Friedrich-Schiller-Universität Jena
Wirtschaftswissenschaftliche Fakultät
Lehrstuhl für Wirtschaftsinformatik
Prof. Dr. J. Ruhland

# Evaluation von Random Forest und Oblique Forest für einen Vergleich in Bezug auf ihre Performance

---

**Masterarbeit**

**Zur Erlangung des Grades Master of Science (M.Sc.)**

**im Studiengang Wirtschaftsinformatik**

Eingereicht von:
Dario Dubberstein
Karl-Marx-Str. 09
Schmölln, 04626
Martrikelnummer: 180296

Gutachter:
Universitätsprofessor
Dr. Johannes Ruhland
Betreuer:
Sven Gehrke

Jena, 09.08.2021

# Kurzfassung

Mit dem aktuell steigendem Internetverkehr und der damit einhergehenden Speicherung und Verarbeitung von Daten, wird dem Data Mining mehr an Bedeutung zugeordnet. Unternehmen setzen vermehrt auf Prognoseverfahren, um Sicherheitsrisiken zu identifizieren oder kaufverhalten vorherzusagen. Durch dieses steigende Interesse gilt es zunehmen die Forschung an aktuellen und die Analyse vorhandener Klassifikationsverfahren voranzutreiben.

In seiner ursprünglichen Arbeit über Random Forests schlug Breiman zwei verschiedene Entscheidungsbaum-Ensembles vor. Eines aus „orthogonalen“ Bäumen mit Schwellenwerten für einzelne Features in jedem Split, und eines aus „schrägen“ Bäumen, die den Feature-Raum durch zufällig orientierte Hyperebenen trennen. Die vorliegende Arbeit untersucht jeweils ein Verfahren mit diesen Eigenschaften, namentlich Random Forest und Oblique Forest, auf ihre Performance. Unter Verwendung von Standardparameter überzeugte der Oblique Forest auf einem Großteil der getesteten Datensätze, wobei die Rechenzeit in vielen Fällen, über dem des Random Forest lag. Bei der Erstellung der Klassifikationsmodelle konnte zudem eine größere Varianz der Ergebnisse bei dem Random Forest beobachtet werden. Im Vergleich konnte der Random Forest mit einem deutlich geringeren zeitlichen Aufwand implementiert werden. Zudem stellen aktuelle Bibliotheken für den Random Forest viele Optimierungsmöglichkeiten zu Verfügung, wodurch sich die Genauigkeit mit einem hohen Maß an Rechenzeit zum Teil anpassen lässt.

# Inhaltsverzeichnis

# I. Abbildungsverzeichnis

# II. Tabellenverzeichnis

# III. Nomenklatur

**B**

Bagging – Vom engl. Bootstrap aggregating

BI – Business-Intelligence-Software

**C**

C4.5 – Algorithmus, um aus Trainingsdaten einen Entscheidungsbaum zu erzeugen

**D**

DM – Data Mining

DID – Dual Information Distance

**E**

ELT – Transformationsprozess für Datenbanken (extrahieren, laden, transformieren)

ETL – Transformationsprozess für Datenbanken (extrahieren, transformieren, laden)

**F**

Feature – Bezeichnung für ein Merkmal in der Statistik und dem Data-Mining

**I**

IG – Informationsgewinn

**K**

KMU – Kleine und mittlere Unternehmen

KI – Künstlichen Intelligenz

**M**

MAE – Der mittlere absolute Fehler (von engl. Mean Absolute Error)

MSE – Die mittlere quadratische Abweichung (von engl. Mean Square Error)

**N**

NP – Nichtdeterministisch in Polynomialzeit

**O**

OF – Oblique Forest

OLAP – Online-Analyseverarbeitung

OOB – Out of bag

**R**

RF – Random Forest

**S**

Split – Methode zur Partitionierung von verfügbaren Daten

**U**

UCI – Vom engl. University of California, Irvin

# 1 Einleitung

*"Since the popular emergence of data science as a field,*
*its practitioners have asserted that*
*80% of the work involved is*
*acquiring and preparing*
*data"*
*— Harvard Business Review*

## 1.1 Hintergrund

Im aktuellen Wettbewerb suchen Unternehmen ständig nach einem Vorteil, der es Ihnen ermöglicht, Waren und Dienstleistungen zu immer geringeren Kosten, höherer Qualität und schneller als ihre Konkurrenten bereitzustellen. Mit der Verwendung neuster ERP-Systeme lassen sich Daten in allen Bereichen des Unternehmens sammeln und verwerten. Nicht nur die Quantität, sondern auch die Qualität der Daten wird dabei immer wichtiger. In der modernen Geschäftswelt wird dabei jeder Unternehmensbereich von Daten angetrieben und mit ihnen gesteuert.

Data Mining ist der Prozess des Verstehens von Daten durch Bereinigen von Rohdaten, Finden von Mustern, Erstellen von Modellen und Testen dieser Modelle. Es umfasst Statistiken, maschinelles Lernen und Datenbanksysteme. Data Mining umfasst oft mehrere Daten-projekte, sodass es leicht mit Analysen, Data Governance und anderen Datenprozessen verwechselt werden kann.[1]

Data Mining hat eine lange Geschichte. Es entstand mit Computern in den 1960er bis 1980er Jahren. In der Vergangenheit war Data Mining ein intensiver manueller Kodierungsprozess und es erfordert auch heute noch Programmierkenntnisse und sachkundige Spezialisten, um Data Mining-Ergebnisse zu bereinigen, zu verarbeiten und zu interpretieren. Einige der manuellen Prozesse können mit Hilfe von wiederholbaren Abläufen, maschinellem Lernen (ML) und Systemen der künstlichen Intelligenz (KI) automatisiert werden.

Große Konzerne und kleine und mittlere Unternehmen (KMU) in allen Branchen können von Data Mining profitieren. Die richtigen Daten helfen Unternehmen, den Umsatz zu steigern,

[1] Max Bramer 2007.

Kosten zu senken und Kunden zu gewinnen. Neben naheliegenden Einsatzgebieten wie der Optimierung von Marketingkampagnen und um betrügerische Aktivitäten zu erkennen und potenzielle Betrugsfälle zu antizipieren, eignet dieses sich auch um das Verhalten von Mitarbeitern zu verstehen, die Fluktuation vorherzusagen und die Personalpolitik zu bewerten.

Business-Intelligence-Software (BI) kann helfen, indem sie Online-Analyseverarbeitung (OLAP), Location Intelligence, Enterprise-Reporting und mehr kombiniert. BI-Software bietet Unternehmen die Möglichkeit, unterschiedliche Datenquellen zu einer einheitlichen Quelle zu verbinden, die Daten zu sammeln und zu strukturieren. Damit kann den Endbenutzern eine Schnittstelle zum Extrahieren von Berichten und dem Erstellen von sog. Dashboards geboten werden. Mit Hilfe dieser Funktionalitäten kann das Treffen von fundiertere Geschäftsentscheidungen unterstützt werden. Die Praxis zeigt, dass die Einführung von Informationssystemen in Unternehmen oft misslingt und diese Misserfolgsquoten sogar bei über 50% liegen. Ein Softwareeinsatz kann nicht allein zu gesteigerter Prozessleistung, stärkerer Transparenz oder Reduzierung von Kosten führen. Nur ein auf die Geschäftsprozesse abgestimmtes und in die IT-Landschaft eingefügtes System ist in der Lage dazu und wird die gewünschten Effekte mit sich bringen.[2]

## 1.2 Ziel der Arbeit

Data Mining wird verwendet, um aus großen Datensätzen nützliche Informationen zu extrahieren und in leicht zu interpretierenden Visualisierungen darzustellen. Entscheidungsbäume wurden erstmals in den 1960er Jahren eingeführt und sind eine der effektivsten Methoden für das Data Mining. Sie sind in mehreren Disziplinen weit verbreitet,[3] da sie einfach zu verwenden sind, keine Mehrdeutigkeiten aufweisen und auch bei fehlenden Werten robust sind.

Es soll die Verwendung einer allgemeineren und leistungsfähigeren Familie von Splitfunktionen, nämlich des Splits mit Linearkombination untersucht werden, die auch als schräge Splits bekannt sind. Typische Aufteilungskriterien für einen einstufigen Entscheidungsbaum sind diskontinuierlich, da kleine Änderungen der Aufteilungsparameter die Zuordnung von Daten zu Zweigen des Baums ändern können. Infolgedessen sind Splitparameter nicht ohne weiteres einer numerischen Optimierung zugänglich, so dass

[2] Kemmner 1991.
[3] Hastie et al. 2009.

schräge Splitfunktionen bei baumbasierten Methoden nicht weit verbreitet sind. Die vorliegende Arbeit untersucht zwei, auf Entscheidungsbäumen aufbauende Klassifizierungsmethoden auf ihre Performance. Ziel ist es, eine Aussage treffen zu können welches Verfahren sich grundsätzlich besser zur Verwendung eignet.

## 1.3 Vorgehensweise und Aufbau

Der in Abbildung 1.1 skizzierte Ablauf dient als Modell zum thematischen Aufbau der Arbeit, um die vorherigen genannten Ziele zu erfüllen. Dieser orientiert sich stark an der Vorgehensweise während der Bearbeitungszeit.

In Kapitel 2 werden theoretische Grundlagen aus der Fachliteratur beschrieben und Definitionen festgelegt, die als Voraussetzung für folgenden Kapitel herangezogen werden. Zunächst wird das grundlegende Verhalten von Klassifikationsverfahren, im genauen das von Entscheidungsbäumen vorgestellt. Basierend auf diesen ersten groben Überblick werden Terminologien für die folgenden Kapitel kurz erläutert. Es folgt ein Einstieg in den Grund für die Verwendung der für die Arbeit wichtigen Klassifikationsverfahren und deren Funktionsweise, Vorteile und eventuelle Nachteile sowie eine Betrachtung der Unterschiede zueinander. Es wird zudem eine Methode vorgestellt, die für die Gestaltung von Data Mining-Projekten benutzt wird. Wichtige Schritte dieser Methoden kommen in den folgenden Kapitel 3 und Kapitel 4 zum Einsatz.

