technische universität
dortmund

Bachelorarbeit

Anwendung von Ensemble-Modellen unter Ressourcenbeschränkungen: Ein Framework für Ensemble Pruning Verfahren

Henri Petuker

Gutachter:
Prof. Dr. Katharina Morik
M.Sc. Sebastian Buschjäger

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
https://www-ai.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Maschinelles Lernen als Methodik der effektiven Datenanalyse ist fester Bestandteil moderner Informationsverarbeitung. Techniken zur personalisierten Anzeige von Suchergebnissen und Empfehlungen im Internet und Social Media sind bereits fest in Alltagsabläufe integriert [16]. Zugleich sind diese Methoden in einigen Bereichen bisherigen Ansätzen überlegen. Komplexe Aufgaben wie autonomes Fahren oder Sprach- und Bilderkennung, die kaum mit von Menschenhand geschriebenen Programmen zu bewältigen sind, sind bedeutende Anwendungsbereiche für das maschinelle Lernen, ebenso wie die effiziente Analyse großer und komplexer Datenmengen [31, S. 3]. Die überzeugenden Resultate und die Flexibilität in der Anwendung wecken das Interesse an aktuellen Entwicklungen des maschinellen Lernens und tragen zur wachsenden Verbreitung bei [16].

Die Informationstechnologie entwickelt sich konstant und mehrere Trends lassen sich seit Jahren beobachten. Die Zahl an elektronischen Geräten und Sensoren, welche Daten erheben und miteinander vernetzt sind, steigt exponentiell [32]. Smartphones, elektronische Wearables und weitere *Internet of Things*-Geräte sind heutzutage nicht mehr wegzudenken. Infolgedessen wächst auch die Menge an gesammelten Daten: Allein durch elektronische Sensoren und Geräte fallen jährlich hunderte von Zettabytes an [32]. Damit wächst auch die Bedeutung des maschinellen Lernens, da dessen Methoden effektive Lösungen für die Analyse von großen und komplexen Datenmengen bieten. Jedoch sorgen die steigenden Mengen an Daten und die hohen Kosten für ihre Übertragung dazu, dass die Auswertung in Rechenzentren und Clouds weniger rentabel wird und folglich lokale Analysen auf den elektronischen Endgeräten an Bedeutung gewinnen [32].
Mit der Integration von Methoden des maschinellen Lernens auf unterschiedlicher Hardware werden auch unterschiedliche Anforderungen an diese gestellt. So ist die Verwendung von etablierten Methoden auf ressourcenbeschränkter Hardware, wie auf kleinen und verteilten Geräten, nicht immer möglich, da Faktoren wie Rechenleistung oder verfügbarer Speicherplatz Limitationen setzen. Im Zentrum steht hierbei die Ausführung der finalen Methoden

auf den Endgeräten, da die Anpassung und Optimierung im Voraus auf leistungsstarker Hardware geschehen kann. Insgesamt besteht ein Interesse die Effizienz vorhandener Methoden des maschinellen Lernens zu steigern, ohne dabei signifikante Verluste bei der Qualität in Kauf nehmen zu müssen.

Diese Arbeit beschäftigt sich in diesem Kontext mit Ensemble Learning, dessen Methoden zu State of the Art Lösungen für Aufgaben aus dem Bereich des maschinellen Lernens gehören [30]. Beispielhaft profitierten Aufgaben aus dem Bereich der Computer Vision wie Objekterkennung und -tracking vom Einsatz von Ensembles. Auch in der medizinischen Diagnostik konnten sicherere Prognosen durch die Unterstützung von Ensemble Methoden erreicht werden [36, S. 18-19].
Zwar überzeugen Ensemble Methoden in ihrer Anwendung, die Größe und Komplexität ihrer Modelle sind aber verhältnismäßig hoch [35]. Aufgrund der weiten Verbreitung von Ensemble Methoden sind Verbesserungen in ihrer Effizienz von Interesse, besonders im Hinblick auf ihre Anwendung auf ressourcenbeschränkten Endgeräten. In dieser Arbeit werden Verfahren untersucht, welche bereits fertige und auf spezifische Aufgaben angepasste Ensemble Modelle bezüglich ihrer Effizienz optimieren.

Ziel dabei ist es neben einer Übersicht über sogenannte *Ensemble Pruning* Verfahren auch ein theoretisches Framework zu entwickeln, welches die heterogenen Ansätze auf wenige wesentliche Eigenschaften reduziert. Dies ermöglicht das Vergleichen dieser Verfahren bezüglich der identifizierten Eigenschaften und hebt ihre Ähnlichkeiten hervor. Folglich soll dieses erweiterbare Framework auch als Anregung für die Erforschung neuartiger Ansätze dienen. Weiterhin wird gezeigt, dass das Ensemble Pruning Problem in dessen verbreitetsten Form als konvexes Optimierungsproblem formuliert werden kann.

1.1 Aufbau der Arbeit

Zu Beginn werden in Kapitel 2 die theoretischen Grundlagen für die Themen dieser Arbeit vorgestellt. Zentrale Begriffe des maschinellen Lernens werden in Kapitel 2.1 erläutert. Darauf aufbauend wird das Konzept der Ensembles eingeführt, das bestehende Methoden des maschinellen Lernens erweitert. Als konkretes Beispiel für Ensembles werden verbreitete Modelltypen wie *random forests* angeführt. Kapitel 2.2 thematisiert das zentrale Problem dieser Arbeit, welches sich mit der Effizienzsteigerung von Ensemblemodellen befasst. Das Ensemble Pruning Problem wird sowohl aus einer praktischen als auch aus einer mathematischen Sichtweise betrachtet. Schließlich werden wichtige Charakteristika von Verfahren, die dieses Problem lösen, aufgelistet. Die Grundlagen der mathematischen Optimierung,

zugeschnitten auf die spätere Betrachtungsweise des Ensemble Pruning Problems, werden in Kapitel 2.3 erläutert.
Kapitel 3 umfasst verwandte Arbeiten, die unter anderem richtungsweisend für diese Arbeit waren. Es werden Werke beleuchtet, die Ensemble Pruning Verfahren kategorisieren oder Konzepte für ein systematisches Framework dieser vorstellen. Ebenso wird eine Arbeit vorgestellt, dessen Betrachtungsweise auf das zugrundeliegende Problem als Basis für die Ergebnisse dieser Arbeit dienten.

Das theoretische Framework für Ensemble Pruning Verfahren ist das Thema des vierten Kapitels. Es wird erläutert, wie sich dieses Framework von bisherigen Betrachtungen des Problems unterscheidet und wie das Problem als mathematisches Optimierungsproblem formuliert werden kann. Ebenso wird gezeigt, dass dieses Optimierungsproblem konvex ist. Anschließend werden Merkmale von bestehenden Ensemble Pruning Verfahren in das Framework integriert und es werden die Limitationen des Frameworks aufgezeigt.
Kapitel 5 umfasst eine Implementation des Frameworks mit einer Auswahl an Ensemble Pruning Verfahren in Python. Es können dabei die in Kapitel 2.1.3 vorgestellten Ensemblemodelle generiert und mittels den in Kapitel 4.3 und Kapitel 4.4 betrachteten Eigenschaften von Ensemble Pruning Verfahren verkleinert werden. Auf Implementierungsdetails wird in Kapitel 5.1 eingegangen. Außerdem werden in Kapitel 5.2 und Kapitel 5.3 zwei Untersuchungen vorgestellt, die mithilfe des Frameworks umgesetzt sind. Zum einen wird untersucht, wie sich die Wahl bestimmter Merkmale von Ensemble Pruning Verfahren auf die Effektivität der verkleinerten Ensembles auswirkt. Zum anderen wird die Sensibilität der Pruning Verfahren bezüglich dem Grad der Größenreduzierung analysiert.
Die Arbeit endet mit einer Zusammenfassung und einer Diskussion des Ensemble Pruning Problems als Pareto-Optimierungsproblem mit den Kriterien Ressourceneffizienz und Effektivität.

Kapitel 2

Theoretische Grundlagen

2.1 Maschinelles Lernen

Maschinelles Lernen befasst sich mit Verfahren, welche anhand von Daten und Lernalgorithmen eine Auswahl an Problemen autonom und ohne menschliche Expertise bewältigen. Diese Verfahren generieren Modelle, die Muster und Strukturen in Daten erkennen und ihr generalisiertes Wissen auf zukünftige Daten anwenden können.
Ein Teilgebiet des maschinellen Lernens ist das überwachte Lernen, dessen Methoden als Ziel die korrekte Vorhersage eines Zielmerkmales der Daten haben. Aus Beispielen werden Modelle generiert, die sowohl auf den Beispieldaten als auch auf unbekannten Daten das Zielmerkmal möglichst korrekt vorhersagen. Es wird zwischen den Problemtypen der Regression und der Klassifikation unterschieden, welche sich am Typ des Zielmerkmales unterscheiden. Zielmerkmale für Regressionsaufgaben sind kontinuierlich $y \in \mathbb{R}$ und für Klassifikation diskret $y \in \mathbb{N}$ [36, S. 1-2]. Zentrale Begriffe für das überwachte Lernen, mit besonderem Bezug auf Klassifikation, werden im Folgenden erläutert.

Daten Die Domäne der zu klassifizierenden Daten wird mit $\mathcal{X}$ bezeichnet, wobei ein $x \in \mathcal{X}$ einer konkreten Instanz beziehungsweise einem Datenpunkt dieser Domäne entspricht. Datenpunkte mit m unterschiedlichen Eigenschaften (engl. *features*) werden als m-dimensionaler Vektor $x \in \mathbb{R}^m$ dargestellt, wobei x_i für das i-te Merkmal der Instanz x steht. Zusätzlich besitzt jede Instanz ein Label der Domäne $\mathcal{Y}$, welches die Instanz zu einer Kategorie $y \in \mathcal{Y}$ zuordnet. Binäre Klassifikationsprobleme verwenden für die Labels die Menge $\{0, 1\}$ oder $\{-1, 1\}$ und Klassifikationsprobleme mit $l \in \mathbb{N}$ Labels die Menge $\{1, ..., l\}$. Formal werden Instanzen aus $\mathcal{X}$ als Realisierung einer Zufallsvariablen aufgefasst, welche nach einer Wahrscheinlichkeitsverteilung $\mathcal{D}$ verteilt ist. Ein erhobener Datensatz für ein Klassifikationsproblem $D = \{(x_j, y_j) \in (\mathcal{X} \times \mathcal{Y}) \mid j \in \{1, ..., N\}\}$ wird somit als N unabhängige Realisierungen von Instanzen nach $\mathcal{D}$ verstanden. Dies wird mit $x \sim \mathcal{D}$ notiert und kann als Beobachtung von x interpretiert werden. Zudem wird eine theoretische Ziel-

funktion $f : \mathcal{X} \to \mathcal{Y}$ eingeführt, welche alle Instanzen korrekt klassifiziert. Folglich gilt: $\forall j \in \{1, ..., N\} : f(x_j) = y_j$ [31, S. 13-14].

Modell Ziel des überwachten Lernens ist es eine Funktion $h : \mathcal{X} \to \mathcal{Y}$ zu generieren, welche korrekte Klassifikationen trifft. Für Instanzen $x \sim \mathcal{D}$ sollte folglich $h(x) = f(x)$ gelten. Damit soll die generierte Funktion h möglichst akkurat die Zielfunktion f repräsentieren und wird in diesem Kontext auch als Hypothese oder Modell bezeichnet.

Lernprozess Der Lernprozess, auch *Training* genannt, beinhaltet die Anpassung eines Modells auf das gegebene Klassifikationsproblem. Hierfür wird ein Teil des erhobenen Datensatzes D als Trainingsdatensatz D_T abgesondert und die restlichen Daten werden als Validierungsdatensatz D_V für folgende Tests des Modells gespeichert. Mit den Beispielen aus D_T und einem Lernalgorithmus wird der Hypothesenraum $\mathcal{H}$ eines Modelltyps durchsucht, welcher alle Funktionen $h : \mathcal{X} \to \mathcal{Y}$ umfasst. Durch Anpassungen von Modellparametern wird versucht ein $h \in \mathcal{H}$ zu finden, das f besonders gut repräsentiert.
Um dies beurteilen zu können, wird der erwartete Fehler (2.1) eingeführt. Dieser entspricht der erwarteten Anzahl an falsch klassifizierten Instanzen des Modells h auf allen $x \sim \mathcal{D}$.

$$err(h) = \mathbb{E}_{x \sim \mathcal{D}}[\mathbb{1}(h(x) \neq f(x))] \tag{2.1}$$

Damit ist das Ziel ein Modell zu generieren, welches den erwarteten Fehler minimiert. Jedoch sind in realen Problemen die zugrundeliegende Verteilung $\mathcal{D}$ und die Funktion f nicht bekannt, weshalb es erforderlich ist andere Maße zu verwenden, die den erwarteten Fehler indirekt erfassen. Hierfür wird der empirische Fehler (2.2) eingeführt, welcher den Fehler von h auf dem Trainingsdatensatz D_T misst, der für die Generierung des Modells h verwendet wurde [31, S. 14-15] [36, S. 2-3].

$$err_{D_T}(h) = \frac{|\{i \in \{1, ..., N_T\} \,|\, h(x_i) \neq y_i\}|}{N_T} \tag{2.2}$$

Validierung Ein niedriger empirische Fehler auf D_T muss jedoch nicht einem niedrigen erwarteten Fehler entsprechen, weshalb eine zusätzliche Validierung des Modells vorgenommen wird. Hierfür wird der Fehler des Modells auf dem separaten Validierungsdatensatz D_V gemessen, der nicht für das Generieren des Modells verwendet wurde. Dieser Validierungsfehler kann Aufschluss über die Generalisierungsfähigkeit des Modells geben.
Um den Einfluss der Wahl der Training- und Validierungsdatensätze zu minimieren, wird zum Beispiel die *k-fold cross* Validierung verwendet. Dabei wird der verfügbare Datensatz D in $\frac{N}{k}$ große Teilmengen partitioniert. Für jedes dieser k Teilmengen wird ein Modell mit den restlichen $k-1$ Teilmengen als D_T und mit der ausgelassenen Teilmenge als D_V trainiert und validiert. Der durchschnittliche Validierungsfehler aller generierten Modelle fungiert als Richtwert für den erwarteten Fehler des Modells [31, S. 116-117, S. 119-120].

Lossfunktion Lossfunktionen $\ell : (\mathcal{Y} \times \mathcal{Y}) \to \mathbb{R}$ messen den Fehler eines Modells auf einem Datensatz. Jeder Lernalgorithmus, der ein Modell durch Parameteranpassungen auf ein Problem des überwachten Lernens anpasst, verwendet eine Lossfunktion und versucht diese während des Lernprozesses zu minimieren. In diesen Fällen spricht man auch von einem Trainingsloss und je nach Problemtyp werden unterschiedliche Lossfunktionen verwendet: Während bei der Generierung von Regressionsmodellen häufig der *mean squared error* minimiert wird, wird bei Klassifikationsmodellen zum Beispiel die *cross entropy* als Lossfunktion genutzt.

Metrik Neben Lossfunktionen werden auch Metriken verwendet, um die Vorhersagen eines Modells nach dessen Generierung zu bewerten. Diese Metriken werden nicht als eine mathematische Metrik im strengen Sinne aufgefasst. Sie dienen als Abstandsmaß entweder für die Vorhersagen eines Modells zu den korrekten Vorhersagen auf einem Datensatz oder für die Vorhersagen von zwei Modellen. Solche Metriken können für Menschen interpretierbarer sein und damit das Vergleichen von Modellen erleichtern. Für binäre Klassifikationsprobleme werden *accuracy* und *Area under the ROC-Curve* (AUC) betrachtet, wobei diese Metriken auch auf Probleme mit mehr als zwei Labels erweitert werden können.

Die Variablen n_0 und n_1 entsprechen der Anzahl an positiven und negativen Instanzen des Datensatzes D, auf dem die *accuracy* des Modells h gemessen werden soll. Weiterhin werden definiert:

$$\begin{aligned} \text{True Positive Rate (TP)} &= \frac{|\text{korrekt klassifizierte pos. Instanzen}|}{|\text{positive Instanzen}|} \\ \text{False Positive Rate (FP)} &= \frac{|\text{falsch klassifizierte neg. Instanzen}|}{|\text{negative Instanzen}|} \\ \text{True Negative Rate (TN)} &= 1 - \text{FP} \end{aligned} \tag{2.3}$$

Die *accuracy* des Modells auf D berechnet sich durch $acc(h) = \frac{\text{TP} \cdot n_0 + \text{TN} \cdot n_1}{n_0 + n_1}$ und gibt Auskunft darüber, wie viele der Instanzen korrekt klassifiziert wurden [13].

Die *Area under the ROC-Curve* (AUC) Metrik misst den Flächeninhalt unter der *reciever operating characteristics curve* (ROC). Diese Kurve setzt für alle Thresholds $\theta \in [0, 1]$ des Modells die True Positive Rate $\text{TP} \in [0, 1]$ und False Positive Rate $\text{FP} \in [0, 1]$ des Modells auf D ins Verhältnis. Threshold bezeichnet hier den Schwellenwert für die *confidence*, mit der das Modell eine positive Klassifikation treffen muss. Insgesamt wird für eine Menge an Thresholds die Performance des Modells gemessen und die AUC fasst diese Informationen zusammen. Im Gegensatz zur *accuracy* bezieht AUC so die *confidence* der Vorhersagen eines Modells ein, weswegen deren Verwendung vorteilhafter sein kann [13].

