Inhaltsverzeichnis

1 Einleitung

1.1 Ziel der Arbeit

Random Forests (Breiman, 2001) gehören zu den mächtigsten Machine Learning Algorithmen (Gron, 2017). In verschiedenen Domänen wurde ihre praktische Anwendbarkeit unter Beweis gestellt. Einige Beispiele sind Anwendungen in der Chemoinformatik (Svetnik u. a., 2003), Ökologie (Prasad u. a. 2006; Cutler u. a. 2007), 3D-Objekterkennung (Shotton u. a., 2011) und Bioinformatik (Diaz-Uriarte u. Alvarez, 2006) sowie ein Data Science Hackathon zur Luftqualitätsvorhersage (http://www.kaggle.com/c/dsg-hackathon) (Tang u. a., 2018). Howard u. Bowles (2012) bezeichnen Random Forests als den erfolgreichsten Allzweck-Algorithmus der neueren Zeit.

Derzeit sind Random Forests noch nicht ausreichend für die Finanzdomäne erforscht. Insbesondere die Wahl von Hyperparametern – die häufig willkürlich oder über Heuristiken gesetzt werden – stellt eine vielversprechende Forschungsfrage da. Eine Gegenüberstellung der Anwendbarkeit von von Random Forests für verschiedene Fragen, wie der Stock Price Prediction oder dem Credit Scoring, hat noch nicht stattgefunden.

Diese Arbeit hat das Ziel, die Anwendung von Decision Trees und Random Forests in der Finanzdomäne kritisch zu evaluieren und, wo sinnvoll, mittels Hyperparametern zu optimieren.

Ziele (DT=Decision Tree, RF=Random Forest):

- Forschungsfrage: Können DT und RF Klassifikatoren sinnvoll (besser als Dummy Classifier oder besser als Markt Index Performance) in der Finanzdomäne eingesetzt werden?
- Unterfrage 1: Können DT und RF sinnvoll für Credit Scoring (Anwendung 1) und Aktienempfehlungen (Anwendung 2) eingesetzt werden?
- Unterfrage 2: Wie sollten die Hyperparameter jeweils für DT und RF pro Anwendungsfall gesetzt werden?
- Unterfrage 2.5: Kann Pruning in Anwendung 1 und Anwendung 2 zur Vermeidung von Overfitting des DT beitragen?
- Unterfrage 3: Wie unterscheiden sich DT und RF Algorithmen bzw. wann eignet sich welcher Klassifikator?
- Unterfrage 3.5: Ist Meta Random Forest sinnvoll? Soft Voting vs Hard Voting?
- Unterfrage 4: Lassen sich relevante Effekte aus der realen Welt (Christmas Effekt, Dividendenausschüttung, Jahresabschluss-Veröffentlichung, etc.) in den Modellen wiederfinden?

- Unterfrage 5: Sind die Ergebnisse mit der Markteffizienzhypothese vereinbar?
- Unterfrage 6: Welche Einschränkungen gibt es bei der Anwendung von DT und RF Klassifikatoren in der Finanzdomäne?

Einschränkungen: Diese Arbeit untersucht nur binäre Klassifikation (keine Regression) und vernachlässigt sämtliche Transaktionskosten (da diese variabel sind etc.)

2 Theoretische Grundlagen

2.1 Machine Learning

Machine Learning bedeutet, einen Computer so zu programmieren, dass ein bestimmtes Leistungskriterium anhand von Beispieldaten oder Erfahrungswerten aus der Vergangenheit optimiert wird (Alpaydın, 2008). Diese Programme führen für ein gegebenen Input nicht immer zum selben Ergebnis, sondern lernen – ähnlich wie der Mensch – anhand von Beispielen. Deshalb eignet sich Machine Learning für Probleme, für die der Mensch keine simplen Regeln verfassen und in einem Algorithmus automatisieren kann. Stattdessen löst der Mensch solche Probleme anhand von Erfahrungen, teilweise anhand von Intuition. Ein Beispiel ist die Diagnose von Krebszellen. Fachärzte durchlaufen eine jahrelange Ausbildung und analysieren eine Vielzahl von Beispielbildern, um eine Einschätzung über neue Zellen treffen zu können (Quelle: TBD). Eine solche Diagnose kann nicht mit einem simplen Algorithmus automatisiert werden, da jedes Zellenbild einzigartig ist. Es ist nicht realistisch, in einem Algorithmus alle Eventualitäten programmatisch abzudecken. Machine Learning hingegen ermöglicht es, den menschlichen Lernprozess nachzubilden und somit komplexe Diagnosen zu erstellen. Vorteile gegenüber dem Menschen sind dabei die Objektivität, da der Computer keinen Emotionen unterliegt, und die Fähigkeit, große Mengen an Daten in einem Bruchteil der Zeit, die ein Mensch benötigen würde, zu vergleichen.