Auf Basis der Grundlagen werden in Kapitel 3 der Versuchsaufbau im Detail betrachtet. Anhand des Vorgehens aus Kapitel 3 wird in Kapitel 4 die Versuchsdurchführung an einem Beispiel demonstriert. Dabei steht das allgemeine Vorgehen bei der Durchführung im Vordergrund. Es werden aber auch Rahmenbedingungen für den Versuchsaufbau skizziert sowie externe Einflussfaktoren ausgeschlossen, damit Aussagekräftige Ergebnisse sicher-gestellt werden können.

Auf Basis der in Kapitel 2 vorgestellten Klassifikationsverfahren und des in Kapitel 3 dargestellten Vorgehens werden in Kapitel 5 die Ergebnisse der Versuchsdurchführung ausgewertet. Anhand dieser Auswertung werden Rückschlusse auf den Einfluss von Ausprägungen der Datensätze auf die Ergebnisse analysiert. Anschließend wird basierend auf den Rückschlüssen eine Reihe and automatisierten Tests durchgeführt, um die getroffenen Hypothesen zu beweisen.

Im letzten Kapitel werden die gesammelten Ergebnisse und Auswertungen in einer Aussagekräftigen Form gesammelt und präsentiert.

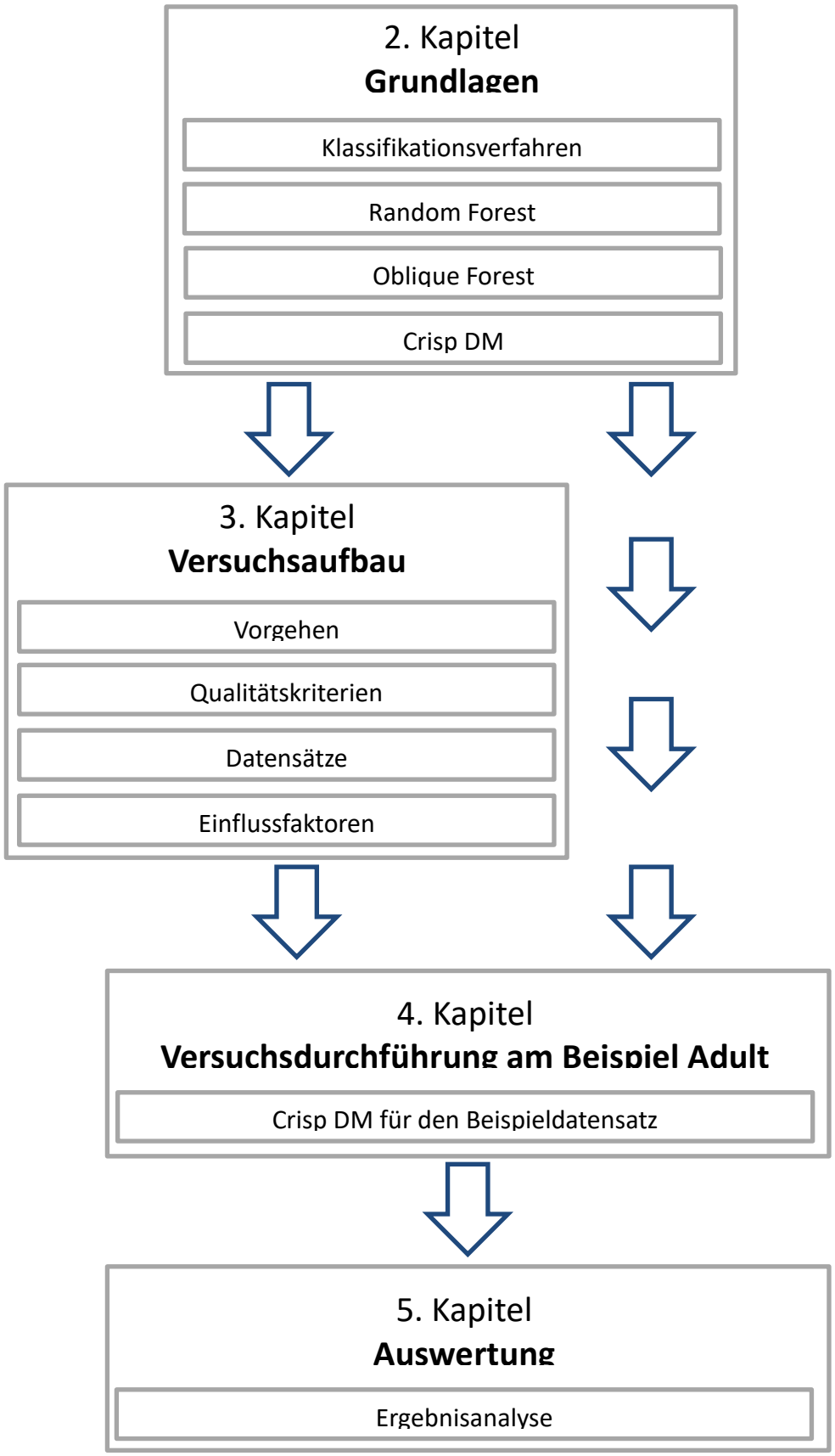



*Abbildung 1.1 Skizzierung des thematischen Aufbaus der Arbeit*

# 2 Grundlagen

## 2.1 Entscheidungsbäume

Beim Data-Mining werden statistische Methoden systematisch auf große Datenbestände angewendet. Entscheidungsbäume sind eine Methode des maschinellen Lernens. Ziel ist es, ein Modell zu erstellen, das den Wert einer Zielvariable basierend auf mehreren Eingabevariablen vorhersagt. Entscheidungsbäume können auch als Kombination mathematischer und rechnerischer Techniken beschrieben werden, um die Beschreibung, Kategorisierung und Verallgemeinerung eines gegebenen Datensatzes zu unterstützen.[4]

Ein Entscheidungsbaum ist eine einfache Darstellung zum Klassifizieren von Beispielen. Ein Entscheidungsbaum oder ein Klassifikationsbaum ist ein Baum, in dem jeder interne (Nicht-Blatt-)Knoten mit einem Eingabemerkmal gekennzeichnet ist. Die Zweige, die von einem Eingabe-Feature beschrifteten Knoten kommen, werden mit jedem der möglichen Werte des Ziel-Features beschriftet. Die restlichen Zweige führen zu einem untergeordneten Entscheidungsknoten auf einem anderen Eingabe-Feature. Jedes Blatt des Baums ist mit einer Klasse oder einer Wahrscheinlichkeitsverteilung über die Klassen gekennzeichnet. Damit wird der Datensatz vom Baum entweder in eine bestimmte Klasse oder in eine bestimmte Wahrscheinlichkeitsverteilung klassifiziert.

Ein Baum wird aufgebaut, indem die Quellmenge, die den Wurzelknoten des Baums bildet, in Teilmengen aufgeteilt wird. Die Aufteilung basiert auf einem Satz von Aufteilungsregeln mit Klassifikationsmerkmalen.[4] Dieser Vorgang wird für jede abgeleitete Teilmenge auf rekursive Weise wiederholt, die als rekursive Partitionierung bezeichnet wird. Die Rekursion ist abgeschlossen, wenn alle Datensätze der Teilmenge an einem Knoten die gleichen Werte der Zielvariablen aufweisen oder wenn die Aufteilung keinen neuen Wert zu den Vorhersagen hinzufügt. Dieser Prozess der Top-Down-Induktion von Entscheidungsbäumen (TDIDT)[5] ist ein Beispiel für einen gierigen Algorithmus und bei weitem eine der am häufigsten verwendeten Strategien zum maschinellen Lernen mit Hilfe von Entscheidungsbäumen.[6]

[4] Shalev-Shwartz and Ben-David 2019.
[5] Quinlan 1986, pp. 81–106.
[6] Ho 1995.

Entscheidungsbäume haben neben anderen Data-Mining-Methoden verschiedene Vor- und Nachteile:

Bäume können ohne großen Aufwand schnell grafisch dargestellt werden, was sie einfach zu verstehen und leicht interpretierbar macht.

Bäume können sowohl numerische als auch kategoriale Daten verarbeiten.[7] Andere Techniken sind normalerweise auf die Analyse von Datensätzen spezialisiert, die nur einen Variablentyp aufweisen. Zum Beispiel können Relationsregeln nur mit nominalen Variablen verwendet werden, während neuronale Netze nur mit numerischen Variablen oder Kategorisierungen verwendet werden können. Frühe Entscheidungsbäume waren nur in der Lage, kategoriale Variablen zu behandeln, neuere Versionen, wie z.B. C4.5 weisen diese Einschränkung nicht mehr auf.[8] Zudem erfordert der Einsatz von Entscheidungsbäumen wenig Datenaufbereitung. Andere Techniken setzen oft eine Datennormalisierung voraus. Bäume können auch mit qualitativen Prädiktoren umgehen, es müssen daher keine künstlich erzeugten Variablen hinzugefügt werden.[9]

Ein weiterer großer Vorteil ist die Verwendung eines sog. White-Box- oder Open-Box-Modells.[8] Bei diesem ist eine gegebene Situation in einem Modell beobachtbar und die Erklärung für die Bedingungen lassen sich leicht durch boolesche Logik erklären. Im Gegensatz dazu ist in einem Black-Box-Modell die Erklärung der Ergebnisse typischerweise schwer nachvollziehbar, beispielsweise bei einem künstlichen neuronalen Netz. Dies führt dazu, dass sich die menschliche Entscheidungsfindung genauer widerspiegelt als bei anderen Ansätzen.[9] Gerade bei Simulationen und Modellierung menschlicher Entscheidungen oder Verhalten ist diese Eigenschaft besonders nützlich. Modelle, die mit Entscheidungsbäumen geniert wurden, lassen sich mit statistischen Tests validieren. Dies ermöglicht es, die Genauigkeit der Approximation des Modells zu berücksichtigen. Irrelevante Funktionen werden weniger verwendet, sodass sie bei nachfolgenden Durchläufen entfernt werden können. Die Hierarchie der Attribute in einem Entscheidungsbaum spiegelt deren Bedeutung wider.[10] Dies bedeutet, dass die Funktionen näher am Wurzelknoten den größten Informationsgehalt haben.[11]

[7] Gareth 2017, p. 103.
[8] Rokach and Maimon 2008.
[9] Gareth 2017, p. 315.
[10] Provost and Fawcett 2013.
[11] Piryonesi and El-Diraby 2020.