2.1.1 Ensembles

Die Verwendung eines einzelnen Modells kann für Aufgaben aus dem maschinellen Lernen ausreichend sein. Ensemble Methoden jedoch erweitern bestehende Vorgehensweisen, indem mehrere Modelle für die gleiche Aufgabe generiert und die Vorhersagen dieser zu einer Gesamtvorhersage kombiniert werden. Die einzelnen Modelle innerhalb eines Ensembles werden dabei mittels bestehenden Methoden des maschinellen Lernens trainiert. Ensemble Methoden bestehen somit zum einen aus der Generierung einer Menge $\mathcal{H}_{Ens} = \{h_i \in \mathcal{H} \,|\, i \in \{1, ..., n\}\}$ an effektiven Modellen und zum anderen aus einer Kombinationsregel für die einzelnen Vorhersagen. Im Folgenden wird eine Auswahl an Methoden der zwei Bereiche vorgestellt.

Generierung Ein effektives Ensemble besteht aus Modellen, die sowohl korrekte Vorhersagen treffen als auch divers zueinander sind [30]. Folglich liegt besonderer Wert auf der Generierung der Modellmenge und die Verfahren hierzu lassen sich in zwei Kategorien aufteilen: sequenzielle und parallele Generierung.
Sequenzielle Generierung, auch *Boosting* genannt, zielt darauf, ab iterativ weitere Modelle zu trainieren, welche sich auf die Fehler der vorangegangenen Modelle fokussieren, und diese nach und nach dem Ensemble hinzuzufügen. Das einflussreichste *Boosting*-Verfahren ist hierbei *AdaBoost* [36, S. 24].
Als Beispiel für die parallele Generierung dient *Bagging* und das verwendete *bootstrapping* Verfahren. Für jedes der zu generierenden Modelle wird ein individueller Trainingsdatensatz erzeugt, indem m-mal Instanzen aus dem gesamten Trainingsdatensatz mit Zurücklegen gezogen werden. Da der Generierungsprozess eines Modells nicht abhängig von den Fehlern der anderen Modelle des Ensembles ist, lässt er sich parallelisieren [36, S. 48].

Kombination Je nach Modelltyp werden unterschiedliche Methoden für das Aggregieren der einzelnen Vorhersagen verwendet. Während Regressionsmodelle stetige Vorhersagen $y^* \in \mathbb{R}$ für eine Instanz ausgeben, ordnen Modelle für Klassifikation die Instanzen einem Label zu. Mit l als Anzahl an möglichen Labels gibt ein Klassifikationsmodell h_i für Datenpunkt x_j einen l-dimensionalen Vektor y_i^j aus. Dieser Vektor ist eine Wahrscheinlichkeitsverteilung über die l Labels und der k-te Eintrag entspricht der Wahrscheinlichkeit, für die Modell h_i auf x_j das k-te Label als korrekt einstuft. Statt einer Wahrscheinlichkeitsverteilung kann ein Klassifikationsmodell auch direkt das Label mit der höchsten Wahrscheinlichkeit nach y_i^j ausgeben [36, S. 71-72].
Für Regressionsensembles kann zum Beispiel *Weighted Averaging* (2.4) verwendet werden, das jedem $h_i \in \mathcal{H}_{Ens}$ ein Gewicht $w_i \in \mathbb{R}$ für dessen Relevanz im Ensemble zuweist

und anschließend alle Vorhersagen aufsummiert. Damit ergibt sich die Vorhersage eines Ensembles H auf Instanz x_j :

$$H(x_j) = \sum_{i=1}^{n} w_i h_i(x_j) \tag{2.4}$$

wobei die Gewichte den Bedingungen $\forall i \in \{1, ..., n\} : w_i \geq 0$ und $\sum_{i=1}^{n} w_i = 1$ unterliegen und zusätzlich bestimmt werden müssen. Das *Averaging* mit gleichen Gewichten $\forall i \in \{1, ..., n\} : w_i = \frac{1}{n}$ ist ebenfalls möglich und ist besonders für Ensembles, in denen die Modelle ähnlich gute Vorhersagegenauigkeiten erreichen, geeignet [36, S. 68-71].
Bezüglich Klassifikationsensembles wird in der Regel eine Form des *Votings* verwendet. Treffen die einzelnen Modelle für Instanz x_j bereits eine eindeutige Vorhersage für ein Label, kann beispielhaft *Plurality Voting* verwendet werden, welches für die Vorhersage des Ensembles H auf x_j das Label mit den meisten Stimmen auswählt. Geben die Modelle für x_j jedoch Wahrscheinlichkeitsverteilung aus, wird *Soft Voting* verwendet, das für jedes Label die Wahrscheinlichkeiten aufaddiert und normiert. Bei einer gleichmäßigen Gewichtung aller Modelle ergibt sich für die Vorhersage des Ensembles auf x_j:

$$H(x_j) = \frac{1}{n} \sum_{i=1}^{n} h_i(x_j) \tag{2.5}$$

$H(x_j)$ und die $h_i(x_j)$ entsprechen l-dimensionalen Vektoren. Auch beim *Soft Voting* können Gewichte hinzugefügt werden, um den Einfluss einzelner Modelle zu verändern. Für die finale Vorhersage des Ensembles wird das Label in $H(x_j)$ mit der höchsten Wahrscheinlichkeit ausgewählt [36, S. 73, S. 75-76] .

Der Einsatz von Ensembles ist im Vergleich zu einzelnen Modellen vorteilhafter. Mit ihnen können folgende Probleme und Limitationen, die einzelne Modelle betreffen, umgangen werden:
Statistisches Problem Reicht der Trainingsdatensatz D_T nicht aus, um den Hypothesenraum $\mathcal{H}$ effektiv zu durchsuchen, können mehrere Hypothesen mit gleichem minimalen Trainingsfehler gefunden werden. Möglicherweise wird während des Lernprozesses eines dieser Hypothesen beziehungsweise dieser Modelle gewählt, welches schlecht generalisiert und einen hohen Validierungsfehler besitzt. Eine gewichtete Kombination solcher Modelle reduziert das Risiko eine falsche Auswahl zu treffen.
Rechnerisches Problem Beruhen Lernprozesse auf heuristischen Methoden, kann nicht immer garantiert werden, dass diese die optimale Hypothese beziehungsweise den minimalen Wert des Trainingslosses finden. So kann ein Modell erzeugt werden, das ein lokales Minimum aber nicht das globale Minimum der Trainingslossfunktion findet. Kombiniert man mehrere solcher Modelle kann das Risiko ein falsches Modell gewählt zu haben wieder reduziert werden.

Repräsentationsproblem Enthält der Hypothesenraum $\mathcal{H}$ eines spezifischen Modells keine geeigneten Repräsentationen der Zielfunktion f, spricht man vom Repräsentationsproblem. Auch diesbezüglich kann durch eine gewichtete Kombination ungeeigneter Modelle eine bessere Repräsentation von f erzielt werden [7].

Modelle, die vom statistischen Problem betroffen sind, sind gekennzeichnet durch eine hohe *Varianz*. Diese sorgt dafür, dass minimale Änderungen in den Merkmalen einer Instanz zu stark unterschiedlichen Klassifikationen dieser führen kann. Grundsätzlich sollte dieses Phänomen bei der Generierung effektiver Modelle vermieden werden. Andererseits werden Modellen, die vom Repräsentationsproblem betroffen sind, die Eigenschaft eines hohen *bias* zugeschrieben. Dies betrifft Modelle, welche für die gegebene Aufgabe weniger geeignet sind. Folglich bieten Ensemble Methoden die Möglichkeit die *Varianz* und den *bias* einzelner Modelle zu verringern, indem mehrere unterschiedliche Modelle trainiert und kombiniert werden [7]. Damit einher gehen eine bessere Generalisierung und genauere Vorhersagen. Zusätzlich können Ensemble Methoden durch die parallele Generierung auf großen oder verteilten Datensätzen angewendet werden, da für das Trainieren einzelner Modelle nur ein Teil der Daten zur Verfügung stehen muss [33]. Im Folgenden wird die Menge an Modellen eines Ensembles mit $\mathcal{H}$ statt mit $\mathcal{H}_{Ens}$ bezeichnet.

2.1.2 Diversität

Um die Unterschiede beziehungsweise die Ähnlichkeiten der Vorhersagen von mehreren Modellen vergleichen zu können, werden Diversitätsmetriken verwendet. Diese Metriken sind für die Bewertung von Ensembles wichtig, da Ensembles von der Diversität ihrer Modelle profitieren [30]. Mittels einer *error-decomposition* kann gezeigt werden, dass der erwartete Fehler mit der Diversität eines Ensembles zusammenhängt [36, S. 100]. Insgesamt lassen sich die Diversitätsmetriken in zwei Kategorien aufteilen.

Paarweise Diversität Eine Metrik für paarweise Diversität weist jedem Paar an Modellen $\forall i, j \in \{1, ..., n\} : (h_i, h_j) \in (\mathcal{H} \times \mathcal{H})$ einen Wert $d_{ij} = div(h_i, h_j)$ zu, wobei div eine gewählte Diversitätsmetrik ist. Die resultierende Matrix $D_{dist} \in \mathbb{R}^{n \times n}$ wird auch als Distanzmatrix bezeichnet. Ein Beispiel für paarweise Diversitätsmetriken ist die *double-fault measure* von Giacinto et Roli [36, S. 105] :

$$d_{ij} = \frac{\sum_{k=1}^{N} \mathbb{1}(h_i(x_k) \neq y_k \wedge h_j(x_k) \neq y_k)}{N} \tag{2.6}$$

Nicht paarweise Diversität Alternativ zur paarweisen Diversität weisen diese Metriken einem einzelnen Modell oder einer Menge an Modellen einen Wert zu. So kann zum Beispiel mit der *Kohavi Wolpert Variance* für das gesamte Ensemble ein Wert für dessen Diversität ermittelt werden:

$$kw = \frac{1}{N \cdot n^2} \sum_{k=1}^{N} \rho(x_k)(n - \rho(x_k)) \tag{2.7}$$

mit $\rho(x_k)$ als Anzahl an Modellen des Ensembles, die x_k korrekt klassifizieren.
Mit einer Vielzahl an Diversitätsmetriken ist es unvermeidlich, dass einige auf ähnliche Weise die Diversität eines Ensembles messen und folglich ähnliche Resultate liefern. So zeigten Kuncheva et Whitaker, dass die beiden Metriken *Kohavi Wolpert Variance* (kw) und *Kappa-Statistic* (κ), welche in Kapitel 4.4.2 vorgestellt wird, auf folgende Weise zusammenhängen:

$$\kappa = 1 - \frac{n}{(n-1)\bar{\rho}(1-\bar{\rho})} kw \tag{2.8}$$

mit $\bar{\rho} = \frac{1}{N \cdot n} \sum_{i=1}^{n} \sum_{k=1}^{N} \mathbb{1}(h_i(x_k) = y_k)$ [36, S. 106-110].

Innerhalb eines Ensembles hängen die Vorhersagegenauigkeiten einzelner Modelle mit der Diversität dieser Modelle zusammen. Je höher die *accuracies* der Modelle ist, desto mehr ähneln sie sich. Dieser Zusammenhang wird *Accuracy-Diversity Tradeoff* genannt und spielt eine zentrale Rolle bei der Generierung effektiver Ensembles [35]. Daher werden zufällige Prozesse, wie das *bootstrapping*-Verfahren beim *Bagging*, in die Ensemblegenerierung integriert, um sowohl diverse als auch akkurate Modelle zu erzeugen.

2.1.3 Entscheidungsbäume und Random Forests

Ein verbreiteter Modelltyp für Klassifikationsprobleme ist der Entscheidungsbaum (engl. *decision tree*). Entscheidungsbäume überzeugen aufgrund ihrer hohen Vorhersagegenauigkeit bei geringem Berechnungsaufwand und wegen ihrer Anwendbarkeit auf großen oder fehlerbehafteten Datensätzen [29].
Sie beruhen auf der Idee durch mehrere Merkmalsabfragen, auch *splits* genannt, die zu klassifizierenden Daten einem Label zuzuordnen. Dabei durchlaufen die Daten die Baumstruktur des Modells und werden an den inneren Knoten (engl. *nodes*), welche die Merkmale abfragen, aufgeteilt. Die resultierenden Submengen werden wiederum an die folgenden Knoten weitergeleitet und aufgeteilt, bis ein Datenpunkt an einem Blatt des Baumes ankommt. Blätter des Baumes entsprechen den Labeln des Klassifikationsproblems und eine Zuordnung eines Datenpunktes zu einem Blatt entspricht einer Vorhersage des Modells. Die baumartige Struktur der Modelle ergibt sich durch die rekursive Generierung dieser. Der wichtigste Faktor bei der Generierung ist das gewählte Kriterium, anhand dessen die

optimalen *splits* beziehungsweise die Merkmalsabfragen ermittelt werden. Beispielhaft wird hierfür der *Gini-Index* (2.9) verwendet:

$$G_{gini}(D; D_1, \ldots, D_k) = I(D) - \sum_{i=1}^{k} \frac{|D_k|}{|D|} I(D_k) \tag{2.9}$$

mit $I(D) = 1 - \sum_{y \in \mathcal{Y}} P(y|D)^2$ und $P(y|D)$ als bedingte Wahrscheinlichkeit. $D_1, ..., D_k$ entspricht der Aufteilung der Daten D an dem Knoten durch den *split* [36, S. 4-6].

Random forests wenden den Ensemble Ansatz auf Entscheidungsbäumen an und sind der verbreitetste Typ der Entscheidungsbaum-Ensembles. Die resultierenden Modelle profitieren von den Vorteilen eines Ensemble Ansatzes und tendieren dazu in Benchmarks alternative Modelle zu übertreffen [29].
Die Menge an Entscheidungsbäumen wird mittels des *bootstrapping* Verfahrens wie beim *Bagging* generiert und die einzelnen Vorhersagen werden durch ungewichtetes *Majority Voting* aggregiert. Jedoch wird ein zusätzlicher Schritt bei der Generierung der Entscheidungsbäume gemacht, um die Wirkung einflussreicher Merkmale der Daten zu reduzieren. Dafür wird bei jeder Ermittlung eines *splits* während der Generierung nur eine zufällig gewählte Untermenge der Merkmale betrachtet. Sowohl das *bootstrapping* Verfahren als auch die zufälligen Auswahlen an Merkmalen tragen zur Erhöhung der Diversität der Modellmenge eines *random forests* bei [29].

Extremely randomized trees erweitern den Diversitätsaspekt von *random forests*, indem mehr Zufälligkeit bei der Generierung der Entscheidungsbäume eingeführt wird. Bei der Wahl von *splits* werden für die Merkmale möglichst optimale Thresholds ermittelt. Thresholds sind konkrete Merkmalsausprägungen, die als Wertegrenzen für die Aufteilung der Daten bezüglich dieses Merkmals dienen. Im Fall von *extremely randomized trees* werden für die Merkmale eine zufällige Auswahl an Threshold-Werten generiert und aus denen die geeignetsten ausgewählt. Weiterhin wird für die Erzeugung der einzelnen Entscheidungsbäume der gesamte Trainingsdatensatz verwendet und kein *bootstrapping* Verfahren wie bei *random forests*. Insgesamt weisen die einzelnen Entscheidungsbäume tendenziell höheren *bias* und *variance* auf verglichen mit denen eines *random forests* [29].

2.2 Ensemble Pruning

Ensemble Methoden unterliegen dem Ansatz mehrere Modelle für ein Problem zu generieren und die Vorhersagen aller zu kombinieren. Dabei spielt die Anzahl an Modellen eine entscheidende Rolle, da mehr Modelle nicht nur mehr Vorhersagen, sondern auch mehr Berechnungen bedeuten. Jedoch trägt eine Erhöhung der Modellanzahl nicht immer zu deutlich besseren Vorhersagen bei: In einer Auswertung von *random forests* Modellen auf 29 Datensätzen konnten ab einer Ensemblegröße von 128 Entscheidungsbäumen keine signifikanten Verbesserungen der AUC Werte festgestellt werden [24]. Insgesamt besteht ein linearer Zusammenhang zwischen der Anzahl an Modellen und der Ausführungsgeschwindigkeit und dem Speicherverbrauch eines trainierten Ensembles, was besonders im Kontext von leistungsschwachen Geräten mit geringem Speicherplatz die Verwendung kleiner Ensembles motiviert.
Mit der intuitiven Idee nur einen Teil der trainierten Modelle für das finale Ensemble auszuwählen, setzt Ensemble Pruning, auch Ensemble Selection genannt, an diese Problemstellung an. Durch ein Ensemble Pruning Verfahren wird ein Subensemble an Modellen bestimmt, welches die voraussichtlich höchste Vorhersagegenauigkeit hat. Für diesen Prozess wird ein Pruning Datensatz D_P benötigt, auf dem die potentiellen Subensembles anhand gewählter Kriterien evaluiert werden. Für den Pruning Datensatz D_P werden für das Ensemble unbekannte Daten verwendet, wobei manche Verfahren auch den Trainingsdatensatz D_T nutzen. Somit kann Ensemble Pruning als zusätzlicher Schritt nach dem Generieren eines Ensembles verstanden werden, dessen Ziel eine Verringerung der Modelle bei Erhaltung der Vorhersagegenauigkeit des Ensembles ist.