Technisch gesehen lernt ein Machine Learning Programm, indem es eine Funktion

$$f(X|\Theta) = \hat{y}, \quad f \in F \tag{2.1}$$

aus dem Funktionenraum F bildet und durch Training mit Beispieldaten die Parameter Θ optimiert. Der Funktionenraum F umfasst alle Funktionen, die ein Modell erlernen kann, und ist vom gewählten Lernalgorithmus abhängig. Die Beispieldaten sind eine Stichprobe χ mit n Datensätzen in der Form

$$\chi = \{(X_1, y_1), ..., (X_n, y_n)\}. \tag{2.2}$$

Dabei steht X für einen m-dimensionalen Input-Vektor der Form

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \tag{2.3}$$

der als Informationsgrundlage zur Bestimmung von $\hat{y}$ dient. Die Variable $\hat{y}$ bezeichnet das Ergebnis der Funktion für einen gegebenen Input-Vektor X. Die Optimierung von $f(X|\Theta)$ bezieht sich hier auf die Suche jener Parameter Θ, welche die genaueste Näherung für die tatsächliche Funktion $f^*(X)$ und somit auch für die tatsächliche Klasse y erzielen (Alpaydın, 2008). Ist die Variable $\hat{y}$ numerisch, so spricht man von Regression. Ein Beispiel

für eine Regression ist die Vorhersage der Temperatur in Grad Celsius. Dabei enthält die Ergebnismenge der Funktion unendlich viele Elemente, zum Beispiel die reellen Zahlen. Ist die Variable $\hat{y}$ kategorisch, so spricht man von Klassifikation. Im Spezialfall einer binären Klassifikation gilt zudem

$$\hat{y} = \begin{cases} 1, & \text{wenn } X \text{ eine positive Instanz ist} \\ 0, & \text{wenn } X \text{ eine negative Instanz ist.} \end{cases} \tag{2.4}$$

Welche Instanzen positiv und welche negativ sind, ist vom Anwender zu definieren. Ein Anwendungsbeispiel für eine binäre Klassifikation ist die Bestimmung, ob eine gegebene Pflanze für den Menschen giftig ist oder nicht. Bei der Klassifikation ist die Menge der möglichen Werte von $\hat{y}$ endlich. In binären Fall kann $\hat{y}$, wie in Gleichung 2.4 gezeigt, genau zwei Werte annehmen.

Nach dem Training trifft das Modell Aussagen über Instanzen, die außerhalb von χ liegen. Die Generalisierungsfähigkeit des Modells mit Parametern Θ wird anhand der Abweichungen zwischen $\hat{y}$ und y für alle i Instanzen eines Validierungs-Datensets χ_{val} mittels einer Fehlerfunktion

$$E(\Theta|\chi_{val}) = \sum_i Diff(\hat{y}, y) \tag{2.5}$$

berechnet. Für die Diff-Funktion gibt es zahlreiche Methoden, deren Eignung vom konkreten Anwendungsfall abhängt (Alpaydın, 2008). Die in dieser Arbeit verwendeten Methoden sind in Kapitel **??**, "Methodik", erläutert. Das finale Modell verwendet anschließend jene Funktion, die die geringsten Abweichungen auf χ_{val} erreicht, also die beste Generalisierung aufweist. Um die erwartete Fehlerrate des finalen Modells zu bestimmen, werden die Abweichungen auf einem Test-Datenset χ_{test} – wie in Abschnitt **??** – herangezogen. Die Aufgabe von Machine Learning ist es also, jene Parameter Θ_{opt} zu finden, welche die Fehlerfunktion minimieren, also

$$\Theta_{opt} = \underset{\Theta}{\operatorname{argmin}}\, E(\Theta|\chi). \tag{2.6}$$

An dieser Stelle sei auf einen grundlegenden Trade-Off im Machine Learning hingewiesen: die Komplexität der Funktion, bedingt durch die Mächtigkeit des Funktionenraumes F, die Menge an Daten in χ und der Generalisierungsfehler sind voneinander abhängig (Alpaydın, 2008). Steigende Komplexität führt bei gleichbleibender Datenbasis zu einem höheren Generalisierungsfehler, kann aber bei größerer Datenbasis zu einem niedrigeren Generalisierungsfehler führen. Wenn die Datenmenge ceteris paribus steigt, nimmt der Generalisierungsfehler ab, und vice versa (Alpaydın, 2008).

Der Schwerpunkt dieser Arbeit liegt auf der Klassifikation. Deshalb wird diese in der folgenden Sektion genauer betrachtet.

2.2 Klassifikation

Klassifikation bezeichnet das Zuweisen einer Klasse $\hat{y}$ zu einem gegebenen Input-Vektor X. Ein Beispiel ist die Bestimmung der Kreditwürdigkeit eines Bankkunden, bekannt als das

Credit Scoring Problem (Abdou u. Pointon, 2011). Die Bank hat bei der Vergabe eines Kredites das Ziel, das Ausfallrisiko zu minimieren und den erwarteten Gewinn zu maximieren. Dazu werden von Kredit-Antragsstellern Datenpunkte wie Vermögen, Gehalt, Kredithistorie und Alter erhoben. Die gesammelten Daten dienen anschließend als Trainingsdaten für einen Machine Learning Algorithmus. Dieser erstellt ein Modell, das für die historischen Daten optimiert ist. Dieses Modell klassifiziert dann die Kreditwürdigkeit der Antragssteller. Dadurch wird die Bank bei der Kreditentscheidung unterstützt. In manchen Fällen wird die Entscheidung anhand von Schwellenwerten komplett automatisiert (Abdou u. Pointon, 2011).