Bäume sind, nicht robust bei Veränderungen in den Daten. Eine kleine Änderung der Trainingsdaten kann zu einer großen Änderung des Baums und folglich der endgültigen Vorhersagen führen.[12]

Das Problem des Erlernens eines optimalen Entscheidungsbaums ist unter verschiedenen Aspekten der Optimalität und sogar für einfache Konzepte als NP-vollständig bekannt.[13] [14] Folglich basieren praktische Entscheidungsbaumlernalgorithmen auf Heuristiken wie dem Greedy-Algorithmus, bei dem nur lokal an jedem Knoten optimale Entscheidungen getroffen werden. Solche Algorithmen können nicht garantieren, dass sie den global optimalen Entscheidungsbaum zurückgeben. Um den Greedy-Effekt der ausschließlich lokalen Optimalität zu reduzieren, wurden einige Methoden wie der Dual Information Distance (DID)-Baum vorgeschlagen.[15] Dadurch ist die durchschnittliche Tiefe des Baums, die durch die Anzahl der Knoten oder Tests bis zur Klassifizierung definiert wird, unter verschiedenen Aufteilungskriterien nicht garantiert minimal oder klein.[16]

### 2.1.1 Algorithmus und Terminologie

Das Ziel der Verwendung eines Entscheidungsbaums besteht darin, ein Trainingsmodell zu erstellen, mit dem die Klasse oder der Wert der Zielvariablen vorhergesagt werden kann, indem einfache Entscheidungsregeln gelernt werden, die aus früheren Daten (Trainingsdaten) abgeleitet werden.[12]

Entscheidungsbäume beginnen mit der Vorhersage einer Klassenbezeichnung für einen Datensatz von der Wurzel des Baums. Es werden dafür die Werte des Wurzelattributs mit dem Attribut des Datensatzes verglichen. Anhand der Vergleichsergebnisse wird ein Wert bestimmt der als nächster Knoten fungiert und anhand einer Verzweigung dem Wurzelnoten zugeordnet wird.

Im Folgenden werden spezifische Terminologien kurz erläutert und mit Hilfe eines Beispiels abgebildet (vgl. Abbildung 2.1).

[12] Gareth 2017.
[13] Hyafil and Rivest 1976.
[14] Murthy 1998.
[15] Ben-Gal et al. 2014.
[16] Gey and Poggi 2015.

- *Wurzelknoten*: Er repräsentiert die gesamte Population oder Stichprobe und wird weiter in zwei oder mehr homogene Sätze unterteilt.
- *Aufteilen (engl. split / splitting)*: Es ist ein Prozess, einen Knoten in zwei oder mehr Unterknoten aufzuteilen.
- *Entscheidungsknoten*: Wenn sich ein Unterknoten in weitere Unterknoten aufteilt, dann wird er Entscheidungsknoten genannt.
- *Blatt- / Endknoten*: Nicht geteilte Knoten werden Blatt- oder Endknoten genannt.
- *Pruning*: Bezeichnet den Prozess des Entfernens von Unterknoten eines Entscheidungsknotens. Diese Methode wird verwendet, um die Tiefe eines Baums einzuschränken.
- *Zweig / Unterbaum*: Ein Unterabschnitt des gesamten Baums wird als Zweig oder Unterbaum bezeichnet.
- *Feature*: Messbare Eigenschaft oder Ausprägung in einem Datensatz
- *Eltern- und Kindknoten*: Ein Knoten, der in Unterknoten unterteilt ist, wird als Elternknoten von Unterknoten bezeichnet, während Unterknoten die Kinder eines Elternknotens sind.

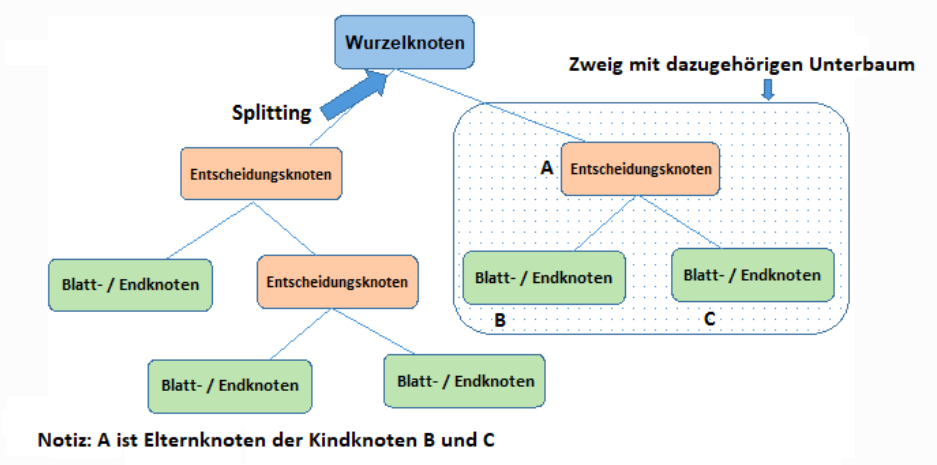



*Abbildung 2.1 Veranschaulichung Terminologie Entscheidungsbaum*

Die Entscheidung, strategische Aufteilungen vorzunehmen, wirkt sich stark auf die Genauigkeit eines Baums aus. Die Entscheidungskriterien sind für Klassifikations- und Regressionsbäume unterschiedlich.[17]

Entscheidungsbäume verwenden mehrere Algorithmen, um zu entscheiden, einen Knoten in zwei oder mehr Unterknoten aufzuteilen. Das Anlegen von Unterknoten erhöht die Homogenität der resultierenden Unterknoten. Dies impliziert, dass die Reinheit des Knotens

[17] Max Bramer 2007.

in Bezug auf die Zielvariable zunimmt. Der Entscheidungsbaum teilt die Knoten auf alle verfügbaren Variablen und wählt dann die Aufteilung aus, die zu den homogensten Unterknoten führt.[18]

Der Aufteilungsprozess besteht aus den folgenden Schritten:

1. Es beginnt mit der ursprünglichen Menge S als Wurzelknoten (vgl. Abbildung 2.2).
2. Bei jeder Iteration des Algorithmus iteriert der Algorithmus durch die ungenutzten Attribute der Menge S und berechnet die Entropie (H) und den Informationsgewinn (IG) dieser Attributmenge.
3. Das Attribut mit der kleinsten Entropie oder dem größten Informationsgewinn wird für das weitere Vorgehen selektiert.
4. Die Menge S wird anschließend durch das ausgewählte Attribut geteilt, um eine Teilmenge der Daten zu erzeugen (vgl. Abbildung 2.3).
5. Der Algorithmus wiederholt sich weiterhin für jede Teilmenge und berücksichtigt nur Attribute, die noch nicht zuvor ausgewählt wurden.
6. Das Ende ist erreicht, wenn alle Attribute kategorisiert sind.


*Abbildung 2.2 Beispiel einer ursprünglichen Menge vor dem ersten Split*[19]

[18] Gareth 2017.
[19] Vgl. Vanderplas– Creating a Decision Tree

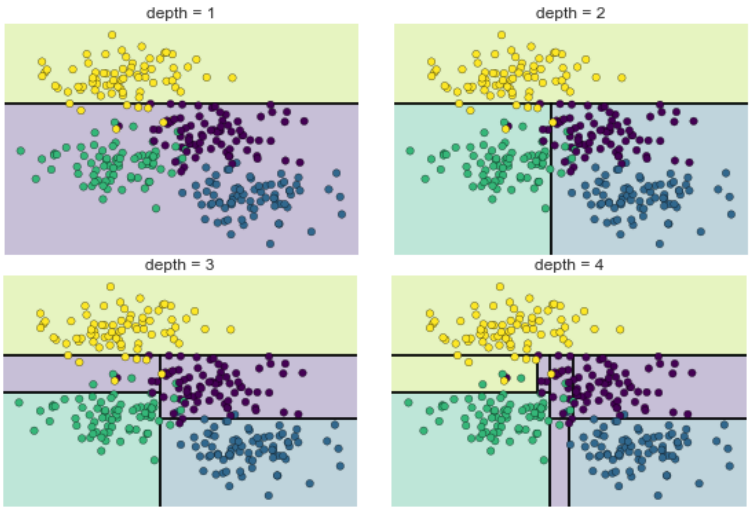



*Abbildung 2.3 Teilung der ursprünglichen Menge anhand von Split 1-4*[20]

Bei hinreichend großen Datensätzen ist die Entscheidung, welches Attribut an der Wurzel oder auf verschiedenen Ebenen des Baums als interne Knoten platziert werden soll, ein aufwändiger Schritt. Wie bereits bei Schritt drei angedeutet gibt es präzise Berechnungsmethoden um aussagekräftige Kennzahlen, auf die die Entscheidungen beruhen, zu erhalten. Darunter fallen in Bezug auf Entscheidungsbäumen z.B. der Informationsgewinn und der Gini-Index.

Informationsgewinn ist eine statistische Eigenschaft, die misst, wie gut ein gegebenes Attribut die Trainingsbeispiele nach ihrer Zielklassifikation trennt. Der Informationsgewinn wird verwendet, um zu entscheiden, auf welches Merkmal bei jedem Schritt beim Aufbau des Baums aufgeteilt werden soll. Es wird oft versucht den Baum so klein wie möglich zu halten, um eine Einfachheit des Modells zu gewährleisten. Dazu wird bei jedem Schritt die Aufteilung gewählt, die zu den konsistentesten untergeordneten Knoten führt. Ein häufig verwendetes Maß für die Konsistenz wird als Informationsgehalt bezeichnet, die in Bits gemessen wird. Für jeden Knoten des Baums stellt der Informationswert die erwartete Informationsmenge dar, die benötigt würde, um anzugeben, ob eine neue Instanz mit Ja oder Nein klassifiziert werden sollte, wenn das Beispiel diesen Knoten erreicht hat.[21]

Der Gini-Index kann als Kostenfunktion verstanden werden, die verwendet wird, um Aufteilungen im Datensatz auszuwerten. Der Index wird berechnet, indem die Summe der quadrierten Wahrscheinlichkeiten jeder Klasse von eins subtrahiert wird. Es bevorzugt größere Partitionen und ist einfach zu implementieren, während Informationsgewinn kleinere

[20] Vgl. Vanderplas 2017 – Decision Tree Levels.
[21] Witten et al. 2011.