Zhou et al. zeigten, dass eine Auswahl an Modellen besser sein kann als das gesamte Ensemble, da generierte Modelle innerhalb des Ensembles ähnliche Vorhersagen treffen können. Unter dieser Annahme konnten sie für Regression und Klassifikation zeigen, dass kleinere Ensembles den erwarteten Fehler nicht verschlechtern. Ein Extremfall dabei ist ein Ensemble aus Modellen, die alle identisch sind. Dieses kann verkleinert werden, ohne die Generalisierung des Ensembles zu verschlechtern [36, S. 120-123].
Diversität im Kontext von Ensembles spielt somit eine bedeutende Rolle. Ein Ensemble aus drei identische Modellen mit einer *accuracy* von jeweils 95% schneidet schlechter ab als ein Ensemble aus drei Modellen mit einer *accuracy* von jeweils 67% und minimalem paarweisen Fehler [35]. Folglich ist es wichtig durch Ensemble Pruning die Diversität der Modelle zu erhalten.

2.2.1 Formale Betrachtung als Optimierungsproblem

Da Ensemble Pruning ein kombinatorisches Auswahlproblem zugrunde liegt, kann es als mathematisches Optimierungsproblem aufgefasst werden. Aus der Menge an Modellen $\mathcal{H}$ der Größe $n = |\mathcal{H}|$ sollen $k \in \mathbb{N} : k \ll n$ Modelle ausgewählt werden, die ein optimales Subensemble bilden. Optimalität bedeutet eine gewählte Lossfunktion ℓ, welche den Fehler des Subensembles misst, zu minimieren.
Die Auswahl und Lösung wird mit einem Vektor $w \in \{0,1\}^n$ modelliert, wobei $w_i = 0$ bedeutet, dass das i-te Modell nicht in der Auswahl enthalten ist, und $w_i = 1$, dass es enthalten ist. Damit kann Ensemble Pruning als Spezialfall einer Kombinationsregel von Modellen mit binären Gewichten angesehen werden. Um die Größe k des Subensembles zu forcieren, wird eine Nebenbedingung eingeführt. Diese misst mit der L1-Norm die Größe der Auswahl w und erzwingt, dass sie k entspricht. Insgesamt ergibt sich das Optimierungsproblem:

$$\arg\min_{w} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\ell \left(\sum_{i=1}^{n} w_i h_i(x), y \right) \right] \qquad \text{s.t. } ||w||_1 = k \tag{2.10}$$

Minimiert man die Lossfunktion auf einem Pruning Datensatz D_P der Größe N, dann:

$$\arg\min_{w} \frac{1}{N} \sum_{j=1}^{N} \ell \left(\sum_{i=1}^{n} w_i h_i(x_j), y_j \right) \qquad \text{s.t. } ||w||_1 = k \tag{2.11}$$

Die Wahl der optimalen Größe k des Subensembles wird als separates Problem betrachtet. Eine verbreitete Möglichkeit die optimale Größe zu ermitteln ist ein *grid search* über ein Intervall an geeigneten Größen. Im Folgenden wird das Ensemble Pruning Problem stets mit fest gewähltem k betrachtet.

2.2.2 Eigenschaften von Ensemble Pruning Verfahren

Mit einem Ensemble der Größe n ergeben sich insgesamt 2^n mögliche Subensembles. Spezifiziert man dabei eine konkrete Größe k ergeben sich $\binom{n}{k}$ mögliche Subensembles, für die jeweils ihre Vorhersagegenauigkeit auf dem Pruning Datensatz D_P ermittelt werden muss. In der Praxis werden Ensembles mit mehreren hundert Modellen generiert, wodurch die Suche nach dem optimalen Subensemble nicht praktikabel ist.
Die NP-schwere dieses kombinatorischen Auswahlproblems erzwingt Ensemble Pruning Verfahren eine approximative Herangehensweise zu wählen. Diese Verfahren werden durch zwei Eigenschaften charakterisiert:

Was wird optimiert? Da Ensemble Pruning ein Optimierungsproblem (2.10) zugrunde liegt, entscheidet eine gewählte Lossfunktion ℓ, welche Subensembles bevorzugt werden. In der Praxis führen Ensemble Pruning Verfahren hierzu häufig Pruning Metriken ein, die einzelne Modelle oder eine Menge an Modellen bewerten und keinen mathematischen Metriken im strengen Sinne entsprechen müssen. So kann eine Pruning Metrik, die zwei Modelle miteinander vergleicht, nicht symmetrisch sein. Statt die Auswahl anhand der Optimierung einer Lossfunktion zu treffen, werden Modelle beziehungsweise Mengen an Modellen ausgewählt, die eine besonders gute Bewertung der jeweiligen Metrik erhalten. Somit übernimmt eine Pruning Metrik die Rolle der Lossfunktion und die Wahl der Pruning Metrik nimmt Einfluss auf das resultierende Subensemble.

Wie wird optimiert? Eine gewählte Lossfunktion beziehungsweise eine Pruning Metrik legt fest, nach welchen Kriterien die Auswahl eines Subensembles getroffen werden soll. Wie diese Auswahl bestimmt und getroffen wird entscheidet ein Algorithmus, der in dem Suchraum an möglichen Subensembles nach einer optimalen Auswahl sucht. Aufgrund der Größe dieser Suchräume kann ein solcher Algorithmus in der Praxis nicht alle Möglichkeiten betrachten und durchsucht den Suchraum in den meisten Fällen approximativ. Deshalb werden diese Algorithmen als Approximationsverfahren bezeichnet. Eine Übersicht über häufige Approximationsverfahren bezogen auf das Ensemble Pruning Problem wird in Kapitel 3 gegeben.

Weitere Parameter von Ensemble Pruning Verfahren wie die Größe k des finalen Subensembles oder die Wahl und Größe des Pruning Datensatzes D_P nehmen durchaus Einfluss auf das Ergebnis und sind wichtige Merkmale. Dennoch werden sie im Folgenden nicht als charakterisierende Eigenschaften von Ensemble Pruning Verfahren aufgefasst.

2.3 Mathematische Optimierung

Mathematische Optimierung befasst sich mit der Modellierung und effizienten Lösung von Optimierungsproblemen. Die allgemeine Form von Optimierungsproblemen entspricht dabei:

$$\begin{aligned} &\min f_0(x) \\ \text{s.t. } &f_i(x) \leq b_i, i = 1, ..., m \\ &f_j(x) = b_j, j = 1, ..., p \end{aligned} \tag{2.12}$$

Vektor $x \in \mathbb{R}^n$ fungiert als Variable, über die optimiert wird. Dabei ist man an einer Eingabe $x^* \in \mathbb{R}^n$ für das Problem interessiert, welche den minimalen Wert der Zielfunktion $f_0 : \mathbb{R}^n \to \mathbb{R}$ liefert und die Nebenbedingungen des Problems einhält. Diese Nebenbedingungen können mit weiteren Funktionen $f_i : \mathbb{R}^n \to \mathbb{R}$ und Kapazitäten $b_i \in \mathbb{R}$ modelliert werden. Dabei unterscheidet man zwischen Nebenbedingungen aus Ungleichungen und aus Gleichungen [3, S. 1] [23, S. 2-3]. Ist man nicht am Wert der Lösung $f(x^*)$ interessiert sondern am Wert der Eingabe x^*, ersetzt man in der Problemformulierung min durch arg min.

Optimierungsprobleme werden anhand der verwendeten Funktionstypen kategorisiert. Ist die Zielfunktion quadratisch und sind die Funktionen der Nebenbedingungen linear, spricht man von einem quadratischen Programm (engl. *quadratic program*):

$$\begin{aligned} \min_x &\frac{1}{2} x^T P x + q^T x + r \\ \text{s.t. } &Gx \preceq h \\ &Ax = b \end{aligned} \tag{2.13}$$

wobei $P \in \mathbb{R}^{n \times n}$, $q \in \mathbb{R}^n$, $r \in \mathbb{R}$, $G \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{p \times n}$ und $h, b \in \mathbb{R}^n$. Die m Ungleichungen und p Gleichungen für Nebenbedingungen werden in einer Matrixschreibweise zusammengefasst [3, S. 152] [23, S. 449].

Zusätzlich zur Einschränkung der Variable $x \in \mathbb{R}^n$ durch Nebenbedingungen können Probleme voraussetzen, dass manche oder alle Einträge von x ganzzahlig sein müssen. Solche Probleme fallen in den Bereich der ganzzahligen Optimierung und quadratische Programme, die solche Beschränkungen verwenden, bezeichnet man als *mixed-integer quadratic program*. Als Beispiel dient das allgemeine quadratische Problem (2.13), wobei die Eingaben für das Problem beziehungsweise die Einträge der Variablen x auf 0 oder 1 beschränkt werden.

$$\begin{aligned} \min_x &\frac{1}{2} x^T P x + q^T x + r \\ \text{s.t. } &Gx \preceq h \\ &Ax = b \\ &x \in \{0, 1\}^n \end{aligned} \tag{2.14}$$

Eine einheitliche Übersicht und effektive Lösungsstrategien für Optimierungsprobleme sind von Interesse, da sie einen Großteil relevanter Probleme ausmachen. Im Kern nahezu jeder Methode des maschinellen Lernens steht ein Optimierungsproblem. Folglich fand und findet die Verwendung von Methoden der mathematischen Optimierung im Bereich des maschinellen Lernens zunehmend an Bedeutung [2].

2.3.1 Konvexität

Konvexität ist eine fundamentale Eigenschaft in der mathematischen Optimierung. Ist die Zielfunktion und der Suchraum eines Optimierungsproblems konvex, ist jedes lokale Minimum ein globales Minimum [23, S. 7-8].
Die Eigenschaft Konvexität bezieht sich sowohl auf Mengen als auch auf Funktionen. Eine Menge $S \subset \mathbb{R}^n$ ist konvex, falls gilt $\forall x, y \in S \wedge \forall \alpha \in [0,1] : (\alpha x + (1-\alpha)y) \in S$. Eine Funktion f ist konvex, falls ihr Definitionsbereich S konvex ist und $\forall x, y \in S \wedge \forall \alpha \in [0,1]$ gilt : $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ [23, S. 8].

Ein quadratisches Programm (2.13) ist konvex, sofern Matrix P positiv semi-definit ist [23, S. 449]. P ist positiv semi-definit, falls für jedes beliebige $x \in \mathbb{R}^n$ gilt : $x^T P x \geq 0$. Besitzt die Zielfunktion eines quadratischen Programms keinen quadratischen Term, sondern nur den Vektor q und eine Konstante r, ist es ein lineares Programm. Lineare Programme sind konvex [3, S. 146].

Kapitel 3

Verwandte Arbeiten

Ensemble Pruning als Methodik, um die Effizienz bereits trainierter Ensemble Modelle zu verbessern, ist ein aktuelles und relevantes Thema. In den letzten 20 Jahren wurde eine Vielzahl an Ensemble Pruning Verfahren vorgestellt. Einen Überblick über diese gegeben sowohl Arbeiten, die ein neues Verfahren vorstellen und dieses einordnen [9, 17, 18, 19, 28, 34, 35], als auch zusammenfassende Werke [29, 30, 36]. Dabei orientieren sich die Kategorisierungen stets an den Approximationsverfahren, welche bei den Pruning Verfahren verwendet werden. Hiernach ergibt sich die verbreitete Kategorisierung:

Greedy-based *Greedy*-Algorithmen versuchen iterativ ein lokales Kriterium optimal zu erfüllen. Im Kontext von Ensemble Pruning bedeutet dies: Das Subensemble wird schrittweise aufgebaut und in jeder Iteration wird das, nach der gewählten Pruning Metrik, optimale und noch nicht ausgewählte Modell dem Subensemble zugewiesen, bis insgesamt k Modelle ausgewählt wurden. Alternativ kann mit dem gesamten Ensemble gestartet und iterativ Modelle aus dem Ensemble entfernt werden, bis die Größe k erreicht wird.
Eine Subkategorie bilden die *ordering-based* Verfahren, welche alle Modelle anhand der Pruning Metrik bewerten, diese sortieren und iterativ die besten k Modelle *greedy* auswählen.

Clustering-based Diese Verfahren zeichnen sich dadurch aus, dass sie die Modelle eines Ensembles anhand einer gewählten Metrik, die die Ähnlichkeit von Modellen misst, kategorisieren. Dafür verwenden sie *Clustering*-Algorithmen, die aus dem Bereich des unüberwachten Lernen stammen. Nach einer Kategorisierung der Modelle können geeignete Repräsentanten gewählt werden, die in das finale Subensemble aufgenommen werden.

Optimization-based In diese Kategorie fallen Verfahren, die das Ensemble Pruning Problem in direkter Weise als Optimierungsproblem aufgreifen. Sie beinhalten zum einen heuristische Optimierungsverfahren wie zum Beispiel evolutionäre Algorithmen, welche durch

zufällige Mutationen und Kombinationen von Modellmengen versuchen die Lossfunktion des Problems zu minimieren. Zum anderen werden auch Ansätze der mathematischen Optimierung, die im Vergleich zu heuristischen Verfahren Gütegarantien besitzen, dieser Kategorie zugeordnet.

Ähnlichkeiten der Ensemble Pruning Verfahren, zum Beispiel in der Wahl des Approximationsverfahrens oder in minimalen Unterschieden bezüglich der gewählten Pruning Metriken, motivierten Tsoumakas et al. in [33] ein theoretisches Framework für *greedy-based* Verfahren zu entwickeln. Mit dem Ziel auf diese Ähnlichkeiten hinzuweisen und die Entwicklung neuartiger Ensemble Pruning Verfahren anzuregen, identifizieren sie vier charakterisierende Merkmale der *greedy-based* Verfahren: die Richtung des *Greedy Search*, also das iterative Auf- oder Abbauen des Subensembles, die verwendete Pruning Metrik für das lokale Kriterium der Modellauswahl, den verwendeten Pruning Datensatz und die fixierte Größe k des finalen Subensembles. Kategorisierungen und Frameworks zu Ensemble Pruning Verfahren helfen doppelte und ähnliche Ansätze zu vermeiden, die sich möglicherweise nur in unwichtigen Merkmalen unterscheiden, und ermöglichen systematische Vergleiche der Verfahren. Das Framework dieser Arbeit setzt an der Idee von Tsoumakas et al. an und entwickelt ein neues, generalisierteres und nicht nur auf *greedy-based* Verfahren beschränktes Ensemble Pruning Framework.
In [26] erweitern die Autoren ihre Ergebnisse um eine empirische Auswertung verschiedener *Greedy Search* Pruning Verfahren durch eine systematische Kombination ihrer im Framework definierten Merkmale. Ihrer Auswertung nach ist die Verwendung des Trainingsdatensatzes als Pruning Datensatz nur bei homogenen Ensembles vorteilhaft, *forward search* als Suchrichtung resultiert in kleineren Subensembles und für die finale Größe sollte das Subensemble mit der höchsten *accuracy* auf dem Pruning Datensatz gewählt werden.

Zhang et al. fassen in [35] Ensemble Pruning als mathematisches Optimierungsproblem auf. Da den heuristischen Verfahren für die Approximation, zum Beispiel dem *Greedy Search* Algorithmus, häufig Gütegarantieren fehlen, wählen sie einen Ansatz mit theoretischem Fundament. Sie formulieren das kombinatorische Auswahlproblem als quadratisches Programm (3.1) und wenden Methoden der semi-definiten Programmierung zur Lösung des Problems an.

$$\begin{aligned} &\min_x x^T \tilde{G} x \\ &\text{s.t.} \sum_i x_i = k,\ x_i \in \{0,1\} \end{aligned} \tag{3.1}$$

Für ihre Pruning Metrik, anhand der das Subensemble ausgewählt wird, führen sie die Matrix $G \in \mathbb{R}^{n \times n}$ ein, wobei $\forall i \in \{1, ..., n\} : G_{ii}$ die Anzahl an Fehler der einzelnen Modelle und $\forall i \in \{1, ..., n\}, \forall j \in \{1, ..., n\}, i \neq j : G_{ij}$ die Anzahl an gemeinsamen Fehlern

aller paarweiser Kombinationen der Modelle misst. n entspricht der Anzahl an Modellen des Ensembles und die Fehler werden auf dem gewählten Pruning Datensatz D_P der Größe N gemessen. Die Matrix G wird normalisiert und die resultierende symmetrische Matrix $\tilde{G}$ wird in das quadratische Problem eingesetzt.

$$\begin{aligned} \tilde{G}_{ii} &= \frac{G_{ii}}{N} \\ \tilde{G}_{ij,\, i \neq j} &= \frac{1}{2}\left(\frac{G_{ij}}{G_{ii}} + \frac{G_{ij}}{G_{jj}}\right) \end{aligned} \tag{3.2}$$

Für die Lösung des Problems bilden sie ihre Formulierung auf das ähnliche Problem „max cut with size k“ (*MC-k*) ab und wenden bestehende Relaxierungstechniken für die Approximation der Lösung an. Im Vergleich zu zwei *greedy-based* Ensemble Pruning Verfahren schneidet ihr Verfahren in ihren empirischen Untersuchungen besser ab. Eine generalisierte Form der von Zhang et al. vorgestellten Sichtweise auf das Problem dient als Basis für das Ensemble Pruning Framework dieser Arbeit.