Das vorangegangene Beispiel beinhaltet typische Elemente einer Klassifikation: Instanzen, Features und Klassen. Eine Instanz ist im Beispiel ein Kunde. Jede Instanz wird durch dieselben Attribute – aber womöglich mit unterschiedlichen Ausprägungen – beschrieben. Diese Attribute werden als Features, Φ, bezeichnet. Die Ausprägungen der Features für eine gegebene Instanz dienen als Input-Vektor X für die Klassifizierung, wie in Gleichung 2.1 dargestellt. Aufgrund der grundlegenden Bedeutung der Features stellen deren Berechnung (Feature Extraction) und Auswahl (Feature Selection) zentrale Schritte im Machine Learning dar, den das Kapitel **??** vertieft. Die möglichen kategorischen Werte der Ergebnis-Variable y, genannt Klassen, könnten im Beispielfall "Kunde mit hohem Risiko" und "Kunde mit geringem Risikoßein. Das Ergebnis einer Klassifikation ist die Klasse der betrachteten Instanz, im Beispiel die Kreditwürdigkeit eines Kunden.

Ein simpler Klassifikator könnte unter anderem anhand von folgender Regel handeln:

$$IF(\text{Salary} > 100.000 \wedge \text{Credit Amount} < 200.000), THEN\, Class = \text{"Low Risk"}. \tag{2.7}$$

Basierend auf dem Gehalt des Kunden und der Kreditsumme, schätzt das Modell das Risiko der Kreditvergabe ein. In der Realität reichen diese zwei Kriterien allerdings häufig nicht aus, um eine fundierte Entscheidung zu treffen. Bankangestellte prüfen ebenfalls die Kredithistorie, die Lebenssituation, den Arbeitsvertrag, den persönlichen Eindruck sowie weitere – möglicherweise auch die Persönlichkeit betreffende – Faktoren (Quelle: TBD). Sofern diese Faktoren als Features in den Daten vorhanden sind, eignet sich der Klassifikator diese in der Trainingsphase an und ergänzt sie in das Entscheidungsmodell. Bis zu einem gewissen Punkt erhöht das Hinzufügen von Features und Regeln den Erfolg des Modells, wie in Abschnitt **??** dargestellt, bevor die zunehmende Komplexität die Generalisierungsfähigkeit vermindert.

Diese Bachelorarbeit beschränkt sich auf binäre Klassifikatoren, also Klassifikatoren mit genau zwei Klassen. Konkret werden Decision Tree und Random Forest Klassifikatoren sowie einige Abwandlungen dergleichen untersucht. Es folgen nun die theoretischen Grundlagen der Erstellung, Anwendung und Erfolgsmessung dieser Modelle.

2.3 Decision Tree Klassifikator

2.3.1 Generelle Funktionsweise und Eigenschaften

Die Induktion von Decision Trees (Quinlan, 1986) gehört zu den simpelsten aber erfolgreichsten Machine Learning Algorithmen (Russell u. Norvig, 2009). Gupta u. a. (2017) nennen unter

anderem die medizinische Diagnostik, intelligente Fahrzeuge, das Credit Scoring (siehe Abschnitt 2.2) und die industrielle Qualitätskontrolle als Anwendungsbereiche. Außerdem bilden Decision Trees die Grundlage für das später behandelte Random Forest Ensemble. Ein Decision Tree funktioniert nach dem Teile-und-Herrsche Prinzip. Die Trainings-Stichprobe $\chi_{training}$ wird nach einem festgelegten Kriterium in Teilmengen geteilt. Die entstandenen Teilmengen wiederum werden nach der gleichen Prozedur aufgeteilt. Diese Rekursion findet so lange statt, bis ein definiertes Endkriterium erfüllt ist und der Decision Tree zur Klassifikation bereit ist. Nachfolgend wird zuerst der Aufbau eines Decision Trees erklärt, sowie anschließend dessen Erstellung und Optimierung.

Anknüpfend an das das Beispiel der Kreditvergabe zeigt Abbildung 2.1 einen möglichen Decision Tree für diesen Anwendungsfall.