Partitionen mit unterschiedlichen Werten bevorzugt. Gini-Index arbeitet mit der kategorialen Zielvariable „Erfolg“ oder „Fehler“. Die Verwendung eines Gini-Index kann daher nur zu Binären-Splits führen. Ein höherer Wert des Gini-Index impliziert eine höhere Ungleichheit und eine höhere Heterogenität.[22]

Diese Kriterien berechnen Werte für jedes Attribut. Die Werte werden sortiert und die Attribute werden in der Reihenfolge im Baum platziert, d. h. das Attribut mit dem höchsten Wert (bei Informationsgewinn) wird an der Wurzel platziert.

Ein häufiges Problem bei Entscheidungsbäumen ist ein sogenanntes Over Fitting. Over Fitting ist ein Konzept in der Datenwissenschaft, das auftritt, wenn ein statistisches Modell genau mit seinen Trainingsdaten übereinstimmt.[23] In diesem Fall kann der Algorithmus nicht auf unsichtbare Daten reagieren, was seinen Zweck verfehlt. Die Verallgemeinerung eines Modells auf neue Daten ermöglicht es Unternehmen, jeden Tag maschinelle Lernalgorithmen zu verwenden, um Vorhersagen zu treffen und Daten zu klassifizieren. Wenn Algorithmen für maschinelles Lernen konstruiert werden, verwenden sie einen Beispieldatensatz, um das Modell zu trainieren. Wenn das Modell jedoch zu lange mit Beispieldaten trainiert oder wenn das Modell zu komplex ist, kann es beginnen, das „Rauschen“ oder irrelevante Informationen innerhalb des Datensatzes zu lernen.[24] Wenn das Modell das Rauschen speichert und sich zu eng an den Trainingssatz anpasst, wird das Modell „überangepasst“ und kann nicht mehr flexibel auf neue Daten verallgemeinern. Wenn ein Modell nicht gut auf neue Daten verallgemeinern kann, kann es die Klassifizierungs- oder Vorhersageaufgaben, für die es vorgesehen ist, nicht ausführen.

Niedrige Fehlerquoten und eine hohe Varianz sind gute Indikatoren für eine Überanpassung. Um diese Art von Verhalten zu verhindern, wird normalerweise ein Teil des Trainingsdatensatzes als „Testsatz“ beiseitegelegt, um auf Überanpassung zu prüfen. Wenn die Trainingsdaten eine niedrige Fehlerrate und die Testdaten eine hohe Fehlerrate aufweisen, signalisiert dies eine Überanpassung.[25] Am vorherigen Beispiel (vgl. Abbildung 2.3) der Aufteilung von Datenmengen wird dies sehr gut erkenntlich. Dieses Verhalten kann ein Indiz dafür sein, dass dies weniger auf die wahre, intrinsische Datenverteilung als vielmehr auf die besonderen Abtast- oder Rauscheigenschaften der Daten zurückzuführen ist. Das heißt,

[22] Schmoldt et al. 1975.
[23] Hothorn et al. 2006.
[24] Strobl et al. 2009.
[25] Hothorn et al. 2006.

dieser Entscheidungsbaum selbst auf nur fünf Ebenen ist deutlich „over fitted“ zu den vorhandenen Daten. (vgl. Abbildung 2.4). Eine Möglichkeit dies entgegenzuwirken ist das im folgenden Kapitel vorgestellte Verfahren.

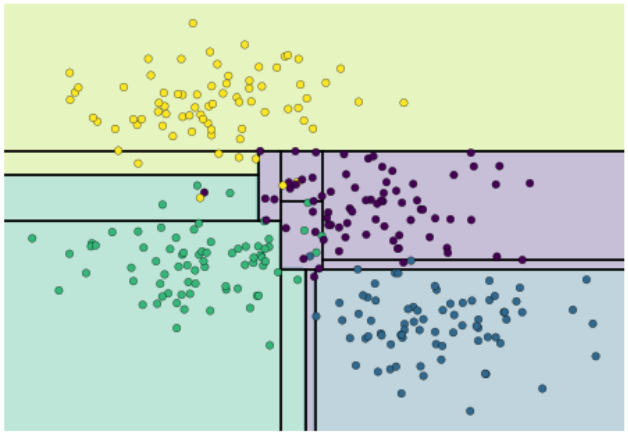


*Abbildung 2.4 Entscheidungsbaum Tiefe 5 zur Veranschaulichung von Over Fitting*[26]

## 2.2 Random Forest

Random Forests haben bei hochdimensionalen und schlecht gestellten Klassifikations- und Regressionsaufgaben an Popularität gewonnen, zum Beispiel auf Mikroarrays,[27] Zeitreihen,[28] oder Spektraldaten,[29] [30] aber auch für Inferenz in Anwendungen wie Bildsegmentierung oder Objekterkennung in Computer Vision.[31] [32] Random Forests sind vergleichbar in der Leistung mit vielen anderen nichtlinearen Lernalgorithmen. Sie kommen oft mit wenig Parameter-Tuning gut zurecht[33] und sind in der Lage präzise Modelle, anhand relevanter Feature-Subsets, zu generieren. Diese Eigenschaft kann sogar gewährleistet werden wenn eine große Menge irrelevanter Prädiktoren vorhanden sind.[34] [35] [36] [37] In jüngerer Zeit haben zusätzliche Eigenschaften des Random Forest an Interesse gewonnen, beispielsweise bei der Merkmalauswahl36 oder der explorativen Analyse von Stichprobennähe.[38] Die Hauptidee des

[26] Vgl. Vanderplas – Decision Tree Overfitting
[27] Jiang et al. 2004.
[28] Shen et al. 2007.
[29] Svetnik et al. 2003.
[30] Pal 2005, pp. 217–222.
[31] A. Chriminasi 2009.
[32] 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2011.
[33] Hastie et al. 2009.
[34] Scornet et al. 2015.
[35] Lin and Jeon 2006, pp. 578–590.
[36] Menze et al. 2007, pp. 1801–1807.
[37] Gérard Biau et al. 2008, pp. 2015–2033.
[38] Xiaogang Su et al. 2009, pp. 141–158.

Random Forest Frameworks, vorgeschlagen von Breiman,[39] besteht darin, viele variable, aber unvoreingenommene Basislerner durch die Bündelung eines ganzen Ausschusses von Prädiktoren zu lernen und die Varianz zu reduzieren. Dieses Konzept ist bekannt als bagging.[40] Die allgemeine Methode der Random Forests wurde erstmals 1995 von Ho vorgeschlagen.[41] Ho stellte fest, dass Wälder von Bäumen, die sich mit schrägen Hyperebenen aufspalten, mit zunehmendem Wachstum an Genauigkeit gewinnen können, ohne unter Übertraining zu leiden, solange die Wälder zufällig eingeschränkt werden, um nur auf ausgewählte Merkmalsdimensionen empfindlich zu sein. Eine nachfolgende Arbeit in der gleichen Richtung[42] kam zu dem Schluss, dass sich andere Aufteilungsmethoden ähnlich verhalten, solange sie zufällig gezwungen werden gegenüber einigen Merkmalsdimensionen unempfindlich zu sein. Diese Beobachtung, dass ein komplexerer Klassifikator (ein größerer Wald) fast monoton genauer wird, steht in scharfem Gegensatz zu der allgemeinen Meinung, dass die Komplexität eines Klassifikators nur bis zu einem bestimmten Genauigkeitsgrad wachsen kann, bevor er durch Überanpassung beeinträchtigt wird. Die Erklärung der Übertrainingsresistenz der Waldmethode findet sich in Kleinbergs Theorie der stochastischen Diskriminierung.[43] [44] [45]

Die frühe Entwicklung von Breimans Begriff der Random Forests wurde durch die Arbeit von Amit und Geman[46] beeinflusst, die die Idee einführten, beim Aufteilen eines Knotens eine zufällige Teilmenge der verfügbaren Entscheidungen zu durchsuchen, im Zusammenhang mit der Züchtung eines einzelnen Baums. Die Idee der zufälligen Unterraumauswahl von Ho[42] war auch einflussreich beim Design von Random Forests. Bei diesem Verfahren wird ein Wald von Bäumen gezüchtet und eine Variation zwischen den Bäumen eingeführt, indem die Trainingsdaten in einen zufällig ausgewählten Unterraum projiziert werden, bevor jeder Baum oder jeder Knoten angepasst wird. Schließlich wurde die Idee der randomisierten Knotenoptimierung, bei der die Entscheidung an jedem Knoten durch ein randomisiertes Verfahren und nicht durch eine deterministische Optimierung ausgewählt wird, zuerst von

[39] Breiman 2001.
[40] Breiman 1996.
[41] Ho 1995.
[42] Ho 1998.
[43] Kleinberg 1990.
[44] Kleinberg 1996.
[45] Kleinberg 2000.
[46] Amit and Geman 1997.

Dietterich eingeführt.[47] Die richtige Einführung von Random Forests wurde in einem akademischen Artikel von Breiman vorgestellt.[48]

Zusammenfassend lassen sich zwei Schlüsselkonzepte festhalten:

1. Wahl einer zufälligen Stichprobe von Trainingsdatensätzen beim Erstellen von Entscheidungsbäumen.
2. Selektion eine zufällige Teilmenge von Features, die bei der Auswahl von Knoten berücksichtigt werden.

Diese Konzepte werden, mit dem bereits erwähnten bagging kombiniert, um ein Ensemble von Bäumen zu erzeugen. Unter Verwendung eines einzigen Lernalgorithmus wird ein Modell auf allen Stichproben erstellt, um die resultierenden Vorhersagen zu kombinieren. Dies erfolgt durch eine parallele Abstimmung oder Mittelung der Vorhersagen.

Am Over Fitting Beispiel aus Kapitel 2.1.1 (vgl. Abbildung 2.4) lässt sich durch die Mittelung von über 100 randomisierten Modellen ein Gesamtmodell erstellen, das unserer Intuition über die Aufteilung des Parameterraums näherkommt (vgl. Abbildung 2.5).