Kapitel 4

Ensemble Pruning Framework

4.1 Vorstellung des Ansatzes

Das Framework dieser Arbeit differenziert zwischen der verwendeten Pruning Metrik beziehungsweise der Lossfunktion und dem Approximationsverfahren eines Ensemble Pruning Verfahrens. Diese neuartige Betrachtungsweise sieht ein Pruning Verfahren nicht mehr als einen eigenständigen Gesamtprozess und regt so die Kombination unterschiedlicher Merkmale von unterschiedlichen Verfahren an. Weitere Parameter für die Pruning Verfahren wie die Größe k des finalen Subensembles und die Wahl des Pruning Datensatzes D_P werden ebenfalls als Merkmale innerhalb des Frameworks aufgefasst.
Die Verwendung von Pruning Metriken anstelle von Lossfunktionen motiviert die Definition des Ensemble Pruning Problems mit gewählter Pruning Metrik, welches das Auswahlkriterium für Subensembles in die Problemstellung integriert. Das resultierende Auswahlproblem wird als *mixed-integer quadratic programming* Problem (4.1) formuliert. Dies erlaubt sowohl die Darstellung von Pruning Metriken, die einzelne Modelle bewerten wie zum Beispiel *accuracy*, als auch von Pruning Metriken, die die Modelle paarweise bewerten wie viele Diversitätsmetriken.

4.1.1 Definition. Das Ensemble Pruning Problem mit gewählter Pruning Metrik ist ein kombinatorisches Auswahlproblem. Gesucht ist eine Auswahl an Modellen, die die Pruning Metrik minimiert und der Größe k entspricht.

$$\begin{aligned} &\arg\min_{w} \alpha(w^T Q w) + (1-\alpha)(c^T w) \\ &\text{s.t. } w^T \mathbb{1}^n - k = 0,\ w \in \{0,1\}^n \end{aligned} \tag{4.1}$$

Für die Darstellung der Pruning Metriken werden der Vektor $c \in \mathbb{R}^n$ und die Matrix $Q \in \mathbb{R}^{n \times n}$ verwendet und $w \in \{0,1\}^n$ dient als Auswahlvariable für die Modelle. n ent-

spricht der Gesamtzahl an Modellen des Ensembles und k der Größe des finalen Subensembles. Werden für ein spezifisches Pruning Verfahren beide Typen der Pruning Metriken verwendet, können diese Mithilfe des Parameters $\alpha \in [0,1]$ zueinander gewichtet werden. Wird nur ein Typ verwendet, kann entweder die Matrix Q oder der Vektor c mit 0 gefüllt werden. Falls die gewählten Pruning Metriken auf das Maximieren ausgelegt sind, können Q beziehungsweise c mit dem Skalar -1 multipliziert werden, da $\forall w \in \{0,1\}^n, \forall Q \in \mathbb{R}^{n \times n}, \forall c \in \mathbb{R}^n$:

$$\begin{aligned} \arg\max_w w^T Q w &= \arg\min_w w^T((-1) \cdot Q) w \\ \arg\max_w c^T w &= \arg\min((-1) \cdot c)^T w \end{aligned} \tag{4.2}$$

Zu beachten ist, dass die Darstellung des Problems mit einer gewählten Pruning Metrik nicht mehr der allgemeinen Form des Ensemble Pruning Problems in Kapitel 2.2.1 entspricht.

4.2 Optimalität des Ansatzes

Die Formulierung des Problems als quadratisches Programm fasst Ensemble Pruning Verfahren mit verschiedenen Pruning Metriken in einer einheitlichen Übersicht zusammen. Zudem ermöglicht sie alle Ensemble Pruning Probleme mit gewählter Pruning Metrik, die der Definition 4.1.1 entsprechen, als ein konvexes Programm darzustellen. Diese Eigenschaft des Problems hat zur Folge, dass eine lokale Lösung des Problems gleich der globalen Lösung entspricht. Für den Beweis dieser Aussage im Satz 4.2.3 werden zwei äquivalenzerhaltende Transformationen des Problems verwendet, die jeweils in Lemma 4.2.1 und 4.2.2 vorgestellt und anschließend bewiesen werden. Außerdem wird für den Beweis des Satzes 4.2.3 angenommen, dass paarweise Pruning Metriken die Distanz eines Modells zu sich selbst mit 0 bewerten und somit die Diagonale der resultierenden Matrix $Q \in \mathbb{R}^{n \times n}$ leer ist. Falls dies für eine paarweise Pruning Metrik nicht zutrifft, kann der in Gleichung (4.9) beschriebene Schritt angewendet werden.

4.2.1 Lemma. *Mit $Q \in \mathbb{R}^{n \times n} \wedge \forall i \in \{1,...,n\} : q_{ii} = 0$ und $\tilde{Q}$ als Transformation von Q nach Gleichung (4.3) gilt: $\forall x \in \{0,1\}^n : x^T Q x = x^T \tilde{Q} x$.*

Beweis. Sei $x \in \{0,1\}^n$ und $Q \in \mathbb{R}^{n \times n} \wedge \forall i \in \{1,...,n\} : q_{ii} = 0$ beliebig gewählt und $\tilde{Q}$ die nach (4.3) definierte Transformation von Q. Dann:

$$
\begin{aligned}
x^T\tilde{Q}x &= \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} \cdot \begin{pmatrix} \tilde{q}_{11} & \tilde{q}_{12} & \dots \\ \vdots & \ddots & \\ \tilde{q}_{n1} & & \tilde{q}_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \tilde{q}_{ij} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \ \frac{1}{2}(q_{ij} + q_{ji}) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j (q_{ij} + q_{ji}) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ji}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ij} \\
&= x^T Q x
\end{aligned}
$$

■

4.2.2 Lemma. *Mit* $\tilde{Q} \in \mathbb{R}^{n\times n} \wedge \forall i \in \{1,...,n\} : \tilde{q}_{ii} = 0$ *und* $\bar{Q}$ *als Transformation von* $\tilde{Q}$ *nach Gleichung* (4.4) *gilt:* $\forall x \in \{0,1\}^n : x^T\tilde{Q}x = x^T\bar{Q}x - k \cdot k_1$ *, wobei* $k = \sum_{i=1}^{n} x_i$ *und* $k_1 > \max_i(\sum_{j=1}^{n} |\tilde{q}_{ij}|)$ *mit* $i \in \{1,...,n\}$ *und* $k_1 > 0$ *.*

Beweis. Sei $x \in \{0,1\}^n$ und $Q \in \mathbb{R}^{n\times n}$ beliebig gewählt. $\tilde{Q}$ ist die Transformation von Q nach Gleichung (4.3) und $\bar{Q}$ ist die Transformation von $\tilde{Q}$ nach Gleichung (4.4). Dann:

$$
\begin{aligned}
x^T\bar{Q}x \ - k \cdot k_1 &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \bar{q}_{ij} \ - k \cdot k_1 \\
&= \sum_{i=1}^{n}\sum_{j=1, i\neq j}^{n} x_i x_j \bar{q}_{ij} \ + \sum_{i=1}^{n} x_i x_i \bar{q}_{ii} \ - k \cdot k_1 \\
&= \sum_{i=1}^{n}\sum_{j=1, i\neq j}^{n} x_i x_j \bar{q}_{ij} \ + \sum_{i=1}^{n} x_i x_i k_1 \ - k \cdot k_1 \\
&= \sum_{i=1}^{n}\sum_{j=1, i\neq j}^{n} x_i x_j \bar{q}_{ij} \ + k \cdot k_1 \ - k \cdot k_1 \\
&= \sum_{i=1}^{n}\sum_{j=1, i\neq j}^{n} x_i x_j \bar{q}_{ij} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \tilde{q}_{ij} \\
&= x^T\tilde{Q}x
\end{aligned}
$$

■

4.2.3 Satz. *Das Ensemble Pruning Problem mit gewählter Pruning Metrik ist konvex.*

Beweis. Es werden 3 Fälle unterschieden: (1) Es wird eine Pruning Metrik für einzelne Modelle verwendet, (2) es wird eine Pruning Metrik für paarweise Kombinationen an Modellen verwendet oder (3) eine Kombination aus (1) und (2). Für alle Fälle gilt: $\mathcal{H} = \{h_i \,|\, i \in \{1, \ldots, n\}\}$ entspricht der Modellmenge und D_P dem Pruning Datensatz.

(1) Sei $\forall i \in \{1, \ldots, n\} : metric(h_i, D_P) = c_i$ mit $metric : (\mathcal{H} \times \mathcal{D}) \to \mathbb{R}$ und $c = (c_1, ..., c_n)^T$. Mit $x \in \{0,1\}^n$ als Auswahlvariable ergibt sich das lineare Programm:

$$\begin{aligned} &\arg\min_x c^T x \\ &\text{s.t. } x^T \mathbb{1}^n - k = 0,\ x \in \{0,1\}^n \end{aligned}$$

Da das Programm linear ist, ist es konvex.

(2) Sei $\forall i, j \in \{1, \ldots, n\} : metric(h_i, h_j, D_P) = q_{ij}$ mit $metric : (\mathcal{H} \times \mathcal{H} \times \mathcal{D}) \to \mathbb{R}$. Notiere alle q_{ij}'s als $Q \in \mathbb{R}^{n \times n}$. Es wird angenommen, dass $\forall h_i \in \mathcal{H} : metric(h_i, h_i, D_P) = 0$.

Falls *metric* symmetrisch ist und somit $\forall h_1, h_2 \in \mathcal{H} : metric(h_1, h_2, D_P) = metric(h_2, h_1, D_P)$ gilt, setze $\tilde{Q} = Q$. Falls *metric* nicht symmetrisch ist, transformiere Q zur symmetrischen Matrix $\tilde{Q}$, wobei:

$$\forall i, j \in \{1, \ldots, n\} : \tilde{q}_{ij} = \frac{1}{2}(q_{ij} + q_{ji}) \tag{4.3}$$

Wähle $k_1 > \max_i(\sum_{j=1}^{n} |\tilde{q}_{ij}|)$ mit $i \in \{1, ..., n\}$ und $k_1 > 0$. Transformiere $\tilde{Q}$ zur symmetrischen und stark diagonal-dominanten Matrix $\bar{Q}$, wobei:

$$\begin{aligned} \forall i \in \{1, \ldots, n\} : \bar{q}_{ii} &= k_1 \\ \forall i, j \in \{1, \ldots, n\},\ i \neq j : \bar{q}_{ij} &= \tilde{q}_{ij} \end{aligned} \tag{4.4}$$

Weil $\bar{Q}$ reell, symmetrisch und stark diagonal-dominant ist, ist sie positiv definit [12, S. 403]. Eine positiv definite Matrix ist auch positiv semi-definit. Mit $x \in \{0,1\}^n$ als Auswahlvariable und dem Ausgleichterm $-k \cdot k_1$ für Transformation (4.4) ergibt sich das quadratische Programm:

$$\begin{aligned} &\arg\min_x x^T \bar{Q} x - k \cdot k_1 \\ &\text{s.t. } x^T \mathbb{1}^n - k = 0,\ x \in \{0,1\}^n \end{aligned}$$

Da $\bar{Q}$ positiv semi-definit ist, ist das quadratische Programm konvex.

(3) Sei $\forall i \in \{1,\dots,n\} : metric_1(h_i, D_P) = c_i$ mit $metric_1 : (\mathcal{H} \times \mathcal{D}) \to \mathbb{R}$ und $\forall i,j \in \{1,\dots,n\} : metric_2(h_i, h_j, D_P) = q_{ij}$ mit $metric_2 : (\mathcal{H} \times \mathcal{H} \times \mathcal{D}) \to \mathbb{R}$. Notiere die c_i's als $c \in \mathbb{R}^n$ und alle q_{ij}'s als $Q \in \mathbb{R}^{n \times n}$. Es wird angenommen, dass $\forall h_i \in \mathcal{H} : metric_2(h_i, h_i, D_P) = 0$.

Transformiere Q zu $\bar{Q}$ wie in (2). Mit $x \in \{0,1\}^n$ als Auswahlvariable und dem Ausgleichterm $-k \cdot k_1$ für Transformation (4.4) ergibt sich das quadratische Programm:

$$\arg\min_x x^T \bar{Q} x + c^T x - k \cdot k_1$$
$$\text{s.t. } x^T \mathbb{1}^n - k = 0,\ x \in \{0,1\}^n$$

Da $\bar{Q}$ positiv semi-definit ist, ist das quadratische Programm konvex.

Damit ist das Ensemble Pruning Problem mit gewählter Pruning Metrik konvex.

■

4.3 Approximationsverfahren des Frameworks

Aufbauend auf der Differenzierung von Pruning Metriken und Approximationsverfahren, betrachtet das Framework dieser Arbeit eine Auswahl beider dieser Eigenschaften. Mit dem Interesse an einem systematischen Vergleich dieser Merkmale und mit der Möglichkeit in Zukunft weitere in das Framework zu integrieren, wurden folgende Approximationsverfahren für das Framework ausgewählt:

Mathematische Optimierung Das primäre Approximationsverfahren des Frameworks sind Methoden der mathematischen Optimierung, da diese mit der Darstellung des Problems dieser Arbeit einhergehen. In Kapitel 5.1.1 wird genauer auf die verwendeten Methoden und Software-Bibliotheken eingegangen.
Wie in Kapitel 4.2 gezeigt wurde, ist die verbreitetste Form des Ensemble Pruning Problems von konvexer Natur. Daher ist die Verwendung von Approximationsverfahren, die von dieser Eigenschaft Gebrauch machen können, vorteilhaft. Die Methoden der mathematischen Optimierung liefern auch in nicht konvexen Problemen Approximationen der Lösung mit vernachlässigbar kleinem Fehler.

Greedy Search Als weiteres Verfahren wurde der *Greedy Search* Algorithmus gewählt, da dieser in bestehenden Ensemble Pruning Verfahren häufige Verwendung findet. Damit ist ein Vergleich zwischen den Methoden der mathematischen Optimierung und dem *Greedy Search* Algorithmus von Interesse.

Nach den Untersuchungen von Martínez-Muños et. al. in [21] sind *ranking-based* Ensemble Pruning Verfahren, die der *greedy-based* Kategorie angehören, mit leistungsaufwändigeren Verfahren bezüglich Genauigkeit und Robustheit vergleichbar. In [5] und [18] wurden modifizierte *Greedy Search* Algorithmen für das Pruning Problem vorgestellt, die mehr Zufälligkeit integrieren und dadurch einen größeren Suchraum durchlaufen. Dies hat zur Folge, dass mehr mögliche Lösungen betrachtet werden und es unwahrscheinlicher ist, dass sich der Algorithmus in einem lokalen Minimum verläuft. Das Framework dieser Arbeit umfasst jedoch nur den *Greedy Search* Algorithmus ohne Zufälligkeit.
Weiterhin ist zu betonen, dass sowohl Methoden der mathematischen Optimierung als auch der *Greedy Search* Algorithmus für die Variante des Problems mit einer Pruning Metrik, welche einzelne Modelle bewertet, die optimale Lösung finden.

Zufällige Auswahl Als Kontrollverfahren dient ein drittes Approximationsverfahren, welches das Subensemble zufällig auswählt. Wird mit k die Größe des Subensembles spezifiziert, werden k der n Modelle unabhängig und ohne Zurücklegen gezogen.
Die Ergebnisse dieses Verfahrens werden als Basisrichtline verwendet. Mit einem Vergleich der Ergebnisse der weiteren Approximationsverfahren kann beurteilt werden, wie effektiv die Verfahren in der Auswahl des Subensembles sind und ob ein wesentlicher Unterschied zwischen einer gezielten und einer zufälligen Auswahl besteht. Je größer das gewählte k ist, desto wahrscheinlicher ist es, dass sich diese Lücke schließt.

4.4 Pruning Metriken des Frameworks und Kategorisierung

Neben dem gewählten Approximationsverfahren charakterisieren auch die Pruning Metriken die Pruning Verfahren. Das Framework dieser Arbeit betrachtet demnach eine Auswahl an Pruning Metriken, welche von bestehenden Ensemble Pruning Verfahren verwendet werden, und vergleicht diese miteinander. Die Kategorisierung der Pruning Metriken und damit indirekt der Pruning Verfahren orientiert sich an der Aufteilung des Problems in die drei Teilprobleme, die sich aus der Definition des Ensemble Pruning Problems mit gewählter Pruning Metrik ergeben. Folglich weicht die Kategorisierung der Ensemble Pruning Verfahren von der verbreiteten Kategorisierung anhand des Approximationsverfahrens ab.

4.4.1 Pruning Metriken für einzelne Modelle

Guo et al. präsentieren in [9] ihr *ranking-based* Ensemble Pruning Verfahren, welches sich auf ihre Resultate aus der *margin theory* stützt. Dafür betrachten sie die Werte des *margin of examples*, welche für jeden Datenpunkt des Pruning Datensatzes sowohl die *confidence*

als auch die Korrektheit der Vorhersage des gesamten Ensembles beinhaltet. Mit Berücksichtigung des *margin of examples*, der Relevanz von hoher *accuracy* und dem Beitrag eines Modells zur Diversität des Ensembles definieren sie ihre Pruning Metrik *margin & diversity based measure* (MDM) für einzelne Modelle. Der Term $margin(x_i)$ repräsentiert den *margin of examples* auf Instanz x_i und $f_d(h, x_i)$ h's Beitrag zur Diversität über x_i. Der Parameter α wird für die Implementation auf 0.2 gesetzt, da dies in den Untersuchungen von Guo et al. die Subensembles mit den höchsten *accuracies* erzeugte.

$$MDM(h, H) = \sum_{x_i \in D_P} [\mathbb{1}(h(x_i) = y_i)(\alpha f_m(x_i) + (1 - \alpha) f_d(h, x_i))] \tag{4.5}$$

mit $f_m(x_i) = \log |margin(x_i)|$ und $f_d(h, x_i) = \log \frac{v_{y_i}^{(i)}}{n}$, wobei $v_{y_i}^{(i)}$ die Anzahl an Vorhersagen für Label y_i auf x_i ist.

