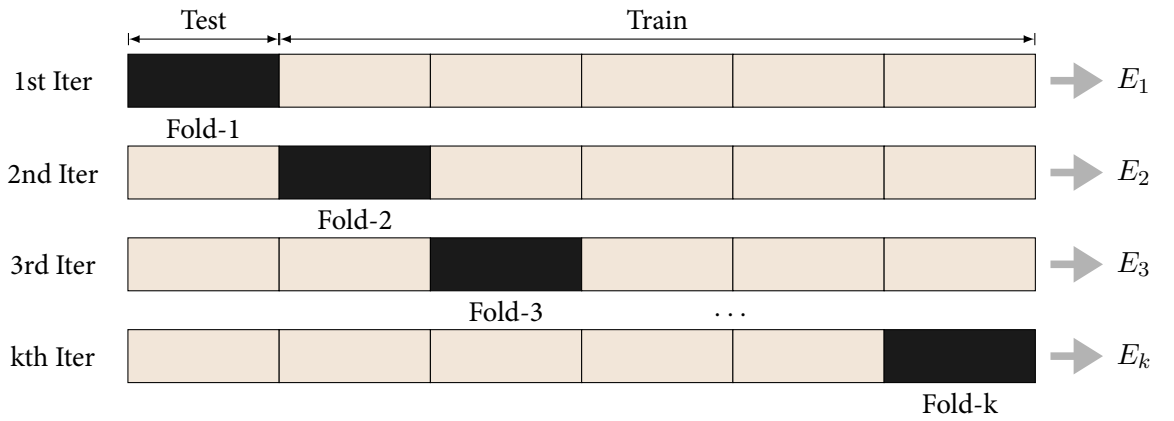


Abbildung 3: K-Fold Cross Validation mit 5 folds.



der dann als Trainingssatz für die *meta-learners* fungiert, generiert. Der neu-generierte Datensatz besteht aus den Vorhersagen oder Outputs der *first-level learners*. Der generierte Datensatz darf nicht exakt auf der Basis der Trainingssamples im ersten Schritt erstellt werden, da sonst ein hohes Risiko an *Overfitting* besteht. Deshalb wird in diesem Ansatz das *k-fold Cross-Validation* angewendet. Beim *k-fold CV* wird der originale Datensatz in k annähernd gleich große Partitionen D_1, D_2, \dots, D_k unterteilt. D_j wird als Testsatz und $\bar{D}_j = D/D_j$ als Trainingssatz des j -ten Folds angenommen. Gibt es T *first-level learners* und $h_t^{(j)} z_{it} = h_t^{(j)}(x_i)$

4 Boosting

Die letzte Methoden ist das sogenannte *Boosting*. Bei diesem Verfahren wird versucht „weak learners“ [5, S.184] in „strong learners“ [5, S.184] umzuwandeln. Zunächst wird mit dem Training des ersten *base learners* auf Basis einer bestimmten Datenverteilung begonnen. Danach werden die Trainingssamples hinsichtlich des Ergebnisses der Trainingsphase angepasst, indem Samples, die als falsch klassifiziert wurden, nun mehr Aufmerksamkeit in den noch folgenden Trainingsphasen der weiteren *base learners* bekommen. Vereinfacht bedeutet das, dass nach dem Training des ersten *base learner's*, die Datenverteilung angepasst wird und daraufhin der zweite *base learner*, dessen Ergebnis für die Anpassung der Datenverteilung für den nächsten Iterationsschritt verwendet wird, trainiert wird. Der Prozess wiederholt sich so lange bis eine vordefinierte Anzahl T an *base learners* erreicht wird. Schlussendlich erhält dann jeder dieser Lernalgorithmen eine Gewichtung und wird zusammen mit den anderen kombiniert. Eine der bekanntesten und einflussreichsten Boosting-Algorithmen ist der *AdaBoost*-Algorithmus (ZITAT). Der Algorithmus setzt voraus dass der gewählte *base learner* in der Lage ist, mit bestimmten Datenverteilungen trainiert werden kann. Eine Möglichkeit für das Erreichen einer solchen Datenverteilung bietet das „re-weighting“, bei dem nach jeder Trainingsrunde ein neues Gewicht den Samples in der Datenverteilung zugeordnet wird. Eine weitere Option bietet das „re-sampling“, welches verwendet werden kann, falls der Lernalgorithmus *re-weighting* nicht unterstützt. Hier wird in jeder Trainingsiteration eine neue Teilmenge aus der gesamten Datenverteilung entnommen. Eine Erweiterung des *re-sampling's* bietet den Neustart des *Boosting's* an, wobei dieser Ansatz in (ZITAT) genauer erläutert wird.