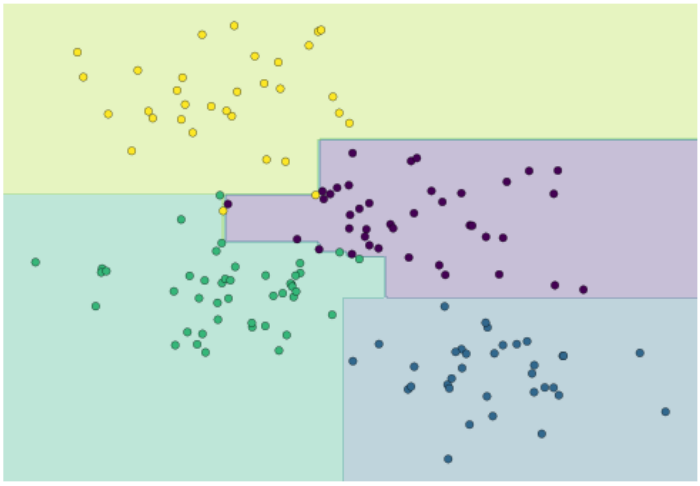


*Abbildung 2.5 Random Forest mit Mittelung von ~100 Entscheidungsbäumen[49]*

## 2.3 Oblique Forest

Die von Ho festgestellte Eigenschaft in Bezug auf die Aufspaltung der Feature-Subsets mit schrägen Hyperebenen[50] wurde beim klassischen Random Forest nicht integriert. Random Forest gehört zu dem Ensemble von „achs-ausgerichteten“ Entscheidungsbäumen. Bei

[47] Dietterich 2000.
[48] Breiman 2001.
[49] Vgl. Vanderplas – Decision Tree Overfitting
[50] Ho 1995.

solchen Entscheidungsbäumen wird der Merkmalraum entlang Richtungen parallel zu den Koordinatenachsen rekursiv aufgeteilt. Wenn Klassen entlang einer einzelnen Dimension untrennbar erscheinen, erfordern achsorientierte Aufteilungen sehr tiefe Bäume mit komplizierten stufenartigen Entscheidungsgrenzen, die zu erhöhter Varianz und Überanpassung führen.[51] [52] Um dies anzugehen, hat Breiman den Forest-RC (F-RC) vorgeschlagen und charakterisiert, das dieser auf lineare Kombinationen von Koordinaten anstatt auf einzelne Koordinaten aufspaltet.[53] Diese sogenannten „schrägen" Ensembles umfassen als Sonderfall die achsenorientierten Ensembles und haben daher eine erhöhte Ausdrucksfähigkeit, die potenziell bessere Lerneigenschaften verleiht (vgl. Abbildung 2.6).

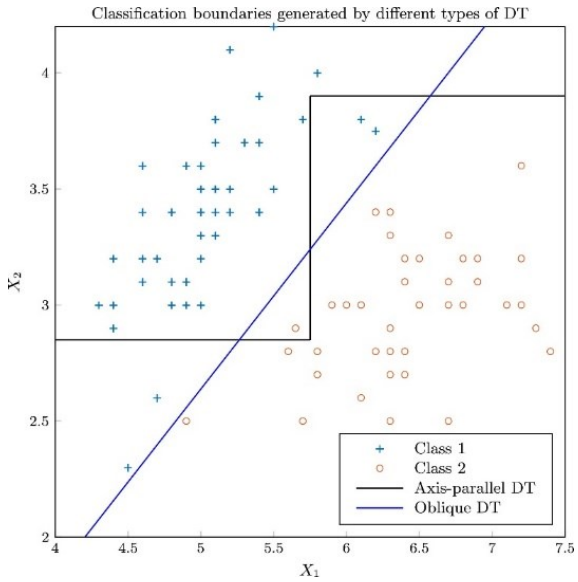



*Abbildung 2.6 Klassifikationsgrenzen von Entscheidungsbäumen[54]*

Leider verlieren diese Verfahren viele der (vgl. Kapitel 2.1) wünschenswerten Eigenschaften, die achsenausgerichtete Bäume besitzen, wie Recheneffizienz, einfache Abstimmung, Unempfindlichkeit gegenüber einem großen Anteil irrelevanter (Rauschen-)Eingaben und Interpretierbarkeit.

Hingegen bekommen achsorientierte Verfahren Schwierigkeiten bei kollinearen Daten mit korrelierten Merkmalen[55] die zwischen den Koordinatenachsen liegen. Dies ist z.B. der Fall bei Zeitreihen aber auch Bildfeldern, da diese in Unterräumen liegen können. In diesem Fall können Klassenverteilungen untrennbar erscheinen, wenn Randverteilungen bei der Suche

[51] Ho 1995.
[52] Ho 1998.
[53] Breiman 2001.
[54] Vgl. Katuwal et al. 2020Classification boundaries generated by axis-parallel and oblique decision tree in a toy example for binary classification problem.
[55] Menze et al. 2009.

nach der besten univariaten Aufteilung ausgewertet werden. Die Trennung von Klassen kann komplexe und tief verschachtelte Bäume erfordern. Multivariate Entscheidungsbäume, Bäume mit Entscheidungsflächen in beliebigen, schrägen Orientierungen zu den Achsen, können besser an Entscheidungen in solchen Unterräumen angepasst werden,[56] was zu Entscheidungsgrenzen führt, die weniger durch geometrische Beschränkungen des Basislerners beeinflusst werden.

## 2.4 Crisp DM

CRISP-DM steht für Cross Industry Standard Process for Data Mining und ist eine Methode aus dem Jahr 1996, die entwickelt wurde, um Data Mining-Projekte zu gestalten. Es besteht aus sechs Schritten, um ein Data Mining-Projekt zu konzipieren. Sie können je nach Bedarf der Entwickler zyklische Iterationen haben. Diese Schritte sind Geschäftsverständnis, Datenverständnis, Datenvorbereitung, Modellierung, Auswertung und Bereitstellung.[57] [58]

**Business Understanding (Aufgabendefinition)**

Im ersten Schritt wird darauf abgezielt den Projektzielen und den verwendeten Daten einen Kontext zu verleihen. Anhand dessen können die Entwickler/Ingenieure eine Vorstellung von der Relevanz der Daten für das vorliegende Geschäftsmodell erhalten. Es werden umfangreichen Meetings, Dokumentationen und unternehmen spezifische Informationen bereitgestellt, um dem Entwicklungsteam zu helfen und Fragen zum relevanten Einsatzgebiet zu stellen.

Am Ende des Business Understanding wird ein klar definiertes Ziel festgelegt und wie dieses mit Hilfe der vorliegenden Daten erreicht werden soll. Zum Beispiel kann das Ziel darin bestehen, den Umsatz zu steigern. Nach Abschluss des Business Understanding wurde zudem definiert, was der Kunde verkauft und wie der Verkaufsablauf erfolgt.

**Data Understanding (Auswahl der relevanten Datenbestände)**

Das Data Understanding zielt auf das Verständnis der vorliegenden Daten ab. Mit Hilfe des zuvor definierten Ziels werden Erwartungen an spezifische Datenausprägungen überprüft und eine Evaluation erstellt, ob dieses mit den vorliegenden Daten erreicht werden kann. Es werden zudem die Qualität der Daten in mehreren Punkten, wie z. B. Datenvollständigkeit,

[56] Murthy et al. 1994.
[57] Colin Shearer 2000.
[58] Pete Chapman 1999.

Werteverteilungen und Data Governance-Compliance überprüft. Dies ist ein entscheidender Teil des Projekts. Es definiert, wie tragfähig und vertrauenswürdig die Endergebnisse sein können und welche Informationen abgeleitet werden können. Anhand des Geschäftsverständnisses lassen sich auch die Relevanz oder die Verwendung eines Datenelements für das Entwicklungsteam festlegen, damit eventuelle Unklarheiten vermieden werden können. Das Verständnis dient auch dazu entscheiden zu können welche Algorithmen bzw. Verfahren sich für den Datensatz eignen und was im nächsten Schritt bereinigt oder angepasst werden muss damit diese zum definierten Ziel führen.

**Data Preparation (Datenaufbereitung)**

Die Datenvorbereitung beinhaltet den ETLs- oder ELTs-Prozess. Der Prozess steht für das Extrahieren, Transformieren und Laden von Daten. Der aktuelle Trend der Dezentralisierung[59] von Organisations- und Unternehmensstrukturen wirkt sich vor allem auf die Speicherung von Daten aus. So kommt es zu einer erhöhten Verwendung von Cloud- und verteilten Datenbanksystemen. Auf diese Eigenschaft muss beim Extrahieren geachtet werden, damit dem Entwicklerteam alle Daten vorliegen.

Mit aktuellen Prognosen von einer Steigerung des täglichen Internetverkehrs von mehr als 200%[60] steigt auch die Bedeutung die Unternehmen Data-Governance-Richtlinien zuweisen. Es kommt trotzdem häufig vor, dass diese in einer Organisation nicht respektiert oder festgelegt werden. Um den Daten eine echte Bedeutung zu verleihen, ist es die Aufgabe von Datenwissenschaftler, die Daten zu standardisieren. Ebenso funktionieren einige Algorithmen unter bestimmten Parametern besser, so können einige nicht auf numerischen oder nominalen Werten angewendet werden. Auch können Werte mit einer großen Varianz zu Problemen führen. Hierbei liegt es am Entwicklungsteam, die Daten zu normalisieren.

Wie im einleitenden Zitat erwähnt kann dieser Schritt einen Großteil der Bearbeitungszeit in Anspruch nehmen. Er ist zudem kritisch für die folgenden Abläufe und entscheidet schlussendlich, ob es zu aussagekräftigen Ergebnissen kommt.

[59] Organisation for Economic Co-operation and Development 2019.
[60] Cisco Systems 2021.

**Modeling (Auswahl und Anwendung von Data Mining Methoden)**

Der vierte Schritt ist die Anwendung eines oder oft auch mehrerer Verfahren und ist der Kern jedes Machine-Learning-Projekts. Bei der Durchführung werden die Ergebnisse generiert die verantwortlich sind die Projektziele zu erfüllen oder zu deren Erfüllung beitragen sollen.

Obwohl es der glamouröse Teil des Projekts ist, ist es durchaus meistens der kürzeste. Die Verfahren arbeiten i.d.R. voll automatisch und es bedarf keines weiteren externen Einflusses.