Lu et al. definierten in [19] eine Pruning Metrik *individual contribution* (IC) für einzelne Modelle eines Ensembles und nutzen ein *ranking-based* Verfahren, um die Modelle auszuwählen. Sie zeigen, dass Modelle mit einer höheren *accuracy* und mit mehr korrekten Vorhersagen für Datenpunkte in der Minoritätsgruppe wichtiger für die Konstruktion des Subensembles sind. Datenpunkte in der Minoritätsgruppe sind welche, für die die Mehrzahl der Modelle eine falsche Vorhersage getroffen hat. Ihre Ergebnisse finden sich in ihrer Metrik IC wieder, da der Term für eine korrekte Vorhersage auf einer Instanz x_j der Minoritätsgruppe $(2v_{max}^{(j)} - v_{h_i(x_j)}^{(j)})$ das Modell h_i für diese Instanz besser bewertet. Die Parameter $\alpha_{ij}, \beta_{ij}, \theta_{ij} \in \{0, 1\}$ regeln die exklusiven Fälle: $(\alpha_{ij} = 1) : h_i(x_j) = y_j$ und x_j war in der Minoritätsgruppe oder $(\beta_{ij} = 1) : h_i(x_j) = y_j$ und x_j war in der Majoritätsgruppe oder $(\theta_{ij} = 1) : h_i(x_j) \neq y_j$.

$$IC(h_i) = \sum_{j=1}^{N} (\alpha_{ij}(2v_{max}^{(j)} - v_{h_i(x_j)}^{(j)}) + \beta_{ij} v_{sec}^{(j)} + \theta_{ij}(v_{correct}^{(j)} - v_{h_i(x_j)}^{(j)} - v_{max}^{(j)})) \tag{4.6}$$

Der Term $v_{y_k}^{(j)}$ steht auch hier für die Anzahl an Vorhersagen für das Label k auf Instanz x_j. max, sec und $correct$ stehen für die Labels mit den meisten Stimmen, den zweitmeisten Stimmen und für das richtige Label.

Zusätzlich zu den genannten Pruning Metriken betrachtet das Framework auch *accuracy* und AUC als auswählbare Option. Diese Kategorie umfasst daher Metriken, die entweder auf eine direkte Weise die Vorhersagegenauigkeit der einzelnen Modelle messen oder mehrere Kriterien wie Vorhersagegenauigkeit und Diversität in einem Wert zusammenfassen.

4.4.2 Pruning Metriken für Modellpaare

Eine der ersten Ensemble Pruning Verfahren wurde von Margineantu et Dietterich in [20] vorgestellt, welches sich ausschließlich auf die Diversität der Modelle des Ensembles konzentriert. Sie verwenden die *Kappa-Statistic* (κ) und berechnen diese für alle Paare an Modellen. In iterativen Schritten wählen sie die Modellpaare für das finale Subensemble aus, welche den kleinsten Wert bezüglich der *Kappa-Statistic* besitzen.

$$\begin{gathered} \kappa = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \\ \Theta_1 = \frac{\sum_{i=1}^{L} C_{ii}}{N}, \quad \Theta_2 = \sum_{i=1}^{L} \left(\sum_{j=1}^{L} \frac{C_{ij}}{N} \cdot \sum_{j=1}^{L} \frac{C_{ji}}{N} \right) \end{gathered} \tag{4.7}$$

mit L als Anzahl der Labels und C_{ij} für die Anzahl an Instanzen x, für die $h_a(x) = y_i$ und $h_b(x) = y_j$ zutrifft. h_a und h_b ist das Modellpaar, für das κ berechnet wird.

Lazarevic et Obradovic verfolgen in [15] einen *clustering-based* Ansatz für ihr Ensemble Pruning Verfahren. Mittels des *k-means clustering* Algorithmus aus dem Bereich des unüberwachten Lernens und einer Distanzmetrik bestehend aus der *Kappa-Statistic* (κ) und dem *correlation coefficient* (ρ) clustern sie die Modelle nach ihrer Ähnlichkeit. Mittels weiteren Schritten wie das Entfernen von redundanten Modellen bestimmen sie das Subensemble, das zurückgegeben wird.

Insgesamt fünf paarweise Diversitätsmetriken zu kombinieren, um so eine einheitlichere Form der Diversität zu erreichen, ist die Kernidee des Ensemble Pruning Verfahrens von Cavalcanti et al. in [4]. Dafür wählen sie die fünf Metriken *disagreement measure* (dis), *Q-Statistic* (Q), *correlation coefficient* (ρ), *Kappa-Statistic* (κ) und *double-fault measure* (DF) aus und addieren diese gewichtet zur Matrix $H \in \mathbb{R}^{n \times n}$ auf. Da die Gewichtung der einzelnen Diversitätsmetriken durch einen evolutionären Algorithmus ermittelt wird, werden in der Implementation für das Framework alle Metriken gleich gewichtet. Mittels aus der Matrix H abgeleiteten Graphen und Graphfärbungen ermitteln sie das finale Subensemble. Zur Berechnung der einzelnen Metriken pro Modellpaar wird eine Kontingenztabelle berechnet, dessen vier Felder folgende Fälle zählen: (a) die Anzahl an Beispielen, die beide Modelle korrekt klassifizieren, (b) die nur das erste Modell korrekt klassifiziert, (c) die nur das zweite Modell korrekt klassifiziert und (d) die beide falsch klassifizieren.

$$dis = \frac{b+c}{N} \quad Q = \frac{ad-bc}{ad+bc} \quad \rho = \frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}} \quad DF = \frac{d}{N} \tag{4.8}$$

4.4.3 Kombination beider Typen

In Kapitel 3 wurde der Ensemble Pruning Ansatz von Zhang et. al. [35] bereits vorgestellt. Obwohl sie für die Darstellung ihrer Pruning Metrik nur eine Matrix $\tilde{G}$ verwenden, enthält ihre Matrix sowohl Werte für die Vorhersagegenauigkeit einzelner Modelle als auch für die Diversität paarweiser Modellpaare. Die Diversitätsmetrik entspricht dabei einer normalisierten Form der *double-fault measure* von Giacinto et Roli (2.6) und die Metrik für die Vorhersagegenauigkeit ist der empirische Fehler beziehungsweise *accuracy*. Für das Framework dieser Arbeit werden diese Metriken getrennt betrachtet, indem die Diagonale der Matrix $\tilde{G}$ als separater Vektor c aufgefasst wird.

$$c = \begin{pmatrix} \tilde{g}_{11} \\ \tilde{g}_{22} \\ \vdots \\ \tilde{g}_{nn} \end{pmatrix} \qquad Q = \begin{pmatrix} 0 & \tilde{g}_{12} & \cdots & \tilde{g}_{1n} \\ \tilde{g}_{21} & 0 & & \\ \vdots & & \ddots & \\ \tilde{g}_{n1} & & & 0 \end{pmatrix} \tag{4.9}$$

Xu et al. wählen in [34] die gleiche Pruning Metrik und Darstellung des Problems wie Zhang et al. in [35], verwenden jedoch einen alternativen Weg die Lösung des quadratischen Programms zu approximieren. Sie relaxieren das Auswahlproblem zu einem *constrained eigenvector problem* und wenden einen Algorithmus, der garantiert global konvergiert und effizienter als der Ansatz von Zhang et al. ist, zur Approximation der Lösung an.

Als zusätzliche Pruning Metrik dieser Kategorie wird eine Kombination der *Kappa-Statistic* mit der *accuracy* betrachtet. Zwar wurde die *Kappa-Statistic* bereits in den Pruning Metriken in [4, 15, 20] verwendet, jedoch konzentrieren sich diese Metriken ausschließlich auf die Diversität der Modelle und beziehen die Vorhersagegenauigkeiten der Modelle nicht ein.

4.5 Übersicht und Limitationen des Frameworks

Für diese Arbeit wurde eine Auswahl an Ensemble Pruning Verfahren betrachtet, deren Eigenschaften einen ersten Überblick über die unterschiedlichen Ansätze geben. Die Differenzierung von Pruning Metrik und Approximationsverfahren hebt dabei die Betrachtungsweise dieser Arbeit von bisherigen Werken zu Ensemble Pruning Verfahren hervor. Die Tabelle 4.1 gibt eine Übersicht über die betrachteten Verfahren und die in das Framework implementierten Metriken. Dennoch ist das Framework dieser Arbeit bezüglich dessen Flexibilität beschränkt, sodass nicht alle Ansätze für neuartige Verfahren integriert werden können. Im Folgenden werden Beispiele aufgegriffen, die diese Limitationen illustrieren.

Autoren	Problemtyp	Pruning Metrik	Approximation
Guo et al.	$\arg\min_w c^T w$	*MDM*	*ordering-based*
Lu et al.		*IC*	*ordering-based*
-		*accuracy*	-
-		*AUC*	-
Cavalcanti et al.	$\arg\min_w w^T Q w$	5 Diversitäten	*optimization-based*
Dietterich et al.		*Kappa-Statistic*	*ordering-based*
Lazarevic et al.		*Kappa+correlation*	*clustering-based*
Xu et al.	$\arg\min_w w^T Q w + c^T w$	*double-fault*+Error	*optimization-based*
Zhang et al.		*double-fault*+Error	*optimization-based*
-		*Kappa+accuracy*	-

Tabelle 4.1: Übersicht und Kategorisierung von Ensemble Pruning Verfahren

In [28] formulieren Qian et al. das Ensemble Pruning Problem als multiobjektives Optimierungsproblem, indem sie die Maximierung der Vorhersagegenauigkeit und die Minimierung der Größe des Subensembles gleichzeitig betrachten. In bisherigen Verfahren und in der Repräsentation des Problems innerhalb dieser Arbeit wird die Ermittlung der optimalen Größe k als separates Problem aufgefasst und typischerweise durch das Ausprobieren verschiedener Werte gelöst. Das von Qian et al. vorgeschlagene Verfahren approximiert durch einen evolutionären Algorithmus die *Pareto-Front* des Problems, welche einer Auswahl an optimalen Subensembles entspricht. Diese sind für ihr Verhältnis von Vorhersagegenauigkeit und Größe des Subensembles jeweils optimal, da sie von keinen anderen Subensembles bezüglich dieser Eigenschaften *dominiert* werden. Aus dieser Menge muss jedoch ein finales Subensemble ausgewählt werden, das den Anforderungen des Verwendungszwecks, zum Beispiel bezüglich Größe und Speicherplatz, entspricht. Sowohl die Integration der Wahl von k in das Auswahlproblem als auch die Approximation durch einen evolutionären Algorithmus sind mit dem Framework dieser Arbeit nicht umsetzbar.

Auch dynamische Ensemble Pruning Ansätze, die zu jeder Vorhersage des Ensembles ein neues Subensembles bestimmen, können nicht integriert werden, da Ensemble Pruning in dem Framework dieser Arbeit als einmalige und permanente Reduzierung der Modellmenge aufgefasst wird. In dem Ansatz von Nguyen et al. in [22] werden für die Vorhersage durch ein Ensemble nur die Modelle ausgewählt, deren *confidence* für diese spezifische Vorhersage einen Schwellenwert überschreiten. Diese dynamische Herangehensweise ist nicht mit

dem Framework umsetzbar. Jedoch kann die *confidence* eines Modells auf dem Pruning Datensatz in den Pruning Prozess integriert werden, wie es zum Beispiel in dem Ansatz von Guo et al. in [9] der Fall ist.

Als Beispiel für eine Pruning Metrik, die nicht in das Framework integrierbar ist, dient die *uncertainty weighted accuracy* von Partalas et al. in [25], da diese explizit auf die Verwendung mit dem *Greedy Search* Algorithmus ausgelegt ist. Sie bewertet das Verhältnis der Vorhersagen einzelner Modelle mit denen des aktuellen Subensembles und bezieht Faktoren wie die Unsicherheit des Subensembles ein. Folglich kann diese Metrik nicht für alternative Approximationsverfahren verwendet wenden, da die Bewertung von paarweisen Kombinationen einzelner Modelle nicht möglich ist und somit eine Integration in das Framework keinen Mehrwert bietet.
Ebenso müssen die in Kapitel 4.3 erwähnten erweiterten *Greedy Search* Algorithmen, die zusätzliche randomisierte Schritte integrieren, als eigenständige Approximationsverfahren betrachtet werden und können nicht mit den Möglichkeiten des Frameworks umgesetzt werden.

Zusammenfassend wurden Fälle gezeigt, die spezifische Besonderheiten von Ensemble Pruning Verfahren demonstrieren und nicht in das Framework dieser Arbeit integrierbar sind. Dies betrifft die Erweiterung des Auswahlproblems um eine variable Größe, die Kategorie der dynamischen Ensemble Pruning Verfahren und spezielle Pruning Metriken, die nur für bestimmte Approximationsverfahren geeignet sind. Mit einer Erweiterung des Frameworks, zum Beispiel bezüglich zusätzlichen oder angepassten Approximationsverfahren, könnten weitere Charakteristiken von Ensemble Pruning Verfahren integriert und damit verglichen werden. Jedoch zeigen die genannten Beispiele, dass eine Diversität unter den Verfahren und ihren Herangehensweisen an das Problem existiert.

Kapitel 5

Umsetzung

5.1 Implementierung

Die Implementierung des Frameworks und der in Kapitel 4 beschriebenen Merkmale von Ensemble Pruning Verfahren sind als Python-Modul realisiert. Die Programmiersprache Python findet mit ihren zahlreichen Software-Bibliotheken im Bereich des maschinellen Lernens weite Verbreitung. Ziel der Implementation ist es die Hauptfunktionalität des Frameworks - das Berechnen des *pruned* Subensembles - unabhängig von den zugrundeliegenden Modellimplementationen der Software-Bibliotheken zu realisieren.
Für die Testumgebung und die Untersuchungen dieser Arbeit werden Ensemblemodelle der *scikit-learn*-Bibliothek [27] verwendet. Mit geringem Aufwand kann das Framework erweitert werden, sodass es weitere Modellimplementationen unterstützt. Weitere *dependencies* für das Framework und die Testumgebung umfassen: „numpy“ [11] für effiziente Datenoperationen, „joblib“ [14] für die Parallelisierung von Prozessen und die „Gurobi Solver Suite“ [10] für das effiziente Lösen von mathematischen Optimierungsproblemen.

Für die Verwendung des Frameworks wird das Python-Modul mittels `import prune_util` eingebunden. Um ein Ensemblemodell zu *prunen*, wird es zusammen mit den Parametern für den Pruning Prozess der jeweiligen implementationsspezifischen Pruning Methode übergeben. Zum Stand dieser Arbeit existiert nur eine Methode `prune_sklearn` für *scikit-learn*-Modelle.
Diese implementationsspezifische Methode berechnet für die einzelnen Modelle des Ensembles die Vorhersagen auf dem Pruning Datensatz `X, y` in `predictions` und `votes`. Diese werden zusammen mit den restlichen Parameter an die interne Pruning Methode `prune` übergeben, welche letztendlich das Subensemble bestimmt. Da diese Methode nicht mehr mit Modell-Objekten arbeitet, funktioniert sie unabhängig der zugrundeliegenden Modellimplementation und gibt ein Array mit den Indices der ausgewählten Modelle zurück. Mit diesem Array wird die Menge an Modellen des übergebenen Ensembles angepasst. Insge-

samt fungiert die Methode `prune` als ein Interface für das Ensemble Pruning Framework. Um weitere Bibliotheken des maschinellen Lernens zu unterstützen, muss folglich nur eine weitere implementationsspezifische Pruning Methode, welche die Vorhersagen der Modelle berechnet und `prune` aufruft, ergänzt werden.

```
def prune_sklearn(sklearn_model, X, y, n_labels, metric, optimizer, p):
def prune(predicions, y, votes, metric, optimizer, p):
```

Um ein Pruning Prozess für ein *scikit-learn* Ensemble zu spezifizieren, werden für `metric` und `optimizer` zwei Strings und für `p` ein $p \in (0, 1)$ für die finale Ensemblegröße übergeben. Zur Auswahl der Pruning Metrik `metric` und des Approximationsverfahrens `optimizer` stehen:

```
["cavalcanti2016", "guo2018", "lazarevic2001", "lu2010", "margineantu1997",
    "zhang2006", "accuracy", "accuracyAndKappa", "auc"]
["greedy", "miqp", "rand"]
```

5.1.1 Approximationsverfahren

Für die zwei zentralen Approximationsverfahren des Frameworks sind jeweils drei Varianten implementiert, da abhängig von der gewählten Pruning Metrik unterschiedliche Problemtypen gelöst werden müssen. Die Kategorisierung folgt dabei der Aufteilung des Ensemble Pruning Problems mit gewählter Pruning Metrik im Beweis des Satzes 4.2.3.