5 Kombinationsansätze

Nachdem nun verschiedene Methoden des *Ensemble Learning's* vorgestellt worden sind, behandelt dieses Kapitel die Kombinationsoptionen nach denen die *base learners* zusammengefügt werden können. Dietterich[3, vgl. S.3f.] liefert drei Perspektiven, die erläutern warum das Kombinieren vorteilhaft ist. Aus statistischer Sicht ist der Hypothesenraum meistens sehr groß, wodurch es mehrere Hypothesen, die die gleichen Ansätze anstreben, gibt. Wenn nur ein einzelner Lernalgorithmus oder *base learner*

Algorithmus 1: AdaBoost

Eingabe: Trainingsdatensatz $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
Base Learner \mathcal{L} ;
Anzahl der Trainingsrunden T .

```
1 Beginn
2    $\mathcal{D}_1(\mathbf{x}) = 1/m$ ;
3   für  $t = 1, 2, \dots, T$  tue
4      $h_t = \mathcal{L}(D, \mathcal{D}_t)$ ;
5      $\epsilon_t = P_{\mathbf{x} \sim D_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$ ;
6     wenn  $\epsilon_t > 0.5$  dann
7       | abbrechen
8     Ende
9      $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ ;
10     $\mathcal{D}_{t+1}(\mathbf{x}) = \frac{\mathcal{D}_t(\mathbf{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{wenn } h_t(\mathbf{x}) = f(\mathbf{x}); \\ \exp(\alpha_t), & \text{wenn } h_t(\mathbf{x}) \neq f(\mathbf{x}); \end{cases}$ 
11     $= \frac{\mathcal{D}_t(\mathbf{x}) \exp(-\alpha_t f(\mathbf{x}) h_t(\mathbf{x}))}{Z_t}$ ;
12 Ende
```

Ausgabe: $F(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}))$

betrachtet wird, dann hängt die Verallgemeinerungsgenauigkeit (*generalization performance*) ausschließlich von der Genauigkeit dieses *base learners* ab. Werden nun mehrere Hypothesen kombiniert, so verringert sich auch das Risiko einer Falschaussage. Werden die Lernalgorithmen aus rechnerischer Perspektive betrachtet, so können diese oftmals in sogenannten „local optimum[s]“ [5, S.193] feststecken. Werden jedoch mehrere Lernalgorithmen verwendet, so senkt sich auch hier wieder das Risiko in einem *local optimum* festzustecken. Zuletzt gibt es noch die repräsentative Perspektive, bei der angenommen wird, dass die richtige Hypothese nicht in Raum der existierenden Hypothesen befindet. Somit kann durch Verwendung mehrerer Hypothesen eine genauere Annäherung an die richtige Hypothese gewährleistet werden [5, vgl. S.194] [3, vgl. S.4]. In den weiteren Unterkapiteln werden die verschiedenen Strategien der Kombination erläutert.

5.1 Averaging

Die erste Art der Kombination ist das *Averaging*, die in der Regression ihre Anwendung findet, da hier ein numerischer Output erwartet wird. Typische *Averaging*-Methoden sind das *simple Averaging*

$$H(x) = \frac{1}{T} \sum_{i=1}^T h_i(x) \quad (1)$$

und das *weighted Averaging*

$$H(x) = \frac{1}{T} \sum_{i=1}^T w_i h_i(x) \quad (2)$$

Es wird angenommen, dass das Ensemble T individuelle Lerner bzw. *base learners* $\{h_1, h_2, \dots, h_T\}$ beinhaltet, wobei $h_i(x)$ dem Output des i -ten *base learners* auf das Sample x entspricht. Bei dem *weighted Averaging* steht w_i für das Gewicht des i -ten individuellen Lerners. Typischerweise gilt $w_i \geq 0$ und $\sum_{i=1}^T w_i = 1$. Viele empirische Studien haben bewiesen, dass *weighted Averaging* nicht notwendigerweise besser abschneidet als *simple Averaging* FEHLLENDE QUELLEN IN ML. Wenn die einzelnen *base learners* ähnliche Genauigkeit aufweisen, so gilt im Allgemeinen, dass *simple Averaging* zu verwenden. Weisen die individuellen Lerner jedoch erhebliche Unterschiede in der Genauigkeit auf, so wird das *weighted Averaging* vorgezogen [5, vgl. S.195].