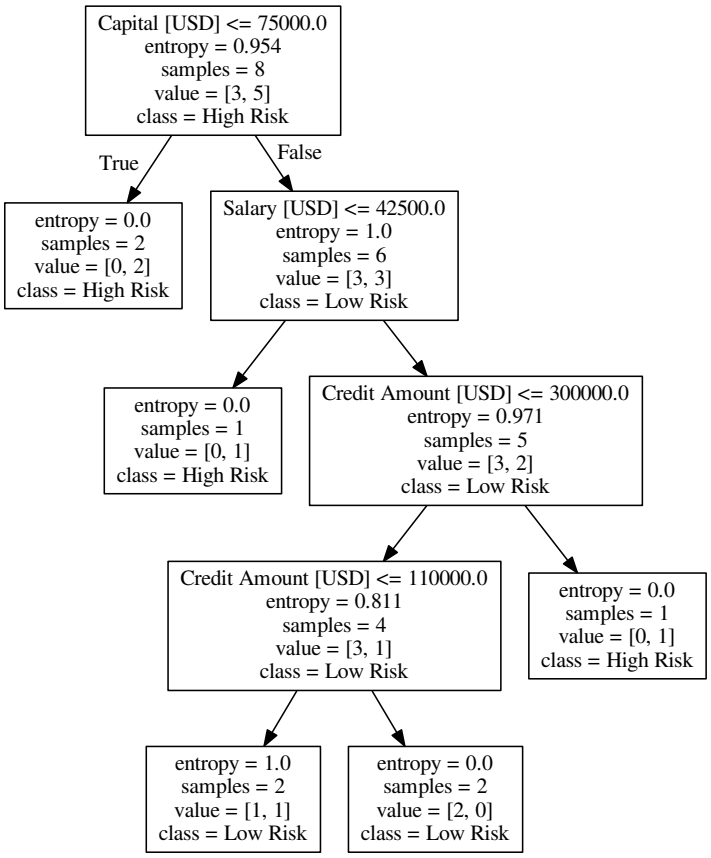

Abbildung 2.1 Beispielhafter Decision Tree für den Anwendungsfall der Kreditvergabe

Ein Decision Tree besteht aus der Wurzel, internen Knoten, Ästen und externen Knoten, genannt Blätter. Die Wurzel ist der einzige interne Knoten, der keinen Vorgänger hat. Ein interner Knoten ist ein Knoten, der Nachfolger hat. Interne Knoten testen jeweils ein bestimmtes Feature Φ der betrachteten Instanz X auf dessen Wert. Äste sind die Verbindungen zwischen Knoten. Jeder Ast bildet eine Wertemenge ab, die im vorgelagerten internen Knoten festgelegt wurde. Ein Blatt ist ein Knoten ohne Nachfolger. Blätter repräsentieren, im Fall der

Klassifikation, die Klassen (Quinlan, 1986). In dieser Arbeit werden binäre Decision Trees betrachtet, die genau zwei Klassen kennen. Auch finden ausschließlich univariate Bäume Anwendung, das heißt interne Knoten testen jeweils nur auf ein Feature, und nicht auf mehrere.

Die Klassifikation einer Instanz beginnt in der Wurzel des Decision Trees. Dort wird die Instanz auf ein Feature getestet und, je nach Wert dieses Features, entweder an den linken oder an den rechten Nachfolger der Wurzel weitergeleitet. Diese Prüfung mit Weiterleitung findet solange statt, bis die Instanz an einem Blatt angekommen ist. Dort wird der Instanz die Klasse des Blattes zugewiesen, wodurch diese Instanz klassifiziert ist (Russell u. Norvig, 2009). Der Graph aller Knoten, die die Instanz durchwandert hat, nennt sich Pfad.

Eine vorteilhafte Eigenschaft des Decision Trees ist die Best Case Laufzeit-Komplexität für die Klassifizierung von $\mathcal{O}(\log_2(n))$, die erreicht wird, wenn die Baumstruktur balanciert ist. Die Variable n steht für die Anzahl der Instanzen, mit denen der Decision Tree erstellt wurde, also $|\chi_{training}|$. Im Worst Case liegt die Komplexität bei $\mathcal{O}(n)$. Das ist der Fall, wenn für jedes ϕ_{opt} an jedem internen Knoten die Instanzen $X \in \chi_{training}$ so aufgeteilt werden, dass eine einzige Instanz als Blatt deklariert wird und die restlichen $n-1$ Instanzen in einen weiteren internen Knoten einfließen. Dort wiederholt sich der Vorgang rekursiv solange, bis alle Instanzen einem Blatt zugewiesen wurden und der Decision Tree fertig erstellt ist.

Ein weiterer Vorteil von Decision Trees ist, dass sie eine Menge von geordneten, simplen Regeln darstellen und somit für den Menschen leicht verständlich sind, wie die Regel aus dem Abschnitt 2.2 zeigt. So lässt sich zum Beispiel die Regel

$$\begin{aligned} &IF(Capital > 75.000 \wedge Salary \leqslant 42.500 \wedge CreditAmount > 300.000), \\ &\quad THEN\, Class = "HighRisk" \end{aligned} \tag{2.8}$$

aus der Abbildung 2.1 herleiten. Diese Verständlichkeit macht den Decision Tree zu einem sogenannten White Box Modell, dessen Entscheidungen für den Menschen nachvollziehbar sind. Black Box Modelle, wie der Random Forest oder neuronale Netze, hingegen treffen schwer- oder nicht-rekonstruierbare Entscheidungen (Gron, 2017). Wegen dieser Vorteile ist der Decision Tree Klassifikator sehr beliebt und wird in manchen Fällen den exakteren aber komplexeren Methoden vorgezogen (Alpaydın, 2008).