Im Falle, dass die Ergebnisse verbesserungsfähig sind, wird die Methodik auf die Datenaufbereitung zurückgreifen und die verfügbaren Daten verbessern. Hierbei ist es üblich, dass es zu einer Wiederholung von Schritt drei und vier kommt, bis ein hinreichend genaues Ergebnis erreicht wurde oder keine weitere Optimierung mehr möglich ist.

**Evaluation (Bewertung und Interpretation der Ereignisse)**

In der Bewertungsphase wird überprüft ob die Ergebnisse gültig und richtig sind. Falls die Ergebnisse falsch sind, sieht die Methodik vor zum ersten Schritt zurückzukehren, um ein Verständnis zu erarbeiten, warum die Ergebnisse falsch ausgefallen sind.

Eine gängige Methode, um zu überprüfen, ob das Modell (Produkt des Modellierungsschritts) der Realität entspricht ist die Aufteilung der Daten in Training- und Testsubsets. Je nach Aufgabenstellung und Kontext gibt es unterschiedliche Techniken. Zum Beispiel im Kontext des überwachten Lernens mit der Aufgabe Items zu klassifizieren, ist eine Möglichkeit, die Ergebnisse anhand der Konfusionsmatrix zu überprüfen. Beim unüberwachten Lernen wird die Bewertung schwieriger, da es keinen statischen Wert gibt, um „richtig“ von „falsch“ zu trennen. In jedem Fall ist es wichtig, eine Fehlerquelle anzugeben. Dieses Fehlermaß sagt dem Benutzer, wie er den Ergebnissen vertrauen kann. Wenn das Fehlermaß in allen Fällen mit 0 oder keinem übereinstimmt, würde dies darauf hindeuten, dass das Modell überangepasst ist und die Realität sich möglicherweise anders verhält.

**Deployment (Anwendung der Ergebnisse)**

Der letzte Schritt ist das Deployment und besteht darin, die Ergebnisse auf nützliche und verständliche Weise zu präsentieren. Dadurch sollte das Projekt seine Ziele erreichen. Es ist der einzige Schritt, der nicht zu einem Zyklus gehört.

Je nach Endbenutzer kann eine nützliche und verständliche Art und Weise variieren. Wenn der Endbenutzer beispielsweise eine andere Software ist, wie im Verkaufs-Website-Programm, das sein Empfehlungssystem fragt, was er einem Käufer vorschlagen soll, wäre

eine JSON-Antwort mit der Antwort auf eine bestimmte Anfrage eine nützliche Methode. In einem anderen Fall, wie bei einer Führungskraft, die projizierte Informationen für die Entscheidungsfindung benötigt, können die Ergebnisse am besten präsentiert werden, indem sie in einer analytischen Datenbank gespeichert und als Dashboard auf einer Business-Intelligence-Lösung präsentiert werden.

# 3 Versuchsaufbau

## 3.1 Vorgehen

Der Arbeitsablaufs zur Modellerstellung ist in Abb. 3.1 skizziert. Für jeden Datensatz wird der gleiche Arbeitsablauf befolgt, um ein unabhängiges Modell für diesen Datensatz zu erstellen. Es wird zunächst geprüft, ob alle Datenmerkmale (Feature) für die Klassifikationsaufgabe wichtig sind. Dies erfolgt unter Verwendung mehrerer Feature-Ranking-Algorithmen. Features werden mit Hilfe der gängiger Ranking-Algorithmen bewertet und eingestuft. Focus der Algorithmen liegen dabei auf dem Informationsgewinn jedes Features, dem Informationsgewinn im Verhältnis zu dem Feature Set und der Korrelation. Diese Algorithmen bewerten/rangieren jedes der Merkmale im Datensatz im Kontext der Ausgabevariablen.[61] [62] [63] Sobald die geordneten Merkmale identifiziert sind, werden mehrere Kombinationen von Merkmalen aus den Merkmalen basierend auf ihren Rangfolgebewertungen ausgewählt. Schließlich werden beide Klassifikationsverfahren auf die ausgewählten Features angewendet, um das endgültige Modell zu trainieren und zu konstruieren.

Während des Modelltrainings wird eine 10-fache Kreuzvalidierung angewendet. Insbesondere ist die Kreuzvalidierung eine Methode zur Bewertung eines Vorhersagemodells durch Aufteilen der ursprünglichen Stichprobe in einen Trainingssatz zum Trainieren des Modells und einen Validierungs-/Testsatz zur Bewertung. Das Verhältnis der Aufteilung liegt dabei bei 80/20 von Training zu Testsatz. Bei der 10-fachen Kreuzvalidierung werden die Originalstichproben nach dem Zufallsprinzip in 10 gleich große Unterstichproben aufgeteilt, und von diesen Unterstichproben wird eine einzelne Unterstichprobe als Validierungsdaten

[61] DASH and LIU 1997.
[62] Ruiz et al. 2003.
[63] Novakovic et al. 2011.

zum Testen des Modells beibehalten, während die restlichen 9 Unterstichproben als Trainingsdaten verwendet werden.

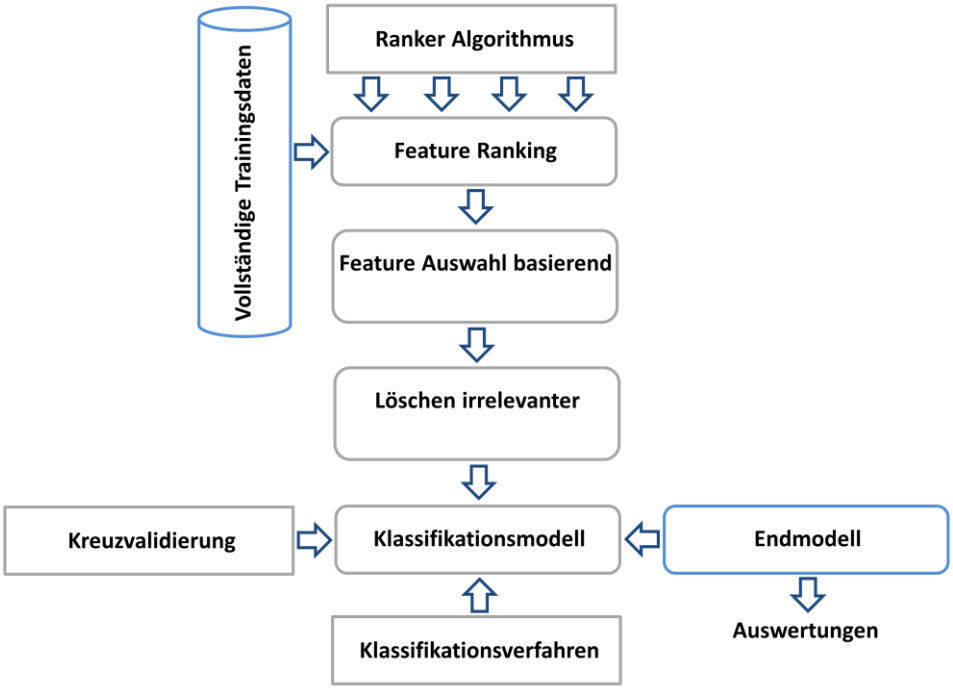



*Abbildung 3.1 Skizzierung Versuchsablauf*

### 3.1.1 Optimierung von Hyperparameter

Ganz allgemein ist ein Hyperparameter ein Parameter des Modells, der vor dem Start des Lernprozesses eingestellt wird. Unterschiedliche Modelle haben unterschiedliche Hyperparameter, die eingestellt werden können. Für die zu untersuchenden Verfahren bietet die verwendete Bibliothek sieben Parameter für Random Forest und vier für Oblique Forest, wobei sich letzteres Verfahren die Parameter mit dem RF teilt. Die drei für den Random Forest spezifischen Hyperparameter wurden folgend am Ende platziert.

- Die Anzahl der Entscheidungsbäume im Wald
- Die maximale Tiefe der einzelnen Bäume. Je größer ein einzelner Baum ist, desto größer ist die Wahrscheinlichkeit, dass die Trainingsdaten überangepasst werden. Random Forests weisen aber viele einzelne Bäume auf, wodurch dieses Problem bereits minimiert wird.
- Die minimalen Stichproben, die an einem internen Knoten der Bäume aufgeteilt werden sollen. In Kombination mit dem vorherigen Parameter, lassen sich die einzelnen Bäume bei Bedarf normalisieren.

- Maximale und Minimale Anzahl von Blattknoten. In Random Forest ist dieser Parameter weniger relevant in Vergleich zu Modellen mit einzelnen Entscheidungsbäumen. Bei diesen kann der Parameter großen Einfluss darauf nehmen, Überanpassungen zu reduzieren und die Erklärbarkeit des Baums zu erhöhen, indem die mögliche Anzahl von Pfaden zu Blattknoten reduziert wird.
- Anzahl der zufälligen Features, die an jedem Knoten zum Aufteilen eingefügt werden sollen.
- Die Größe des Bootstrapping-Datasets, mit dem jeder Entscheidungsbaum trainiert werden soll.
- Die Kriterien, nach denen auf jeden Knoten aufgeteilt werden soll (Gini oder Entropie für eine Klassifizierungsaufgabe oder MSE oder MAE für die Regression)

Die verschiedenen Hyperparameter werden auf dem Trainingssatz getestet und sobald die optimierten Parameterwerte ausgewählt wurden, wird ein Modell unter Verwendung der gewählten Parameter und des Testsatzes konstruiert und dann auf dem Trainingssatz getestet um zu überprüfen, wie genau das Modell den Datensatz klassifiziert. Es gibt verschiedene Möglichkeiten für die Wahl der Hyperparameter, die für ein vorliegendes Modell angepasst werden sollen. Eine gute Möglichkeit zur visuellen Überprüfung auf potenziell optimierte Werte von Modellhyperparametern ist eine Validierungskurve. Eine Validierungskurve kann in einem Diagramm gezeichnet werden, um zu zeigen, wie gut ein Modell mit unterschiedlichen Werten eines einzelnen Hyperparameters abschneidet.