5.2 Voting

Da nun auf die Anwendung für die Regression beschrieben wurde, befasst sich dieser Abschnitt mit den Methoden für die Klassifizierung. Hierfür wird das *Voting*, dass in drei verschiedenen Arten vorgestellt wird. Bei dem *Voting* gibt es individuelle Lerner $\{h_1, h_2, \dots, h_T\}$ und h_i kann N verschiedene Klassenlabels $\{c_1, c_2, \dots, c_N\}$ vorhersagen. Entspricht der N -dimensionale Vektor $(h_i^1(x), h_i^2(x), \dots, h_i^N(x))$ und h_i dem Output des Samples x , so ist $h_i^j(x)$ gleich der Output von h_i für das Klassenlabel c_j . Die erste *Voting*-Methode ist das *Majority Voting*

$$H(x) = \begin{cases} c_j, & \text{if } \sum_{i=1}^T h_i^j(x) > 0.5 \sum_{k=1}^N \sum_{i=1}^T h_i^k(x); \\ \text{reject}, & \text{otherwise.} \end{cases} \quad (3)$$

,die zweite Methode ist das *Plurality Voting*

$$H(x) = c_{\arg \max_j \sum_{i=1}^T h_i^j(x)} \quad (4)$$

und zuletzt das *Weighted Voting*

$$H(x) = c_{\arg \max_j \sum_{i=1}^T w_i h_i^j(x)} \quad (5)$$

. Wird das *Weighted Voting* genauer betrachtet so lässt sich erkennen, dass es vom *Plurality Voting* abgeleitet ist und als Erweiterung noch Gewichtungen berücksichtigt werden. w_i entspricht somit dem Gewicht von h_i . Ein Nachteil des *Majority Voting*'s ist die *reject*-Option, da in vertrauenswürdigen Anwendungen, wie z.B. in der Medizin, immer ein Resultat getroffen werden muss [5, vgl. S.195]

6 Anwendungsbeispiele von Ensemble Learning

- KDD-Cup most famous data mining competition: -tasks: network intrusion detection (1999) molecular bioactivity and protein locale prediction (2001) musik recommendation (2011) etc. - various techniques, ensemble learning methods have drawn the most attention and won the competitions for the most times - (2009-2011) first and second place winners have used ensemble methods

- Netflix Prize held by the online DVD-rental service *Netflix* - predictions about how much someone is going to enjoy a movie based on their preferences - on september 21, 2009 Netflix awarded the 1M Dollar grand prize to the team *BellKor's Pragmatic Chaos* - solution was based on combining various classifiers
- ensemble methods have been successfully applied to diverse real-world tasks. For example computer vision in almost all branches such as object detection, recognition and tracking - credit card fraud detection, bankruptcy prediction, protein structure classification, weather forecasting, aircraft engine fault diagnosis

Literaturverzeichnis

- [1] R. Boutaba et al. „A comprehensive survey on machine learning for networking: evolution, applications and research opportunities“. In: *Journal of Internet Services and Applications* 9.1 (2018), S. 1–99. ISSN: 1867-4828. DOI: 10.1186/s13174-018-0087-2. URL: <https://link.springer.com/article/10.1186/s13174-018-0087-2> (siehe Seite 1).
- [2] L. Breiman. „Bagging predictors“. In: *Machine Learning* 24.2 (1996), S. 123–140. ISSN: 0885-6125. DOI: 10.1007/BF00058655. URL: <https://link.springer.com/article/10.1007/BF00058655> (siehe Seiten 2, 3).
- [3] T. G. Dietterich. „Ensemble Methods in Machine Learning“. In: Bd. 1857, S. 1–15. DOI: 10.1007/3-540-45014-9_1 (siehe Seiten 5, 7).
- [4] Z.-H. Zhou. *Ensemble Methods*. Chapman and Hall/CRC, 2012. ISBN: 9780429151095. DOI: 10.1201/b12207. URL: <https://www.taylorfrancis.com/books/mono/10.1201/b12207/ensemble-methods-zhi-hua-zhou> (siehe Seiten 1, 3).
- [5] Z.-H. Zhou. *Machine Learning*. Singapore: Springer Singapore, 2021. ISBN: 978-981-15-1966-6. DOI: 10.1007/978-981-15-1967-3 (siehe Seiten 1–3, 5, 7, 8).

Kolophon

Dieses Dokument ist ein \LaTeX 3 Dokument der KOMA-Script Klasse und wurde mit *Minion Pro* gesetzt. Alle eigenen Zeichnungen sind mit TikZ und PGFPLOTS erstellt. Kompiliert wurde es mit Lua \LaTeX und biber auf Windows am 27. Dezember 2022.