Im Gegensatz zu einigen statistischen Modellen wie der linearen oder exponentiellen Regression, gibt es im Funktionenraum F von Decision Trees stets eine Funktion $f(X|\Theta)$, die die Klassen aller Trainings-Instanzen korrekt abbildet. Ein Decision Tree kann prinzipiell beliebig viele Regeln definieren und somit jede Instanz aus $\chi_{training}$ korrekt abbilden. Diese Eigenschaft bringt allerdings einen Nachteil mit sich. Decision Trees sind für Overfitting (siehe Abschnitt **??**) anfällig. Denn falls der Decision Tree, ohne Kontrolle der Baumstruktur, für jede Instanz einen eigenen Blattknoten anlegt, erzielt er zwar auf $\chi_{training}$ ein optimales Ergebnis, ist jedoch nicht zur Generalisierung und somit nicht zum Lernen fähig. Entsprechende Gegenmaßnahmen, um die Baumstruktur zu kontrollieren, finden sich im Unterabschnitt 2.3.2.

2.3.2 Erstellung eines Decision Trees

Die Induktion anhand des ID3-Algorithmus führt zu einem von mehreren äquivalenten Decision Trees. Es gibt mehr als einen Decision Tree, der die Regeln einer gegebenen Stichprobe $\chi_{training}$ kodiert (Quinlan, 1986). Nach Ockhams Rasiermesser ist es erstrebenswert, den Baum mit der geringsten Komplexität den anderen vorzuziehen (siehe Unterabschnitt 2.3.4). Diesen zu suchen ist jedoch ein NP-vollständiges Problem (Quinlan, 1986). Stattdessen bietet sich eine lokale Suche über Heuristiken an. Die nachfolgenden Lernalgorithmen sind vom Greedy-Typ: Von der Wurzel aus wählt der Algorithmus in jedem Schritt die in der aktuellen Position beste Aufteilung der Instanzen, um den Decision Tree iterativ zu erstellen (Alpaydın, 2008). Es folgt eine Beschreibung dieser Aufteilung sowie, anschließend, der Güte einer gegebenen Aufteilung.

Beginnend mit allen Trainingsinstanzen $\chi_{training}$ an der Wurzel, entsteht der Decision Tree durch rekursive Aufteilungen derselben an seinen internen Knoten. Eine Aufteilung bezeichnet hier das Bilden von Teilmengen der betrachteten Trainingsinstanzen an einem internen Knoten K_0, um pro Teilmenge einen Ast sowie einen weiteren Knoten K_1 zu generieren. Eine Aufteilung erfolgt stets anhand des optimalen Features ϕ_{opt}. Es sind zwei Fälle zu unterscheiden: entweder ist das Feature diskret oder es ist numerisch. Ist das Feature diskret, wird für jede Ausprägung dieses Features ein Ast generiert. Ist das Feature numerisch, wird es diskretisiert. Dazu werden ein oder mehrere Schwellenwerte festgelegt, die entsprechende Teilmengen definieren. Weist ein Feature ϕ_i zum Beispiel die Werte $[v_{start}, v_{end}]$ auf, könnte der Algorithmus die Schwellenwerte $\{s_1, s_2\}$ so wählen, dass gilt

$$v_{start} < s_1 < s_2 < v_{end}. \tag{2.9}$$

Die entstehenden Teilmengen wären dann

$$\begin{aligned} V_1 &= \{v | v < s_1\}, \\ V_2 &= \{v | v \geq s_1 \wedge v < s_2\} \text{ und} \\ V_3 &= \{v | v \geq s_2\} \end{aligned} \tag{2.10}$$

und würden zu drei entsprechenden Ästen führen. Der Spezialfall von genau einem Schwellenwert, wodurch sich zwei Teilmengen ergeben, wird als binäre Aufteilung bezeichnet (Alpaydın, 2008). Binäre Aufteilungen führen graphisch gesehen zu Hyperrechtecken, wie in Abbildung (TBD: DT Decision Boundaries Plot) dargestellt.

Die Anzahl der möglichen Aufteilungen ist gleich der Anzahl der vorhandenen Features, m. Die möglichen Aufteilungen unterscheiden sich hinsichtlich ihres Einflusses auf den finalen Decision Tree. Deshalb wird die Güte der möglichen Aufteilungen verglichen, um die sinnvollste Aufteilung zu identifizieren.