Das erste Approximationsverfahren des Frameworks beruht auf Methoden der mathematischen Optimierung. Die Lösungen der *mixed-integer quadratic programming* Probleme, welche sich nach der Definition 4.1.1 ergeben, werden mittels der Gurobi Solver Suite [10] approximiert, die für diese Optimierungsprobleme den Simplex-Algorithmus verwendet. Dafür wird für den entsprechenden Problemtyp ein Gurobi Modell generiert und mittels `addVars`, `setObjective` und `addConstr` bezüglich den Auswahlvariablen, der Zielfunktion und den Nebenbedingungen konfiguriert. Im Folgenden wird die zweite Variante des Ensemble Pruning Problems mit einer paarweiser Pruning Metrik und der daraus resultierenden Matrix $Q \in \mathbb{R}^{n \times n}$ vorgestellt.

```
def solve_miqp_2(Q, p):
    n = len(Q[0])  # Anzahl an Modellen
    k = int(p * n)  # Anzahl an auszuwaehlenden Modellen

    model = Model('quadratic')
    model.modelSense = GRB.MINIMIZE
    w = model.addVars(n, vtype=GRB.BINARY)
    model.setObjective(quicksum(w[i]*w[j]*Q[i][j] for i in range(n) for j
    in range(n)))
```

```
    model.addConstr(quicksum(w[i] for i in range(n)) == k)
    model.optimize()

    return [i for i in range(n) if w[i].x >= 0.99]
```

Auch für das *greedy-based* Approximationsverfahren sind drei Varianten implementiert, die für jeden Problemtyp unterschiedlich vorgehen. Die erste Variante sortiert die Modelle anhand ihrer Metrikwerte und gibt die k Indices der Modelle mit den besten Werten zurück. Variante 2 verwendet die paarweise Pruning Metrik, um iterativ einzelne Modelle dem Subensemble via dem *Greedy Search* Verfahren hinzuzufügen. Die dritte Variante bezieht beide Metriktypen in das *Greedy Search* Verfahren mit ein. Im Folgenden wird die Methode für die zweite Variante vorgestellt, der folgende Parameter übergeben werden: `predictions` als $n \times m$ Array mit den Vorhersagen aller n Modelle des Ensembles auf allen m Instanzen des Pruning Datensatzes, `metric` als Methode für die paarweise Pruning Metrik, `y` als korrekte Vorhersagen, `p` als finale Größe und `ordering` als Variable für Maximierung oder Minimierung der Metrikwerte. Die interne Methode `aggregate_votes` kombiniert die Vorhersagen der mit `list(S)` übergebenen Modelle des aktuellen Subensembles via *Majority Voting*.

```
def solve_greedy_2(predictions, metric, y, p, ordering):
  # Schritt 1: Initialisieren
  k = int(len(predictions) * p) # Anzahl an auszuwaehlenden Modellen
  H = set(range(len(predictions)))  # Auswahlmenge an Modellen
  S = set() # aktuelles Subensemble

  # Schritt 2: Waehle Modell mit hoechster accuracy auf Pruning Daten
  # (Standardoption fuer Variante 2)
  accuracies = []
  for i in range(len(predictions)):
    accuracies.append(accuracy_score(y, predictions[i]))
  chosen = np.argmax(accuracies)
  S.add(chosen)
  H.remove(chosen)

  # Schritt 3: Fuege iterativ k-1 Modelle via Greedy Search zu S hinzu
  for i in range(k-1):
    # Vorhersage des aktuellen Subensembles
    predictionS = aggregate_votes(predictions, list(S))

    # bestimme aus Menge H das Modell mit hoechstem bzw niedrigstem Wert
    order = []
    scores = []
    copyH = H.copy()
    for j in range(len(H)):
```

```
      current = copyH.pop()
      order.append(current)
      scores.append(metric(predictions[current], predictionS, y))
    zipped = zip(order, scores)
    zipped = sorted(zipped, key = lambda t: t[1])
    indices, values = zip(*zipped)

    if(ordering == "maximize"):
      S.add(indices[len(H)-1])
      H.remove(indices[len(H)-1])
    elif(ordering == "minimize"):
      S.add(indices[0])
      H.remove(indices[0])

  return list(S)
```

Als zusätzliches Approximationsverfahren wurde die Methode `solve_random(n, p)` implementiert, welche eine zufällige Auswahl an $n \cdot p$ Modellindices zurückgibt und zum Bewerten der Effektivität der Pruning Verfahren verwendet wird.

5.1.2 Pruning Metriken

Die Methoden für die Berechnung der Pruning Metriken werden aufgeteilt in zwei Kategorien: Methoden für Metriken, die ein einzelnes Modell bewerten, und Methoden für Metriken, die paarweise Kombinationen von Modellen bewerten. Die Namen dieser beginnen, der Definition des quadratischen Optimierungsproblems für Ensemble Pruning (4.1) entsprechend, mit c beziehungsweise q und enthalten den Autornamen und das Veröffentlichungsdatum. Für die Methoden der ersten Kategorie werden neben den Vorhersagen des Modells auf dem Pruning Datensatz in `predictions` und den korrekten Vorhersagen `y` auch die Anzahl an Modellen `n` und ein zweidimensionales Array `votes` als Parameter übergeben. Dieses Array enthält für jede Instanz des Pruning Datensatzes die Anzahl an Vorhersagen für jedes mögliche Label und wird mit der Methode `get_initial_votes` berechnet. `votes[j,k]` entspricht somit der Anzahl an Vorhersagen für das k-te Label auf der j-ten Instanz. Für die Methoden der paarweisen Metriken reichen die Vorhersagen beider Modelle und die korrekte Vorhersage als Parameter aus. Insgesamt ergeben sich folgende Signaturen für die zwei Typen an Methoden:

```
def c_authorYear(predictions, y, n, votes):
def q_authorYear(predictions_h1, predictions_h2, y):
```

Für die *Kappa-Statistic* wurde die Implementation `sklearn.metrics.cohen_kappa_score` verwendet. Ebenso stützen sich die Berechnungen für *accuracy* und *AUC* auf Methoden in `sklearn.metrics`. Die restlichen Methoden für Pruning Metriken sind Nachimplementationen, die sich auf die Beschreibungen der jeweiligen Werke stützen.

Im Falle des Approximationsverfahrens mit dem *gurobi*-Solver müssen Vektoren $c \in \mathbb{R}^n$ oder Matrizen $Q \in \mathbb{R}^{n \times n}$ berechnet werden. Bereits mit einer Ensemblegröße von 128 Entscheidungsbäumen ergeben sich 16384 Einträge in Matrix Q. Die Berechnung dieser Metrikwerte, für die alle Vorhersagen der Modelle auf dem Pruning Datensatz betrachtet werden müssen, ist ein Bottleneck der Pruning Prozesse und motiviert eine Parallelisierung. Diese wurde mit der *joblib*-Bibliothek realisiert und die Methode für die Berechnung der Matrix Q mittels einer `metric` Methode der zweiten Kategorie wird vorgestellt.

```
def fill_q_parallel(predictions, y, metric, jobs):
  n_est = len(predictions)
  Q = Parallel(n_jobs=jobs, backend="threading")(
      delayed(metric) (predictions[i], predictions[j], y)
      for i in range(n_est) for j in range(n_est) )
  Q = np.reshape(Q, (n_est,n_est))
  for i in range(n_est):
    Q[i][i] = 0
  return Q
```

5.1.3 Testumgebung und Zeit- und Speichermessung

Die Testumgebung für die Untersuchungen dieser Arbeit stellen Methoden bereit, die systematisch die im Framework auswählbaren Pruning Konfigurationen durchlaufen und auf Ensemblemodelle anwenden. Sowohl die zugrundeliegenden Modelle als auch die Datenverarbeitung werden mit Methoden der *scikit-learn*-Bibliothek realisiert. Für die Untersuchungen können folgende Aspekte konfiguriert werden:

Test Konfiguration Diese Parameter umfassen zum einen den verwendeten Datensatz, der mittels *k-fold cross* Validierung aufgeteilt wird in Trainings- und Validierungsdatensätze. Neben dem `k` für die Anzahl an *k-fold cross* Iterationen wird auch die Variable `split_prune_test` $\in (0, 1)$ belegt. Diese legt fest, wie groß der Anteil des Validierungsdatensatzes ist, der für den Pruning Datensatz abgesondert wird.

Ensemble Konfiguration Mit dem Parameter `forestsize` wird die Größe der zu generierenden Ensemblemodelle der *scikit-learn*-Bibliothek festgelegt. Es werden pro *k-cross fold* Iteration sowohl ein *random forest* als auch ein *extremely randomized trees* Ensemble generiert. Zu beachten ist, dass für folgende Pruning Prozesse innerhalb einer *k-fold cross* Iteration stets neue *deep copies* dieser Ensembles erstellt werden und keine neuen

Ensembles generiert werden. Da der Generierungsprozess zufällige Elemente wie das *bootstrapping* Verfahren bei *random forests* enthält, wird mit *deep copies* die Vergleichbarkeit der Pruning Verfahren erhalten.

Pruning Konfiguration In den Arrays `ps`, `metrics` und `optimizers` werden die finalen Größen der Subensembles, die Pruning Metriken und die Approximationsverfahren für die Untersuchung festgelegt. Mit einem *grid search* über alle Konfigurationen werden die resultierenden Pruning Verfahren auf *deep copies* der ursprünglichen Ensembles angewendet. Für den Pruning Prozess werden das Ensemble und die jeweilige Pruning Parameter der `prune_sklearn` Methode des Ensemble Pruning Frameworks übergeben.

Für alle Ensemblemodelle werden die *accuracy* und *AUC* auf dem Validierungsdatensatz der jeweiligen *k-fold cross* Iteration gemessen. Zusätzlich werden weitere Daten erhoben, um die *pruned* Ensembles untereinander und mit dem ursprünglichen Ensemble zu vergleichen. Die Ausführungszeit der `predict`-Methode auf dem Validierungsdatensatz und die Gesamtzahl an *nodes* aller Entscheidungsbäume eines Ensembles werden als Richtwerte für den Ressourcenbedarf betrachtet. Da der genaue Speicherbedarf und die genaue Ausführungszeit abhängig von der zugrundeliegenden Implementation der Modelle sind, werden allgemeinere Werte zur theoretischen Abschätzung des Ressourcenbedarfs erhoben.

5.2 Ziele und Konfiguration der Untersuchungen

Mit dem Framework dieser Arbeit und der beschriebenen Testumgebung werden zwei Untersuchungen durchgeführt. Diese umfassen alle Pruning Verfahren, die sich durch Kombinationen der implementierten Pruning Metriken und Approximationsverfahren ergeben. Insgesamt sind dies 14 unterschiedliche Verfahren. Da für Pruning Verfahren, die Pruning Metriken der ersten Kategorie verwenden, sowohl der *Greedy* Algorithmus als auch die Methoden des *gurobi*-Solvers das optimale Subensemble finden, wird für diese Verfahren nur ein Repräsentant betrachtet. Dies betrifft die Pruning Metriken von Guo et al. [9] und Lu et al. [19] und auch *accuracy* und *AUC* als Pruning Metriken.

Untersuchung 1 Ziel der ersten Untersuchung ist es die Effektivität der betrachteten Pruning Verfahren zu analysieren. Dafür werden diese auf *random forest* und *extremely randomized trees* Ensembles mit 128 Entscheidungsbäumen angewendet. Diese Größe wurde wegen den in Kapitel 2.2 erwähnten Ergebnissen einer umfassenden Untersuchung von *random forests* Ensembles gewählt. Weiterhin wird für die Generierung der *scikit-learn* Ensembles der *Gini-Index* als *split*-Kriterium genutzt, die Tiefe der Bäume ist unbeschränkt und die maximale Anzahl an betrachteten *features* pro *split* beträgt der Wurzel

Datensatz	Beispiele	Features	Klassen
covertype	581012	54	7
letter	20000	17	26
magic	19020	10	2
satlog	6435	36	6
segment	2320	19	7
sensorless	58509	48	11
spambase	4601	57	2
wine-quality	6492	11	7

Tabelle 5.1: Verwendete Datensätze der UCI Machine Learning Repository [8]

der Gesamtzahl an *features*. Alle Pruning Verfahren reduzieren die Modellgröße auf 32 Entscheidungsbäume, weshalb in dieser Untersuchung die Gesamtanzahl an *nodes* und die Ausführungszeit der Ensembles ignoriert wird. Die *accuracy* und *AUC* der *pruned* Ensembles wird innerhalb der *k-fold cross* Iterationen auf Validierungsdatensätzen gemessen und anschließend über alle Iterationen gemittelt. Schließlich wird ein Ranking der Pruning Verfahren über alle betrachteten Datensätze erstellt.

Untersuchung 2 Ziel der zweiten Untersuchung ist es die Sensibilität der Pruning Verfahren bezüglich der finalen Ensemblegröße zu analysieren. Die Generierung der *random forest* und *extremely randomized trees* Ensembles erfolgt dabei wie in Untersuchung 1. Drei ausgewählte Pruning Verfahren, die die Metriken von Cavalcanti et al. [4], Lu et al. [19] und Zhang et al. [35] mit den Methoden des *gurobi*-Solvers kombinieren, reduzieren die Ensemblegröße von 128 auf 32, 16 und 8 Entscheidungsbäume. Zusätzlich zur *accuracy* der *pruned* Ensembles werden die Ausführungszeit der *predict*-Methode auf dem Validierungsdatensatz und die Gesamtzahl der *nodes* gemessen. Schließlich werden die Größen und die *accuracies* der Ensembles ins Verhältnis gesetzt.

Für die Untersuchungen werden insgesamt 8 Datensätze der „UCI Machine Learning Repository“ [8] verwendet, um eine Auswahl an Klassifikationsproblemen abzudecken. Eine Übersicht über die Datensätze bezüglich Anzahl an Beispielen, Features und Klassen gibt Tabelle 5.1. Für die Anzahl an Iterationen der *k-fold cross* Validierung werden 5 Iterationen gewählt, da die Hälfte des Validierungsdatensatzes für den Pruning Datensatz reserviert wird. Als Testsystem dient ein Rechner mit einem „Intel® Core™ i7-6700“ Prozessor und 32GB an Arbeitsspeicher.

5.3 Ergebnisse der Untersuchungen

Untersuchung 1 Ein Auszug der Ergebnisse für Untersuchung 1 mit *random forest* Ensembles wird in Tabelle 5.2 dargestellt. Die vollständigen Ergebnisse für *random forest* und für *extremely randomized trees* Ensembles sind dem Anhang der Arbeit angefügt. Für das finale Ranking der Ensemble Pruning Verfahren wurde der Durchschnitt über die Rankings aller Datensätze berechnet. Bei gleicher *accuracy* der Ensembles wurden die *AUC* Werte betrachtet.

Für die Untersuchung mit *random forest* Ensembles war das Pruning Verfahren mit der Metrik von Zhang et al. [35] und dem *gurobi*-Solver am effektivsten. Im direkten Vergleich mit der ähnlichen Pruning Metrik aus *accuracy* und *Kappa-Statistic*, welche ebenfalls für einzelne Modelle ihre *accuracy* und für Modellpaare ihre Diversität berechnet, generierte die Metrik von Zhang et al. effektivere Subensembles in 11 von 16 Fällen.

		covertype		spambase		wine-quality		8 Datensätze
		acc	AUC	acc	AUC	acc	AUC	Ranking
Original		0.93956	0.99627	0.95002	0.98613	0.69387	0.85054	
zufällig		0.93299	0.99375	0.94741	0.98407	0.67785	0.81553	
Cavalcanti	M	0.93505	0.99471	0.94959	0.98407	0.67909	0.81616	7
Cavalcanti	G	0.93471	0.99473	0.95219	0.98409	0.68433	0.81491	6
Guo	M/G	0.93126	0.99393	0.94828	0.98434	0.68278	0.82630	13
Lazarevic	M	0.93289	0.99437	0.95133	0.98432	0.68155	0.82102	3
Lazarevic	G	0.93360	0.99474	0.95046	0.98416	0.68093	0.82128	12
Lu	M/G	**0.93898**	0.99540	0.95133	0.98544	0.68217	0.82881	2
Margineantu	M	0.93195	0.99413	0.95045	0.98464	0.67754	0.81839	9/10
Margineantu	G	0.93176	0.99416	0.94959	0.98420	0.67939	0.81869	14
Xu/Zhang	M	0.93831	0.99539	0.94959	0.98397	0.68248	0.81928	**1**
Xu/Zhang	G	0.93858	0.99537	0.95350	0.98452	**0.68740**	0.82968	4
accuracy	M/G	0.93896	0.99538	**0.95567**	0.98606	0.68463	0.82682	5
acc+Kappa	M	0.93192	0.99414	0.95089	0.98477	0.67785	0.81741	11
acc+Kappa	G	0.93380	0.99458	0.94959	0.98383	0.68124	0.82656	8
AUC	M/G	0.93801	0.99537	0.95263	0.98545	0.67754	0.81345	9/10

Tabelle 5.2: Ausschnitt Untersuchung 1: *random forests*, 128 auf 32 Modelle, 5-fache k-fold cross Validation (M=*gurobi*-Solver, G=*greedy*-Verfahren)

Im Vergleich der Verfahren mit Pruning Metriken, die sich nur auf die Diversität der Ensemblemodelle konzentrieren, stellte sich die Metrik von Lazarevic et al. [15] mit dem *gurobi*-Solver auf Rang 3 am effektivsten heraus. Bezüglich den Pruning Verfahren, die Pruning Metriken der ersten Kategorie verwenden, erzielte die Metrik von Lu et al. [19] Rang 2 und generierte auf zwei Datensätzen die Subensembles mit der höchsten *accuracy*.

Mit *Critical Difference Diagrams* [1, 6] wird die Effektivität von Modellen über mehrere Datensätze auf statistische Signifikanz untersucht und dargestellt. Abbildung 5.1 visualisiert die Ergebnisse der ersten Untersuchung mit *random forests*. Für die betrachteten Pruning Verfahren konnte kein statistisch signifikanter Unterschied in den *accuracies* der Subensembles festgestellt werden. Zwar können die Verfahren anhand der *accuracy* Werte gerankt werden, es kann jedoch nicht ausgeschlossen werden, dass die Ergebnisse bei der Konfiguration mit einer Reduzierung auf 32 Entscheidungsbäume dem Zufall entsprechen und die betrachteten Pruning Verfahren die gleiche Effektivität besitzen.