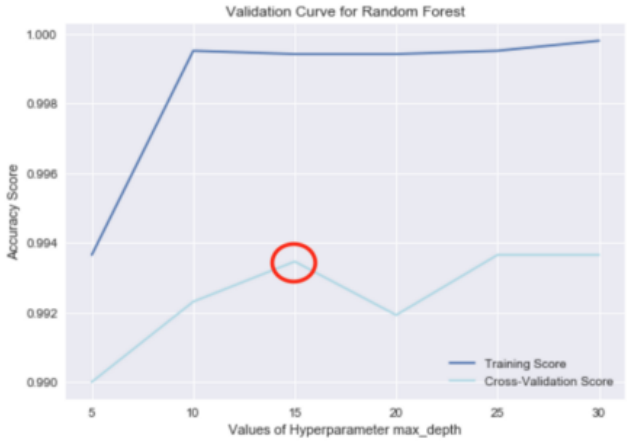



*Abbildung 3.2 Validierungskurve Hyperparameter maximale Tiefe*[64]

Anhand der Abbildung 3.2 ist erkennbar, dass der höchste Genauigkeitswert bei der Kreuzvalidierung nahe bei 99,3 % liegt, wenn max_depth auf 15 oder >25 gewählt wird. Zu

[64] Reilly Meinert 2019.

beobachten ist auch, dass sich der Genauigkeitswert ab dem max_depth Wert 25 nicht weiter variiert. Dies kann ein Hinweis für ein potentielles over-fitting sein und sollte daher vermieden werden. Die Wahl des Hyperparameters würde in diesem Fall auf 15 festgelegt werden.

Neben der Untersuchung von Validierungsmetriken kommt die Rastersuche oft zum Einsatz. Rastersuche bezieht sich auf eine Technik, die verwendet wird, um die optimalen Hyperparameter für ein Modell zu identifizieren. Im Gegensatz zu Parametern ist das Auffinden von optimalen Hyperparametern in Trainingsdaten unerreichbar. Um die richtigen Hyperparameter zu finden, kann daher ein Modell für jede Kombination von Hyperparametern erstellt werden. Die Rastersuche gilt daher als eine sehr traditionelle Hyperparameter-Optimierungsmethode, da im Grunde alle möglichen Kombinationen probiert werden. Die Modelle werden anschließend durch Kreuzvalidierung evaluiert. Um die Optimierung in den Versuchsablauf (vgl. Abbildung 3.1) zu integrieren, wird für die Erstellung des Klassifikationsmodells ein weiterer Bearbeitungsschritt hinzugefügt (vgl. Abbildung 3.3). Hierbei wird zuerst ein erstes Modell zur Klassifikation, anhand des Verfahrens erstellt und anschließend validiert. Basierend auf diesem Modell kann eine Rastersuche für die Optimierung durchgeführt werden und sukzessive eine Anpassung der Parameter erfolgen. Nach jeder Änderung der Hyperparameter wird ein neues Klassifikationsmodell zur Verifizierung erstellt. Sollten keine Optimierungen mehr möglich sein, wird das Modell als Endmodell deklariert und für weitere Auswertungen hinterlegt.

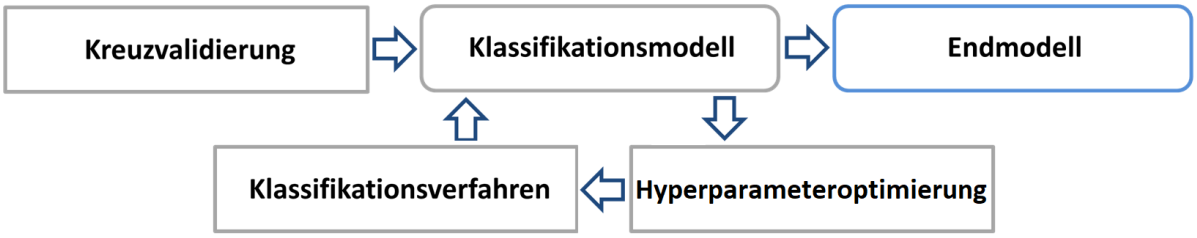



*Abbildung 3.3 Anpassung des Versuchsablaufs für das Klassifikationsmodell*

## 3.2 Rahmenbedingungen

Alle Experimente wurden auf einer CPU Intel(R) Core (TM) i7-8700k (@ 4.7GHz) mit 64 GB RAM, GPU NVIDIA GeForce RTX 3080Ti und dem Betriebssystem Windows 10 durchgeführt. Zur Anwendung verschiedener Klassifikationsalgorithmen wurde Python 3.9.2 und die scikit-learn Bibliothek mit der Entwicklungsumgebung Visual Studio Code verwendet. Dabei liefert die Bibliothek direkt eine Implementierung des Random Forest Algorithmus. Für die Realisierung einer Oblique Forest Implementierung wurde die Bibliothek Oblique „Decision

Trees“ in Python für das sklearn-Paket von Mulyar[65] basierend auf der C Implementation von Murthy et al.[66] erweitert. Die Erweiterung der „Oblique Forest Tree“ Implementation orientiert sich dabei an der von Norouzi et al.[67] vorgeschlagenen CO2 Forest Optimierung.

## 3.3 Qualitätskriterien

Die Leistung der Modelle kann mit verschiedenen gängigen Metriken bewertet werden. Folgend werden unterschiedliche Schätzfunktionen kurz erläutert. In Rahmen dieser Arbeit kamen beide Schätzverfahren zum Einsatz. Für einen Gesamtvergleich zwischen 150 Klassifikatoren wurde ein Durchschnittswert basierend auf der Genauigkeit berechnet. Bei der Wahl der Datensätze wurde auf eine hinreichende Größe geachtet damit die Ergebnisse durch die Aufteilung in Test- und Trainingsdaten nicht verfälscht werden. Zudem zeigte die Genauigkeit eine kleinere Varianz bei der Modellerstellung auf. Dies war eine entscheidende Eigenschaft für die automatisierte Modellerstellung aller 150 Datensätze. Für die genauere Betrachtung anhand ausgewählter Datensätze wurde die OOB betrachtet. Bei einigen selektierten Datensätzen handelt es sich um künstlich erstellte, mit dem Ziel das Verhalten der Methoden, in bestimmten Scenarios zu beobachten. Die OOB lieferte dafür ein genaueres Ergebnis.

**k-fache Kreuzvalidierungs-Schätzfunktionen**

Genauigkeit $A = \frac{T_p+T_n}{T_p+T_n+F_p+F_n}$,[68] ist das intuitivste Leistungsmaß und ist ein Verhältnis der korrekt vorhergesagten Beobachtungen zu den Gesamtbeobachtungen.

Sensitivität (oder Recall) $R = \frac{T_p}{T_p+F_n}$ Recall ist das Verhältnis der korrekt vorhergesagten positiven Beobachtungen zu allen Beobachtungen in der tatsächlichen Klasse.

Präzision $P = \frac{T_p}{T_p+F_p}$, ist das Verhältnis der korrekt vorhergesagten positiven Beobachtungen zu den gesamten vorhergesagten positiven Beobachtungen.

Diese Größen beziehen sich auch auf den F1-Score $F1 = 2\frac{P\times R}{P+R}$, was definiert ist als das harmonische Mittel von Präzision und Recall.

[65] Andriy Mulyar 2020.
[66] Murthy et al. 1994.
[67] Norouzi et al. 2015.
[68] Ergebnis der Einteilung (Richtig o. Falsch), Zuordnung der Klassifikation (positiv o. negativ) dabei ergeben sich vier mögliche Fälle:
$T_p$: $Richtig\ u.positiv$ | $T_n$: $Richtig\ u.negativ$ | $F_p$: $Falsch\ u.positiv$ | $F_n$: $Falsch\ u.negativ$

**Out-of-Bag (OOB) Schätzfunktion**

Wie in Kapitel 2.2 erklärt, verwendet der Random Forest Algorithmus grundsätzlich zwei Methoden. Das Bagging und der zufälligen Wahl von Features statt dem gesamten Feature-Set.

Angenommen, es werden für die Berechnung des Waldes eine Anzahl von Bäumen $S$ festgelegt (z.B.: mit Hilfe des Hyperparameters). Mit Hilfe dieser werden zuerst $S$ Datensätze mit "gleicher Größe wie das Original" verarbeitet, die durch zufälliges Resampling von Daten in der Trainingsmenge $T$ mit Ersetzung (n-mal für jeden Datensatz) erstellt wurden. Dies führt zu $\{T1, T2, \ldots TS\}$-Datensätzen. Jeder dieser Datensätze wird als Bootstrap-Dataset bezeichnet. Durch "mit Ersetzung" kann jeder Datensatz $Ti$ doppelte Datensätze enthalten und zudem können mehrere Datensätze aus dem Originaldatensatz in $Ti$ fehlen. Dies wird Bootstrapping genannt.

Bagging ist der Prozess, bei dem Bootstraps genommen und dann die auf jedem Bootstrap gelernten Modelle aggregiert werden.

Basierend auf den Datensätzen Ti erstellt der Random Forest Algorithmus S Bäume und verwendet $m$ $(= sqrt(M)$ oder $= floor(lnM + 1))$ zufällige Untermerkmale aus $M$ möglichen Merkmalen, um einen beliebigen Baum zu erstellen. Dies wird als zufällige Unterraummethode bezeichnet.

Für jedes $Ti$-Bootstrap-Dataset wird somit ein Baum Ki erstellt. Zur Klassifizierung der Eingabedaten $D = \{x1, x2, \ldots, xM\}$ durchlaufen diese jeden Baum und erzeugen $S$ Ausgaben $Y = \{y1, y2, \ldots, y\}$. Die endgültige Vorhersage ist eine Mehrheitsabstimmung für dieses Set.

Nachdem die Klassifikatoren ($S$-Bäume) erstellt wurden, wird für alle $(Xi, yi)$ im ursprünglichen Trainingssatz $T$ überprüft, ob diese enthalten sind, und wählt alle $Tk$ aus, die $(Xi, yi)$ nicht enthalten. Zu beachten ist, dass diese Teilmenge eine Reihe von Boostrap-Datensätzen ist, die keinen bestimmten Datensatz aus dem ursprünglichen Datensatz enthalten. Dieses Set wird Out-of-bag-Beispiele genannt. Es gibt n solcher Teilmengen (eine für jeden Datensatz im Originaldatensatz $T$). Der OOB-Klassifizierer ist die Aggregation von Stimmen NUR über $Tk$, so dass er $(xi, yi)$ nicht enthält. Die Out-of-bag-Schätzung für den Generalisierungsfehler ist die Fehlerrate des Out-of-bag-Klassifikators auf dem Trainingssatz.