Als Kriterium für die Güte der Aufteilung verwendet der ID3-Algorithmus die aus ihr resultierende Änderung in Entropie, bezeichnet als Information Gain (IG) (Quinlan, 1986). Die Entropie H liegt per Definition zwischen Null und Eins und trifft eine Aussage über die Homogenität einer gegebenen Menge an n Instanzen. In der Informationstheorie bezeichnet die Entropie "die minimale Anzahl an Bits, die nötig sind, um die Klassifikationsgenauigkeit einer Instanz zu codieren"(Alpaydın, 2008). An jedem Knoten K wird die Entropie berechnet

mit (Shannon, 1948)

$$H(K) = -\sum_{w} p_i \log_2(p_i), \tag{2.11}$$

wobei w die möglichen Ausprägungen von Φ_{opt} bezeichnet. Die Variable p_i bezeichnet die relative Häufigkeit der Klasse y_i innerhalb der Instanzen, deren Ausprägung von Φ_{opt} gleich w ist. Besitzen alle Instanzen innerhalb der Ausprägungen w dieselbe Klasse, so ist die Entropie Null (da $\log_2(1) = 0$) und der Knoten bekommt keine Nachfolger sondern ist ein Blatt. Besitzt jedoch mindestens eine Instanz eine andere Klasse, so ist die Entropie größer als Null. Dann sucht ID3 nach jener Aufteilung, welche den größten IG mit sich bringt. Angenommen die Aufteilung erfolgt anhand von Feature ϕ_j. Dann ergibt sich der IG zwischen einem Knoten K_d der Tiefe d und seinen potenziellen Nachfolgern $K_d|\phi_j$ der Tiefe $d+1$ aus

$$IG(K_d, K_d|\phi_j) = H(K_d) - H(K_d|\phi_j). \tag{2.12}$$

Weiterhin angenommen, dass ϕ_j k Ausprägungen vorweist, dann folgt die Entropie der potenziellen Nachfolger der Tiefe $d+1$ aus

$$H(K_d|\phi_j) = \sum_{k} P(\phi_j = v_k) H(K_d|\phi_j = v_k), \tag{2.13}$$

wobei $P(\phi_j = v_k)$ die Wahrscheinlichkeit bezeichnet, dass ϕ_j den Wert v_k annimmt, und $H(K_d|\phi_j = v_k)$ die erwartete Entropie an jenem Knoten in Tiefe $d+1$ bezeichnet, an welchen eben diese Instanzen mit der Ausprägung v_k gelangen. Diese erwartete Entropie ergibt sich wiederum aus der Gleichung 2.11.

Da ID3 den IG maximiert, minimiert er die erwartete Anzahl an Schritten, die von der Wurzel bis zum Blatt nötig sind, also die Tiefe des Baumes. Eine minimale Tiefe wiederum bedeutet eine geringere Komplexität und ist nach Ockhams Rasiermesser den Alternativen vorzuziehen. Außerdem beschleunigt eine geringe Tiefe die Klassifikation, da die Instanz weniger Knoten und somit weniger Tests durchlaufen muss. Da der beschriebene Greedy-Algorithmus nur lokal sucht, ist das Ergebnis allerdings nicht zwangsläufig das globale Optimum.

Die Instanzen an allen internen Knoten in jeder Tiefe d werden nach dem beschriebenen Procedere aufgeteilt und den entstandenen Nachfolger-Knoten in Tiefe $d+1$ zugewiesen. Dort ruft sich der Algorithmus rekursiv auf. Sobald alle Instanzen an einem Knoten dieselbe Klasse aufweisen, wird dieser Knoten ein Blatt und die Instanzen werden nicht weiter aufgeteilt. Dem Blatt wird die Mehrheitsklasse seiner Instanzen zugewiesen. Pseudocode zur Erstellung eines Decision Trees ist in Algorithmus **??** zu finden. Die Laufzeitkomplexität zur Erstellung des Modells liegt bei $\mathcal{O}(m \times n \log_2(n))$, da an jedem Knoten alle m Features verglichen werden Gron (2017).

Russell u. Norvig (2009) weisen darauf hin, dass Features mit einem starken Verzweigungsfaktor nach dem ID3-Verfahren bevorzugt werden. Ein Beispiel dafür ist ein Zeitstempel, der für jede Instanz aus $\chi_{training}$ einzigartig ist. Dann würde eine Aufteilung anhand dieses Features stets zu einer erwarteten Entropie von Null führen, und würde somit stets den größtmöglichen IG erzielen. Der ID3-Algorithmus würde dieses Feature bevorzugen. Das würde zu starkem Overfitting führen, da das Modell nicht zur Generalisierung befähigt würde, also nicht lernt. Solchen stark-verzweigenden Features kann mit einer Bestrafung

basierend auf dem Verzweigungsfaktor, zum Beispiel mithilfe des Gain Ratios, entgegengewirkt werden (Russell u. Norvig, 2009). Ein Nachfolger des ID3-Algorithmus namens C4.5 berücksichtigt den Gain Ratio bei der Erstellung des Decision Trees (Gupta u. a., 2017).