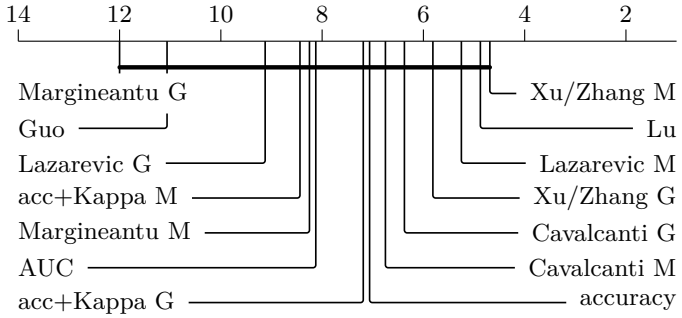

Abbildung 5.1: Critical Difference Diagram zur Untersuchung 1: *random forests* (M=*gurobi*-Solver, G=*greedy*-Verfahren)

In Tabelle 5.3 werden der *gurobi*-Solver und der *greedy*-Algorithmus gegenübergestellt. Betrachtet werden nur die Pruning Metriken, für die sich die Ergebnisse der beiden Approximationsverfahren unterscheiden. Es wird gezählt, wie häufig die Subensembles der jeweiligen Pruning Konfigurationen ihr Gegenüber bezüglich *accuracy* und *AUC* schlagen. Dabei stellte sich im direkten Vergleich der *gurobi*-Solver als geeigneteres Approximationsverfahren heraus.

	Cavalcanti	Lazarevic	Margineantu	Xu/Zhang	acc+Kappa
gurobi	11	13	11	8	7
greedy	5	3	5	8	9

Tabelle 5.3: Vergleich der Solver: *random forests* und *extremely randomized trees*, Anzahl Siege

Untersuchung 2 Die Ergebnisse der zweiten Untersuchung für *random forest* Ensembles sind in Tabelle 5.4 zusammengefasst. Die Werte der Tabelle entsprechen dem Mittel der drei verwendeten Ensemble Pruning Verfahren. Die Ergebnisse für die *extremely randomized trees* Ensembles sind dem Anhang der Arbeit angefügt.
Die Anzahl an Entscheidungsbäumen eines Ensembles verläuft linear mit der Gesamtzahl an Knoten. Damit bedeutet eine Halbierung der Ensemblegröße eine theoretische Halbierung des Speicherbedarfs.
Aufgrund der effizienten Implementierung der *scikit-learn* Modelle und den verhältnismäßig kleinen Validierungsdatensätzen, findet in den meisten Fällen keine Verbesserung bezüglich der Laufzeit der *predict* Methoden statt. Lediglich im Fall des „covertype“ Datensatzes, welcher mit fast 600.000 Beispielen signifikant größer als die anderen ist, nimmt die Größe des Validierungsdatensatzes und der Entscheidungsbäume Einfluss auf die Laufzeit. Somit kann der voraussichtliche lineare Zusammenhang zwischen Ensemblegröße und Laufzeit durch die Ergebnisse dieser Untersuchung nicht gezeigt werden.

	128 Modelle			32 Modelle			16 Modelle			8 Modelle		
	acc	Nodes	Zeit	acc	Nodes	Zeit	acc	Nodes	Zeit	acc	Nodes	Zeit
covertype	0.93884	$6.8 \cdot 10^6$	0.328s	0.93688	$1.7 \cdot 10^6$	0.112s	0.93181	$833 \cdot 10^3$	0.106s	0.92113	$413 \cdot 10^3$	0.106s
letter	0.96450	$538 \cdot 10^3$	0.104s	0.96067	$134 \cdot 10^3$	0.103s	0.95130	$67 \cdot 10^3$	0.103s	0.93173	$33 \cdot 10^3$	0.102s
magic	0.88097	$351 \cdot 10^3$	0.103s	0.87981	$88 \cdot 10^3$	0.102s	0.87497	$44 \cdot 10^3$	0.102s	0.86712	$22 \cdot 10^3$	0.102s
satlog	0.91933	$90 \cdot 10^3$	0.105s	0.91349	$23 \cdot 10^3$	0.102s	0.91309	$11 \cdot 10^3$	0.103s	0.89699	$6 \cdot 10^3$	0.103s
segment	0.98182	$21 \cdot 10^3$	0.103s	0.97922	$5 \cdot 10^3$	0.102s	0.97836	$3 \cdot 10^3$	0.103s	0.97403	$1 \cdot 10^3$	0.103s
sensorless	0.99853	$186 \cdot 10^3$	0.104s	0.99854	$45 \cdot 10^3$	0.104s	0.99831	$22 \cdot 10^3$	0.104s	0.99751	$11 \cdot 10^3$	0.104s
spambase	0.95002	$75 \cdot 10^3$	0.104s	0.95263	$19 \cdot 10^3$	0.102s	0.94988	$9 \cdot 10^3$	0.102s	0.94408	$5 \cdot 10^3$	0.102s
wine-quality	0.68771	$304 \cdot 10^3$	0.104s	0.68278	$76 \cdot 10^3$	0.103s	0.67364	$38 \cdot 10^3$	0.103s	0.65445	$19 \cdot 10^3$	0.102s

Tabelle 5.4: Untersuchung 2: *random forests*, 5-fache k-fold cross Validation, Durchschnittswerte von 3 Ensemble Pruning Verfahren

Abbildung 5.2 visualisiert die Auswirkungen der drei ausgewählten Pruning Verfahren auf die *accuracies* der verkleinerten *random forest* Ensembles. Die Pruning Verfahren werden hierbei einzeln dargestellt. Die Abbildung für *extremely randomized trees* ist ebenfalls dem Anhang beigefügt.
In den meisten Fällen verschlechtert sich die *accuracy* der Ensembles bei einer Reduzierung auf 32 Entscheidungsbäume nur minimal. Dies trägt voraussichtlich zu den Ergebnissen des *Critical Difference Diagrams* in Untersuchung 1 bei. Erst ab einer Größe von 16 Entscheidungsbäumen und damit einer Reduzierung auf 12.5% der Originalgröße sind deutliche Verschlechterungen bei den Ensembles der Datensätze „letter“ und „wine-quality“ sichtbar. Eine Reduzierung auf nur 8 Entscheidungsbäume bedeutet grundsätzlich eine signifikante Verschlechterung der *accuracies* um mindestens 1%.

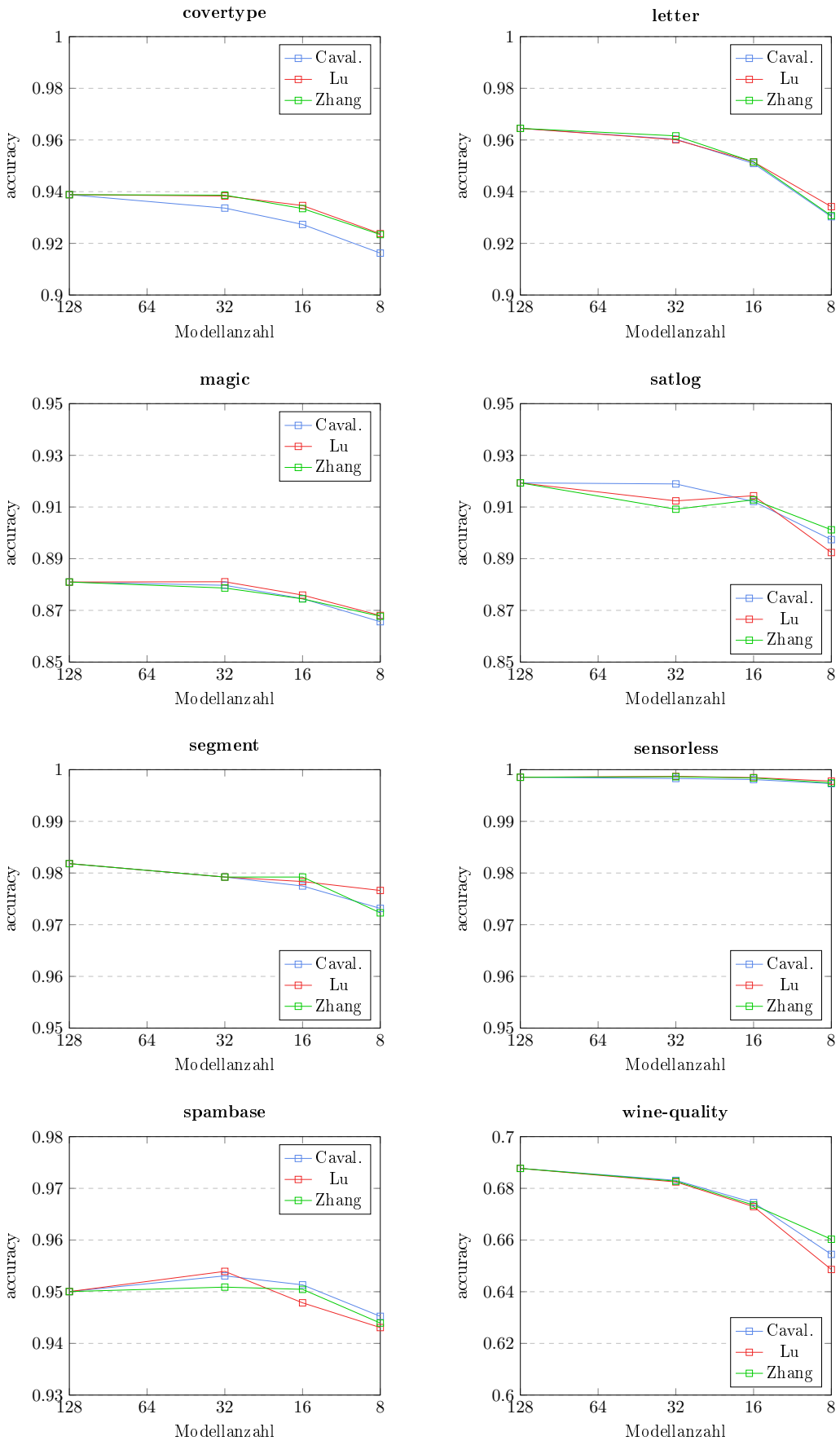

Abbildung 5.2: Untersuchung 2: *random forests*, 5-fache k-fold cross Validation

Kapitel 6

Schlussworte

Die Methoden des maschinellen Lernens sind effektive Werkzeuge, um große und komplexe Datenmengen auszuwerten, und werden zunehmend auf mehr und heterogeneren Geräten eingesetzt. Mit ihnen lassen sich bisher unlösbare Probleme bewältigen und sie sind flexibel auf verschiedene Aufgaben anwendbar. Besonders Ensemble Methoden, die als State of the Art Ansätze gelten, finden weite und erfolgreiche Anwendungen. Ihre Integration in verteilter und ressourcenbeschränkter Hardware motiviert eine Verbesserung der Effizienz dieser Modelle. Ensemble Pruning Verfahren existieren seit über 20 Jahren und versuchen durch eine Reduktion der Ensemblegröße die Effizienz der Modelle zu steigern, ohne die Vorhersagegenauigkeit einzuschränken. Dabei können diese Verfahren als zusätzlicher Schritt nach der Generierung eines Ensemblemodells aufgefasst werden und erfordern in ihrer Verwendung minimalen Mehraufwand.
Aktuelle Entwicklungen wie der große Wachstum an elektrischen Endgeräten und gesammelten Datenmengen sorgen dafür, dass das Interesse an der effizienten Datenverarbeitung steigt und infolgedessen eine Vielzahl an neuen Ensemble Pruning Verfahren vorgestellt werden. Ein einheitliches Framework, das die heterogenen Ansätze auf wesentliche Aspekte reduziert, schafft einen Überblick und ermöglicht das faire Vergleichen dieser. Somit können besonders einflussreiche Merkmale oder häufige Ähnlichkeiten der Verfahren identifiziert werden.

Das Ensemble Pruning Framework dieser Arbeit differenziert zwischen den zwei Merkmalen, der Pruning Metrik beziehungsweise der Lossfunktion und des Approximationsverfahrens, und baut auf der Beobachtung auf, dass die meisten Ensemble Pruning Verfahren eine Pruning Metrik einführen. Statt die Auswahl des Subensembles über eine Lossfunktion zu optimieren, wird die Auswahl mit Hilfe einer Metrik getroffen. Dies motiviert die Definition des Ensemble Pruning Problems mit gewählter Pruning Metrik und es wurde gezeigt, dass es konvex ist.

Mit einer Implementation des Frameworks wurde eine Auswahl an Pruning Verfahren auf ihre Effektivität untersucht. Zwar generierten gewisse Pruning Verfahren auf den betrachteten Datensätzen Ensembles mit höherer *accuracy*, jedoch sind die Unterschiede in den Ergebnissen zu gering, um von einer statistischen Signifikanz zu sprechen. Eine zweite Untersuchung beleuchtete den Zusammenhang zwischen der Vorhersagegenauigkeit und dem allgemeinen Ressourcenbedarf von Ensemblemodellen. Grundsätzlich kann eine Reduzierung der Ensemblegröße eine beachtliche Verschlechterung der *accuracy*-Werte bedeuten. In den betrachteten Problemen wirkte sich jedoch eine Reduzierung von 128 auf 32 Entscheidungsbäume nur geringfügig auf die *accuracies* der Ensembles aus. Dennoch muss letztendlich der Anwender abwägen, welches Verhältnis von Vorhersagegenauigkeit und Ressourceneffizienz dieses Pareto-Optimierungsproblems angemessen für den Verwendungszweck der Ensemblemodelle ist.

Weiterhin unterliegt das Framework dieser Arbeit Limitationen und spezielle Ansätze von Pruning Verfahren können nicht integriert werden. Dies betrifft insbesondere dynamische Ensemble Pruning Verfahren oder Verfahren, die das Auswahlproblem modifizieren wie der multiobjektive Ansatz von Qian et al. in [28]. Eine Erweiterung des Frameworks bezüglich zusätzlichen Pruning Metriken und Approximationsverfahren, die umfassendere Vergleiche ermöglichen würden, ist jedoch möglich.
Auch die in dieser Arbeit vorgenommen Untersuchungen bilden nur eine erste Anwendung des Frameworks und bieten Raum für zukünftige Erweiterungen. Das Testen der *pruned* Ensembles auf ressourcenbeschränkten Endgeräten oder die Messung des genauen Energiebedarfs der Inferenzoperationen können ebenfalls interessante Erweiterungen sein. Eine Verbesserung der Zugänglichkeit und der Verbreitung von Ensemble Pruning Verfahren könnte erreicht werden, indem eine Auswahl an Verfahren in große Software-Bibliotheken des maschinelles Lernens integriert werden.

Insgesamt wurde mit dieser Arbeit versucht ein theoretisches und praktisches Fundament zu legen, welches Ensemble Pruning Verfahren vergleichbar macht, effektive Merkmale von bestehenden Verfahren hervorhebt und die Erforschung neuartiger Ansätze anregt. Mit der steigenden Relevanz von effektiven und lokalen Datenanalysen auf ressourcenbeschränkten Endgeräten setzt sie damit an einem aktuellen Problem der modernen Informationstechnologie an. Zwar wurden in dieser Arbeit nur Ensemble Modelle betrachtet, da diese aber als State of the Art Lösungen im Bereich des maschinellen Lernens gelten, wird ihre Bedeutung in Zukunft weiterhin bestehen.