2.3.3 Grundlegende Parameter

Ein Machine Learning Algorithmus optimiert die Parameter Θ, um die Fehlerrate des Klassifikators zu minimieren (siehe Abschnitt 2.1). Die optimalen Werte, Θ_{opt}, resultieren also aus dem Trainingsprozess und sind erst im Nachhinein bekannt. Neben diesen Parametern gibt es Hyperparameter, die vom Anwender zu Beginn zu setzen sind. Einige dieser Hyperparameter beeinflussen die finalen Parameter Θ_{opt}, zum Beispiel indem sie den Wertebereich einschränken. Die Tabelle 2.1 gibt einen Überblick über grundlegende Parameter (Typ in der Tabelle ist "P") sowie Hyperparameter (Typ in der Tabelle ist "H") von Decision Trees und deren Bedeutung.

Tabelle 2.1 (Hyper-)Parameter eines Decision Trees und deren Bedeutung

Name	Typ	Beschreibung	Zweck (Beispiele)
Maximale Anzahl an betrachteten Features	H	Beschränkung der Features, die pro Knoten verglichen werden	Beschleunigung des Trainings
Maximale Tiefe	H	Begrenzung der Tiefe des Decision Trees	Reduktion von Overfitting
Aufteilungskriterium	H	Kriterium zur Auswahl von Φ_{opt} für eine Aufteilung an Knoten K, z.B. IG oder Gini Index	Messung der Reinheit von Instanzen
Minimale Anzahl an Instanzen für Aufteilung	H	Schwellenwert bezüglich der Anzahl an Instanzen für eine weitere Aufteilung der Instanzen	Reduktion von Overfitting
Minimale Anzahl an Instanzen für Blatt	H	Schwellenwert bezüglich der Anzahl an Instanzen für die Erstellung eines neuen Blattes	Reduktion von Overfitting
Maximale Anzahl an Blättern	H	Begrenzung der Menge an Blättern im Decision Tree	Reduktion von Overfitting
Minimaler Unreinheits-reduktion für Aufteilung	H	Schwellenwert bezüglich des IG für eine weitere Aufteilung der Instanzen	Reduktion von Overfitting
Gewichtung der Instanzen in $\chi_{training}$	H	Gewichtung der Beispieldaten für das Training	Hervorhebung repräsentativer Instanzen bzw. Unterdrückung von Ausreißern
Tiefe	P	Tiefe des Decision Trees, Anzahl der Stufen von Wurzel bis zum entferntesten Blatt	Beurteilung der finalen Struktur
Anzahl an Blättern	P	Anzahl der Blätter im finalen Decision Tree	Beurteilung der finalen Struktur
Anzahl an internen Knoten	P	Anzahl an Knoten mit Nachfolgern (inklusive Wurzel, exklusive Blätter)	Beurteilung der finalen Struktur
Signifikanzen der Features	P	Bedeutung jedes Features Φ_i bei der Klassifizierung, z.B. berechnet anhand des IG durch Φ_i	Identifizierung der einfluss-reichsten Features
Anzahl der Features	P	Anzahl der Features, auf die die internen Knoten des Decision Trees testen	Beurteilung der finalen Struktur
Anzahl der Klassen	P	Anzahl der Klassen, die die Blätter des Decision Tree abbilden	Beurteilung der finalen Struktur

2.3.4 Optimierung von Decision Trees

Nach Erstellung ist ein Decision Tree oft zu komplex und neigt zu Overfitting (Gron, 2017). Eine Methode zur Reduzierung von Overfitting ist das Pruning, also das Kürzen des Baumes. Es wird zwischen Prepruning und Postpruning unterschieden. Prepruning findet noch während der Erstellung des Decision Trees statt. Ein Ansatz für Prepruning ist es, eine Bedingung für Aufteilungen an jedem Knoten zu definieren. Dadurch soll verhindert werden, dass Entscheidungen, die auf zu wenigen Instanzen basieren, die Varianz des Klassifikators erhöhen und somit die Generalisierung verschlechtern (Alpaydın, 2008). So könnte man

festlegen, dass eine weitere Aufteilung nur dann stattfindet, wenn mindestens fünf Prozent der Trainingsinstanzen aus $\chi_{training}$ zu diesem Knoten gelangt sind. Andernfalls wird dieser Knoten zu einem Blatt, betitelt mit der Mehrheitsklasse.

Postpruning hingegen verändert den Baum, nachdem er komplett erstellt wurde. Die Decision Tree Pruning-Methode versucht jene Teilbäume zu identifizieren, welche für das Overfitting verantwortlich sind. Dazu wird vor dem Erstellen des Decision Trees eine Pruning-Menge $\chi_{pruning} \subsetneq \chi$ festgelegt, welche nicht in das Training einfließt. Das Training findet stattdessen mit den Instanzen aus $\chi \setminus \chi_{pruning}$ statt. Anschließend erfolgen pro Teilbaum zwei Messungen von Fehlerraten auf $\chi_{pruning}$. In der ersten Version klassifiziert der Decision Tree in seiner finalen Form die Instanzen aus $\chi_{pruning}$, ohne Veränderung des betrachteten Teilbaums. In der zweiten Version wird der jeweils betrachtete Teilbaum mit seiner Mehrheitsklasse ersetzt, also als Blatt simuliert. Ist der Fehler in der zweiten Version nicht signifikant schlechter als jener in der ersten Version, so wird der betrachtete Teilbaum dauerhaft in ein Blatt umgewandelt. Damit verringert sich die Komplexität und somit das Overfitting des Decision Trees (Russell u. Norvig, 2009). In der Praxis hat sich Postpruning als zielführender als Prepruning herausgestellt (Alpaydın, 2008).

Literaturverzeichnis

[Abdou u. Pointon 2011] Abdou, Hussein ; Pointon, John: Credit Scoring, Statistical Techniques and Evaluation Criteria: A Review of the Literature. In: *Int. Syst. in Accounting, Finance and Management* 18 (2011), 04, S. 59–88. http://dx.doi.org/10.1002/isaf.325. – DOI 10.1002/isaf.325

[Alpaydın 2008] Alpaydin, E.: *Maschinelles Lernen*. Oldenbourg, 2008. – ISBN 9783486581140

[Breiman 2001] Breiman, Leo: Random Forests. In: *Machine Learning* 45 (2001), Oktober, Nr. 1, S. 5–32. http://dx.doi.org/10.1023/A:1010933404324. – DOI 10.1023/A:1010933404324. – ISSN 0885–6125

[Cutler u. a. 2007] Cutler, D. R. ; Edwards, Thomas C. ; Beard, Karen H. ; Cutler, Adele ; Hess, Kyle ; Gibson, Jacob ; Lawler, Joshua J.: Random forests for classification in ecology. In: *Ecology* 88 11 (2007), S. 2783–92

[Diaz-Uriarte u. Alvarez 2006] Diaz-Uriarte, Ramon ; Alvarez, Sara: Gene Selection and Classification of Microarray Data Using Random Forest. In: *BMC bioinformatics* 7 (2006), 02, S. 3. http://dx.doi.org/10.1186/1471-2105-7-3. – DOI 10.1186/1471–2105–7–3

[Gron 2017] Gron, Aurlien: *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st. O'Reilly Media, Inc., 2017. – ISBN 1491962291, 9781491962299

[Gupta u. a. 2017] Gupta, Bhumika ; Rawat, Aditya ; Jain, Akshay ; Arora, Arpit ; Dhami, Naresh: Analysis of Various Decision Tree Algorithms for Classification in Data Mining. In: *International Journal of Computer Applications* 163 (2017), 04, S. 15–19. http://dx.doi.org/10.5120/ijca2017913660. – DOI 10.5120/ijca2017913660

[Howard u. Bowles 2012] Howard, Jeremy ; Bowles, Mike: *The Two Most Important Algorithms in Predictive Modeling Today*. https://conferences.oreilly.com/strata/strata2012/public/schedule/detail/22658, 2012. – Besucht: 19. Oktober 2019

[Prasad u. a. 2006] Prasad, Anantha ; Iverson, Louis ; Liaw, Andy: Newer Classification and Regression Tree Techniques: Bagging and Random Forests for Ecological Prediction. In: *Ecosystems* 9 (2006), 03, S. 181–199. http://dx.doi.org/10.1007/s10021-005-0054-1. – DOI 10.1007/s10021–005–0054–1

[Quinlan 1986] Quinlan, J. R.: Induction of decision trees. In: *Machine Learning* 1 (1986), Mar, Nr. 1, S. 81–106. http://dx.doi.org/10.1007/BF00116251. – DOI 10.1007/BF00116251

[Russell u. Norvig 2009] Russell, Stuart ; Norvig, Peter: *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ, USA : Prentice Hall Press, 2009. – ISBN 0136042597, 9780136042594

[Shannon 1948] Shannon, Claude E.: A Mathematical Theory of Communication. In: *The Bell System Technical Journal* 27 (1948), 7, Nr. 3, S. 379–423. http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x. – DOI 10.1002/j.1538–7305.1948.tb01338.x

[Shotton u. a. 2011] Shotton, J. ; Fitzgibbon, A. ; Cook, M. ; Sharp, T. ; Finocchio, M. ; Moore, R. ; Kipman, A. ; Blake, A.: Real-time Human Pose Recognition in Parts from Single Depth Images. In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2011 (CVPR '11). – ISBN 978–1–4577–0394–2, S. 1297–1304

[Svetnik u. a. 2003] Svetnik, Vladimir ; Liaw, Andy ; Tong, Christopher ; Culberson, J. C. ; Sheridan, Robert P. ; Feuston, Bradley P.: Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. In: *Journal of Chemical Information and Computer Sciences* 43 (2003), Nr. 6, S. 1947–1958. http://dx.doi.org/10.1021/ci034160g. – DOI 10.1021/ci034160g. – PMID: 14632445

[Tang u. a. 2018] Tang, Cheng ; Garreau, Damien ; Luxburg, Ulrike von: When do random forests fail?, 2018