Anhang A

Tabellen und Abbildungen

		covertype		letter		magic		satlog		segment		sensorless		spambase		wine-quality		
		acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	Ranking
Original		0.93956	0.99627	0.96450	0.99953	0.88170	0.93434	0.93230	0.98872	0.98009	0.99892	0.99846	0.99999	0.95002	0.98613	0.69387	0.85054	
zufällig		0.93299	0.99375	0.95480	0.99865	0.87613	0.92749	0.91752	0.98735	0.97403	0.99864	0.99795	0.99997	0.94741	0.98407	0.67785	0.81553	
Cavalcanti	M	0.93505	0.99471	**0.96150**	0.99865	0.87791	0.93061	**0.93547**	0.98749	0.98009	0.99882	0.99832	0.99997	0.94959	0.98407	0.67909	0.81616	7
Cavalcanti	G	0.93471	0.99473	0.95910	0.99890	0.87802	0.93013	0.93210	0.98658	0.97749	0.99881	0.99839	0.99999	0.95219	0.98409	0.68433	0.81491	6
Guo	M/G	0.93126	0.99393	0.95660	0.99915	0.87865	0.92876	0.91975	0.98583	0.98009	0.99879	0.99757	0.99997	0.94828	0.98434	0.68278	0.82630	13
Lazarevic	M	0.93289	0.99437	0.96000	0.99886	0.87875	0.93038	0.93107	0.98768	0.98182	0.99886	0.99846	0.99997	0.95133	0.98432	0.68155	0.82102	3
Lazarevic	G	0.93360	0.99474	0.95760	0.99911	0.87875	0.93006	0.92392	0.98538	0.97662	0.99873	0.99843	0.99999	0.95046	0.98416	0.68093	0.82128	12
Lu	M/G	**0.93898**	0.99540	0.95990	0.99903	0.87939	0.92943	0.92352	0.98720	**0.98182**	0.99896	0.99843	0.99997	0.95133	0.98544	0.68217	0.82881	2
Margineantu	M	0.93195	0.99413	0.95790	0.99893	0.88012	0.93122	0.92927	0.98748	0.97922	0.99873	0.99829	0.99997	0.95045	0.98464	0.67754	0.81839	9/10
Margineantu	G	0.93176	0.99416	0.95640	0.99887	0.87854	0.92956	0.92432	0.98672	0.97662	0.99877	0.99822	0.99997	0.94959	0.98420	0.67939	0.81869	14
Xu/Zhang	M	0.93831	0.99539	0.96140	0.99889	0.87907	0.93046	0.93170	0.98789	0.98009	0.99886	**0.99863**	0.99998	0.94959	0.98397	0.68248	0.81928	**1**
Xu/Zhang	G	0.93858	0.99537	0.95950	0.99895	0.87865	0.92886	0.92707	0.98718	0.97835	0.99889	0.99829	0.99999	0.95350	0.98452	**0.68740**	0.82968	4
accuracy	M/G	0.93896	0.99538	0.95710	0.99912	**0.88044**	0.92965	0.92750	0.98780	0.97662	0.99878	0.99805	0.99997	**0.95567**	0.98606	0.68463	0.82682	5
acc+Kappa	M	0.93192	0.99414	0.95710	0.99894	0.87970	0.93102	0.92847	0.98750	0.97922	0.99870	0.99829	0.99997	0.95089	0.98477	0.67785	0.81741	11
acc+Kappa	G	0.93380	0.99458	0.95870	0.99879	0.87907	0.92935	0.93027	0.98595	0.98181	0.99876	0.99808	0.99999	0.94959	0.98383	0.68124	0.82656	8
AUC	M/G	0.93801	0.99537	0.95830	0.99908	0.87907	0.92996	0.92530	0.98726	0.97922	0.99880	0.99805	0.99997	0.95263	0.98545	0.67754	0.81345	9/10

Tabelle A.1: Untersuchung 1: *random forests*, 128 auf 32 Entscheidungsbäume, 5-fache k-fold cross Validation (M=*gurobi*-Solver, G=*greedy*-Verfahren)

		covertype		letter		magic		satlog		segment		sensorless		spambase		wine-quality		
		acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	AUC	Ranking
Original		0.93957	0.99599	0.97270	0.99968	0.87971	0.93471	0.93968	0.99079	0.97749	0.99904	0.99949	0.99999	0.95481	0.98808	0.69079	0.87000	
zufällig		0.93232	0.99351	0.96110	0.99897	0.87319	0.92798	0.92070	0.98864	0.97922	0.99858	0.99945	0.99997	0.94872	0.98647	0.67139	0.83476	
Cavalcanti	M	0.93317	0.99378	0.96780	0.99925	0.87676	0.92967	0.93228	0.99109	**0.98355**	0.99865	0.99952	0.99997	0.95437	0.98679	0.68063	0.84445	**1**
Cavalcanti	G	0.93239	0.99396	0.96670	0.99918	0.87518	0.92879	0.93088	0.98887	0.97922	0.99852	0.99949	0.99996	0.95176	0.98606	0.68432	0.84433	12
Guo	M/G	0.93058	0.99320	0.96350	0.99912	0.87645	0.92803	0.93410	0.98993	0.97229	0.99836	0.99945	0.99996	0.95263	0.98650	0.67693	0.83500	14
Lazarevic	M	0.93163	0.99342	0.96730	0.99932	0.87729	0.92914	0.93128	0.99043	0.98268	0.99863	0.99956	0.99997	0.95220	0.98682	0.68771	0.84985	5
Lazarevic	G	0.93164	0.99358	0.96630	0.99916	0.87603	0.92885	0.92770	0.98990	0.97922	0.99846	0.99945	0.99997	0.95524	0.98675	0.68340	0.85150	11
Lu	M/G	0.93974	0.99482	**0.96800**	0.99916	0.87697	0.93020	0.92928	0.98916	0.98355	0.99827	0.99956	0.99998	0.95306	0.98714	0.68125	0.83247	4
Margineantu	M	0.93062	0.99310	0.96570	0.99925	0.87739	0.92915	0.93828	0.99057	0.97749	0.99856	0.99938	0.99997	0.95393	0.98663	0.68247	0.85183	10
Margineantu	G	0.93132	0.99339	0.96410	0.99919	0.87655	0.92900	**0.93965**	0.98938	0.97576	0.99898	0.99938	0.99997	0.95394	0.98659	**0.68955**	0.84431	9
Xu/Zhang	M	0.93843	0.99456	0.96780	0.99909	0.87666	0.92965	0.92828	0.99004	0.98355	0.99864	**0.99959**	0.99996	0.95350	0.98698	0.68094	0.85148	6
Xu/Zhang	G	0.93905	0.99479	0.96710	0.99918	0.87697	0.93024	0.93048	0.98794	0.98355	0.99829	0.99956	0.99999	0.95394	0.98744	0.68217	0.84431	2
accuracy	M/G	**0.93989**	0.99508	0.96720	0.99909	0.87624	0.92939	0.92588	0.98752	0.98268	0.99869	0.99952	0.99998	0.95350	0.98786	0.68771	0.83558	7
acc+Kappa	M	0.93071	0.99313	0.96570	0.99925	**0.87781**	0.92936	0.93808	0.99064	0.97835	0.99856	0.99938	0.99997	0.95437	0.98671	0.68371	0.85136	8
acc+Kappa	G	0.93220	0.99386	0.96580	0.99913	0.87582	0.92953	0.93048	0.98943	0.97835	0.99845	0.99942	0.99996	**0.95567**	0.98595	0.68217	0.84499	13
AUC	M/G	0.93914	0.99480	0.96670	0.99903	0.87603	0.92992	0.93368	0.98872	0.98095	0.99870	0.99949	0.99998	0.95480	0.98766	0.68864	0.85182	3

Tabelle A.2: Untersuchung 1: *extremely randomized trees*, 128 auf 32 Entscheidungsbäume, 5-fache k-fold cross Validation (M=*gurobi*-Solver, G=*greedy*-Verfahren)

	128 Modelle			32 Modelle			16 Modelle			8 Modelle		
	acc	Nodes	Zeit	acc	Nodes	Zeit	acc	Nodes	Zeit	acc	Nodes	Zeit
covertype	0.93891	$19.4 \cdot 10^6$	0.508s	0.93651	$4.8 \cdot 10^6$	0.153s	0.93165	$2.4 \cdot 10^6$	0.116s	0.92076	$1.2 \cdot 10^6$	0.111s
letter	0.97360	$1.1 \cdot 10^6$	0.185s	0.96637	$261 \cdot 10^3$	0.105s	0.95883	$130 \cdot 10^3$	0.105s	0.94003	$65 \cdot 10^3$	0.105s
magic	0.87697	$1.1 \cdot 10^6$	0.104s	0.87466	$280 \cdot 10^3$	0.104s	0.87066	$140 \cdot 10^3$	0.104s	0.85987	$70 \cdot 10^3$	0.105s
satlog	0.92248	$215 \cdot 10^3$	0.104s	0.91086	$54 \cdot 10^3$	0.105s	0.90510	$27 \cdot 10^3$	0.104s	0.90633	$13 \cdot 10^3$	0.105s
segment	0.98182	$64 \cdot 10^3$	0.104s	0.98009	$16 \cdot 10^3$	0.105s	0.97807	$8 \cdot 10^3$	0.105s	0.97489	$4 \cdot 10^3$	0.105s
sensorless	0.99956	$432 \cdot 10^3$	0.125s	0.99951	$103 \cdot 10^3$	0.106s	0.99936	$51 \cdot 10^3$	0.106s	0.99912	$25 \cdot 10^3$	0.106s
spambase	0.95437	$201 \cdot 10^3$	0.105s	0.95350	$51 \cdot 10^3$	0.105s	0.94944	$25 \cdot 10^3$	0.105s	0.94336	$13 \cdot 10^3$	0.105s
wine-quality	0.68925	$694 \cdot 10^3$	0.104s	0.67990	$173 \cdot 10^3$	0.105s	0.67179	$87 \cdot 10^3$	0.104s	0.65517	$43 \cdot 10^3$	0.104s

Tabelle A.3: Untersuchung 2: *extremely randomized trees*, 5-fache k-fold cross Validation, Durchschnittswerte von 3 Ensemble Pruning Verfahren

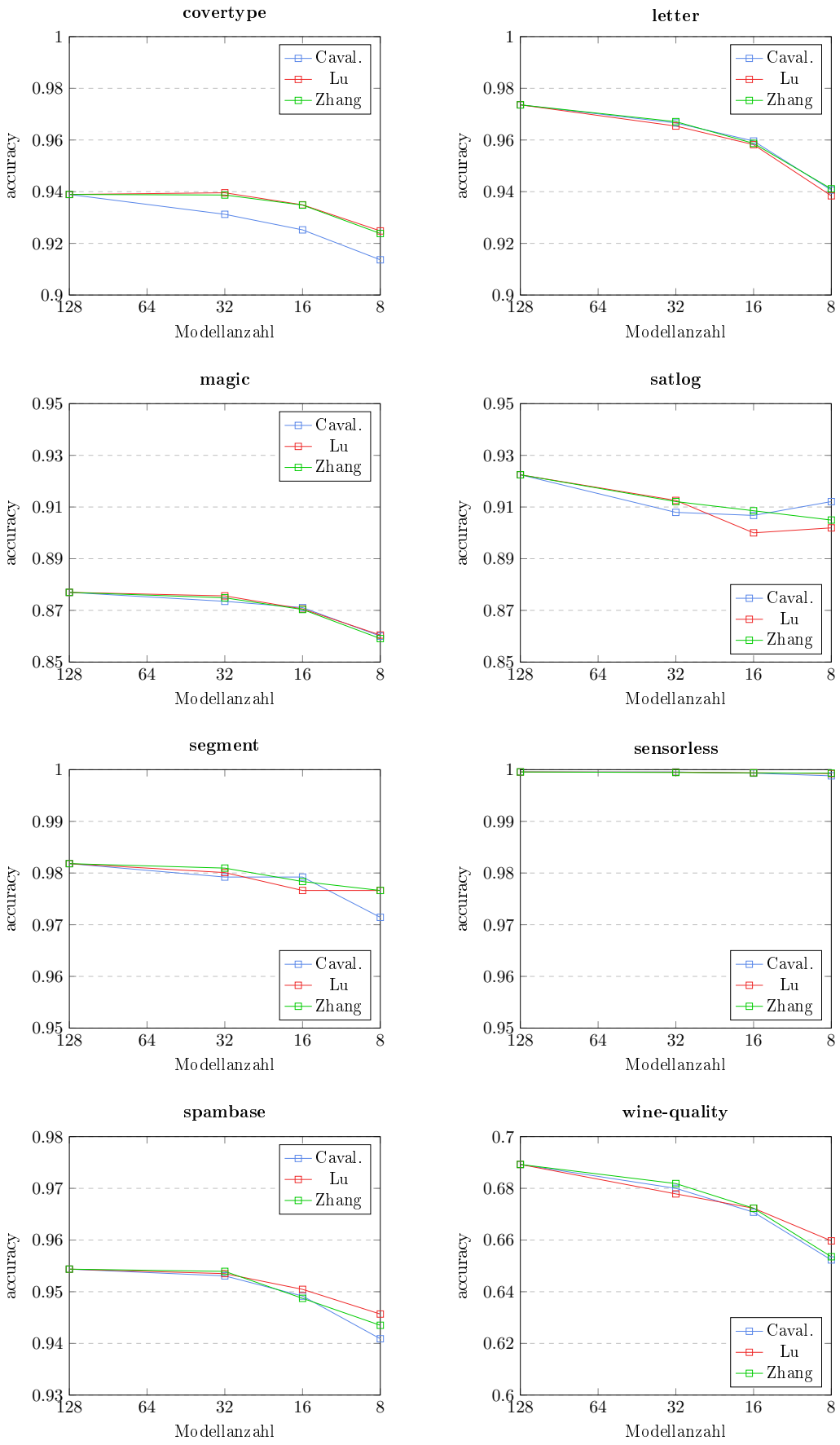

Abbildung A.1: Untersuchung 2: *extremely randomized trees*, 5-fache k-fold cross Validation

Literaturverzeichnis

[1] Benavoli, Alessio, Giorgio Corani und Francesca Mangili: *Should we really use post-hoc tests based on mean-ranks?* The Journal of Machine Learning Research, 17(1):152–161, 2016.

[2] Bennett, Kristin P und Emilio Parrado-Hernández: *The interplay of optimization and machine learning research.* The Journal of Machine Learning Research, 7:1265–1281, 2006.

[3] Boyd, Stephen, Stephen P Boyd und Lieven Vandenberghe: *Convex optimization.* Cambridge university press, 2004.

[4] Cavalcanti, George DC, Luiz S Oliveira, Thiago JM Moura und Guilherme V Carvalho: *Combining diversity measures for ensemble pruning.* Pattern Recognition Letters, 74:38–45, 2016.

[5] Dai, Qun: *A novel ensemble pruning algorithm based on randomized greedy selective strategy and ballot.* Neurocomputing, 122:258–265, 2013.

[6] Demšar, Janez: *Statistical comparisons of classifiers over multiple data sets.* The Journal of Machine learning research, 7(1):1–30, 2006.

[7] Dietterich, Thomas G et al.: *Ensemble learning.* The handbook of brain theory and neural networks, 2:110–125, 2002.

[8] Dua, Dheeru und Casey Graff: *UCI Machine Learning Repository*, 2017.

[9] Guo, Huaping, Hongbing Liu, Ran Li, Changan Wu, Yibo Guo und Mingliang Xu: *Margin & diversity based ordering ensemble pruning.* Neurocomputing, 275:237–246, 2018.

[10] Gurobi Optimization, LLC: *Gurobi Optimizer Reference Manual*, 2021.

[11] Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt et al.: *Array programming with NumPy.* Nature, 585(7825):357–362, September 2020.

[12] Horn, Roger A und Charles R Johnson: *Matrix analysis*. Cambridge university press, 2012.

[13] Huang, Jin und Charles X Ling: *Using AUC and accuracy in evaluating learning algorithms*. IEEE Transactions on knowledge and Data Engineering, 17(3):299–310, 2005.

[14] Joblib Development Team: *Joblib: running Python functions as pipeline jobs*, 2021.

[15] Lazarevic, Aleksandar und Zoran Obradovic: *Effective pruning of neural network classifier ensembles*. In: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, Band 2, Seiten 796–801. IEEE, 2001.

[16] LeCun, Yann, Yoshua Bengio und Geoffrey Hinton: *Deep learning*. nature, 521(7553):436–444, 2015.

[17] Li, Nan, Yang Yu und Zhi-Hua Zhou: *Diversity regularized ensemble pruning*. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Seiten 330–345. Springer, 2012.

[18] Liu, Zhuan, Qun Dai und Ningzhong Liu: *Ensemble selection by GRASP*. Applied intelligence, 41(1):128–144, 2014.

[19] Lu, Zhenyu, Xindong Wu, Xingquan Zhu und Josh Bongard: *Ensemble pruning via individual contribution ordering*. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Seiten 871–880, 2010.

[20] Margineantu, Dragos D und Thomas G Dietterich: *Pruning adaptive boosting*. In: *ICML*, Band 97, Seiten 211–218. Citeseer, 1997.

[21] Martínez-Muñoz, Gonzalo, Daniel Hernández-Lobato und Alberto Suárez: *An analysis of ensemble pruning techniques based on ordered aggregation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2):245–259, 2008.

[22] Nguyen, Tien Thanh, Anh Vu Luong, Manh Truong Dang, Alan Wee-Chung Liew und John McCall: *Ensemble selection based on classifier prediction confidence*. Pattern Recognition, 100:107104, 2020.

[23] Nocedal, Jorge und Stephen Wright: *Numerical optimization*. Springer Science & Business Media, 2006.

[24] Oshiro, Thais Mayumi, Pedro Santoro Perez und José Augusto Baranauskas: *How many trees in a random forest?* In: *International workshop on machine learning and data mining in pattern recognition*, Seiten 154–168. Springer, 2012.

[25] Partalas, Ioannis, Grigorios Tsoumakas und Ioannis Vlahavas: *An ensemble uncertainty aware measure for directed hill climbing ensemble pruning.* Machine Learning, 81(3):257–282, 2010.

[26] Partalas, Ioannis, Grigorios Tsoumakas und Ioannis Vlahavas: *A study on greedy algorithms for ensemble pruning.* Aristotle University of Thessaloniki, Thessaloniki, Greece, 2012.

[27] Pedregosa, F., 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 und E. Duchesnay: *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12:2825–2830, 2011.

[28] Qian, Chao, Yang Yu und Zhi-Hua Zhou: *Pareto ensemble pruning.* In: *Proceedings of the AAAI Conference on Artificial Intelligence*, Band 29, 2015.

[29] Rokach, Lior: *Decision forest: Twenty years of research.* Information Fusion, 27:111–125, 2016.

[30] Sagi, Omer und Lior Rokach: *Ensemble learning: A survey.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4):e1249, 2018.

[31] Shalev-Shwartz, Shai und Shai Ben-David: *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[32] Sze, Vivienne, Yu-Hsin Chen, Joel Emer, Amr Suleiman und Zhengdong Zhang: *Hardware for machine learning: Challenges and opportunities.* In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*, Seiten 1–8. IEEE, 2017.

[33] Tsoumakas, Grigorios, Ioannis Partalas und Ioannis Vlahavas: *A taxonomy and short review of ensemble selection.* In: *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, Seiten 41–46, 2008.

[34] Xu, Linli, Bo Li und Enhong Chen: *Ensemble pruning via constrained eigenoptimization.* In: *2012 IEEE 12th International Conference on Data Mining*, Seiten 715–724. IEEE, 2012.

[35] Zhang, Yi, Samuel Burer und W Nick Street: *Ensemble pruning via semi-definite programming.* Journal of machine learning research, 7(Jul):1315–1338, 2006.

[36] Zhou, Zhi-Hua: *Ensemble methods: foundations and algorithms.* CRC press, 2012.