Die Untersuchung von Fehlerschätzungen für bagging Klassifikatoren in Breiman[69] liefert empirische Beweise dafür, dass die OOB-Schätzung genauso genau ist wie die Verwendung eines Testsatzes der gleichen Größe wie der Trainingssatz. Daher beseitigt die Verwendung der OOB-Fehlerschätzung die Notwendigkeit eines beiseitegelegten Testsatzes. Um ein Aussagekräftige Ergebnis zu erhalten, kommt diese Eigenschaft vor allem bei kleinen Datensätzen zu tragen.

## 3.4 Datensätze

Die Bewertung anhand der Leistung der Klassifizierungsverfahren benötigt eine diverse Auswahl an Datensätzen. Dies stellt ein aussagekräftiges Ergebnis am Ende der Auswertung sicher. Um einen ersten Gesamtvergleich zwischen beiden Klassifizierungsverfahren herzustellen wurden 70 „zufällig“ ausgewählte UCI[70] Datensätze analysiert. Die Auswahlkriterien für die Wahl der einzelnen Datensätze wurde dabei vorab eingeschränkt. Wichtig erschien eine möglichst weitreichende Selektion sicherzustellen. Es wurde daher darauf geachtet unterschiedliche Charakteristiken der Datensätze, z.B. multivariate und sequenzielle Datensätze auszuwählen.

Nach dem ein Leistungsbild anhand des Gesamtvergleichs erstellt wurde, kommt es zu einer Untersuchung von speziell selektierten Datensätzen. Anhand dieser soll versucht werden ein grundsätzliches Verhalten der Verfahren basierend auf den anzuwendenden Datensatz vorher zu sagen. Für die ausgewählten Datensätze werden die ersten drei Schritte des Crisp DM folgend genauer betrachtet. Dabei soll der Grund für die Auswahl des jeweiligen Datensatzes erläutert werden und welchen Wert dieser für die Arbeit und der Zielerreichung darstellt.

Die Auswahl und Anwendung der Methode in der Modeling Phase ergibt sich durch die Zielsetzung der Arbeit. Evaluation und Anwendung wird im nächsten Kapitel Auswertung analysiert.

## 3.5 Business Understanding

#### 3.5.1.1 UCI Datensätze

Die Untersuchung einer großen Menge an UCI Datensätzen soll zu einem grundsätzlichen Verständnis der Leistung beider Verfahren führen. Dabei wird anhand einer kompakten

[69] Breiman 1996.
[70] Dua and Graff 2017.

Übersicht ersichtlich wie sich die Verfahren in Bezug auf Genauigkeit und Berechnungszeit verhalten.

#### 3.5.1.2 Verfahren spezifische und komplexe Datensätze

Es sollen Datensätze die künstlich mit einem geometrischen Muster generiert wurden untersucht werden. Ziel ist es zu analysieren, ob und wenn ja, in welchem Maße sich die geometrische Ausprägung der Daten auf die Leistung der Verfahren auswirkt.

### 3.5.2 Data Understanding

#### 3.5.2.1 UCI Datensätze

Die vorliegenden 70 Datensätze (vgl. Tabelle 3.1) unterteilen sich in 27 binären Klassen und 43 Mehrklassen Klassifikationsprobleme. Auch zeichnen sich die Datensätze durch eine hohe Varianz bei der Anzahl der Merkmale sowie der Instanzen aus.

Auffällig sind drei Sätze mit weniger als 35 Instanzen. Diese können zu unrealistischen Ergebnissen führen, da jede der Stichproben, die mit ersetzen aus diesen Daten entnommen wurde, aus nicht mehr als den $< 33$ unterschiedlichen Werten bestehen. Die Fälle zu mischen und einige von diesen nicht zu ziehen, würde nicht viel an der Fähigkeit ändern, etwas Neues über die zugrunde liegende Verteilung zu erfahren. Eine kleine Stichprobe ist also ein Problem für Bootstrap. Nicht nur die Bootstrap Eigenschaft des Random Forest Verfahren leidet unter einer kleinen Anzahl an Instanzen. Entscheidungsbäume werden trainiert, indem die Daten bedingt auf die Prädiktorvariablen aufgeteilt werden, um solche Unterstichproben zu finden, die die größte Unterscheidungskraft haben. Für die Berechnung des bedingten Mittelwertes aus den Stichproben (um kontinuierliche Werte in Regressionsbäumen oder bedingte Wahrscheinlichkeiten in Entscheidungsbäumen vorherzusagen), würden die Schlussfolgerungen nur auf diese wenigen Fälle gestützt werden. Die Teilstichproben, die verwendet werden, um die Entscheidungen zu treffen, wären also noch kleiner als die ursprünglichen Daten.

| Datensatz | Instanzen | Merkmale | Klassen | | | | | |
|---|---|---|---|---|---|---|---|---|
| abalone | 4177 | 8 | 3 | | lymphography | 148 | 18 | 4 |
| adult | 48842 | 14 | 2 | | magic | 19020 | 10 | 2 |
| annealing | 798 | 38 | 6 | | mammographic | 961 | 5 | 2 |
| arrhythmia | 452 | 262 | 13 | | miniboone | 130064 | 50 | 2 |
| audiology-std | 226 | 59 | 18 | | mushroom | 8124 | 21 | 2 |
| balloons | 16 | 4 | 2 | | nursery | 12960 | 8 | 5 |
| bank | 45211 | 17 | 2 | | oocMerl2F | 1022 | 25 | 3 |
| breast-cancer | 286 | 9 | 2 | | oocMerl4D | 1022 | 25 | 3 |
| bc-wisc | 699 | 9 | 2 | | oocTris2F | 912 | 25 | 2 |
| breast-tissue | 106 | 9 | 6 | | oocTris5B | 912 | 32 | 3 |
| car | 1728 | 6 | 4 | | page-blocks | 5473 | 10 | 5 |
| ctg-10classes | 2126 | 21 | 10 | | pima | 768 | 8 | 2 |
| ctg-3classes | 2126 | 21 | 3 | | planning | 182 | 12 | 2 |
| credit-approval | 690 | 15 | 2 | | post-operative | 90 | 8 | 3 |
| cylinder-bands | 512 | 35 | 2 | | primary-tumor | 330 | 17 | 15 |
| echocardiogram | 131 | 10 | 2 | | ringnorm | 7400 | 20 | 2 |
| ecoli | 336 | 7 | 8 | | seeds | 210 | 7 | 3 |
| energy-y1 | 768 | 8 | 3 | | semeion | 1593 | 256 | 10 |
| energy-y2 | 768 | 8 | 3 | | soybean | 307 | 35 | 18 |
| fertility | 100 | 9 | 2 | | spambase | 4601 | 57 | 2 |
| flags | 194 | 28 | 8 | | steel plates | 1941 | 27 | 7 |
| glass | 214 | 9 | 6 | | synthetic-control | 600 | 60 | 6 |
| haberman-survival | 306 | 3 | 2 | | teaching | 151 | 5 | 3 |
| hayes-roth | 132 | 3 | 3 | | thyroid | 3772 | 21 | 3 |
| hill-valley | 606 | 100 | 2 | | tic-tac-toe | 958 | 9 | 2 |
| horse-colic | 300 | 25 | 2 | | titanic | 2201 | 3 | 2 |
| ilpd-indian-liver | 583 | 9 | 2 | | twonorm | 7400 | 20 | 2 |
| image-segmentation | 210 | 19 | 7 | | vc-2classes | 310 | 6 | 2 |
| ionosphere | 351 | 33 | 2 | | vc-3classes | 310 | 6 | 3 |
| iris | 150 | 4 | 3 | | wall-following | 5456 | 24 | 4 |
| led-display | 1000 | 7 | 10 | | waveform | 5000 | 21 | 3 |
| lenses | 24 | 4 | 3 | | waveform-noise | 5000 | 40 | 3 |
| letter | 20000 | 16 | 26 | | wine | 179 | 13 | 3 |
| libras | 360 | 90 | 15 | | yeast | 1484 | 8 | 10 |
| lung cancer | 32 | 56 | 3 | | zoo | 101 | 16 | 7 |

*Tabelle 3.1 Übersicht Auswahl UCI Datensätze*

Ein ähnliches Problem kann bei einer zu kleinen Anzahl an Featuren auftreten. Random Forest verwendet Bootstrapping-Samples in jeder Phase des Ensembles. Der Haupt-unterschied zu Bagging besteht darin, dass es verschiedene zufällige Teilmengen von m Features (Prädiktoren) verwendet, anstatt alle p-Features an jedem Baumknoten. Die besten Partitionierungsregeln für Knoten werden nur aus dem Kandidaten-Feature-Set ausgewählt. Es wird somit immer nur eine Teilmenge von Features verwendet, dies führt im Allgemeinen zu vielen verschiedenen und unkorrelierte Bäumen. Die Mittelung der unkorrelierten Bäume führt zu einer deutlichen Reduzierung der Klassifikationsvarianz und somit zu einer Steigerung

der Klassifikation Richtigkeit.[71] Breiman[72] schlug vor die Größe des Kandidaten-Feature-Sets an jedem Knoten als $m \approx log2(p+1)$ festzulegen. In folgenden Forschungen im Zusammenhang mit Random Forest hat sich die Standardgröße des Kandidaten-Feature-Sets als $m \approx \sqrt{p}$ bei Klassifikationsproblemen und $m \approx p/3$ bei Regressionsproblemen etabliert.[73]

Bei den ausgewählten UCI Datensätzen weisen 65 Datensätze eine Unausgewogenheit bei mindestens einem Feature auf. Die Eigenschaft des Bootstrapping kann auch bei unausgewogenen Datensätzen zu Problemen führen, da es eine Stichprobe des Trainingssatzes verwendet, um jeden Baum zu bilden. Bei unausgewogenen Daten steigt die Wahrscheinlichkeit von Bootstrap-Stichproben, die wenige oder keine der Minderheiten-klassen enthalten, merklich an, was dazu führt, dass das berechnete Modell kein aussagekräftiger Prädiktor für die Minderheitenklasse ist. Datensätze, die eine solche Unausgewogenheit aufzeigen, können durch verschiedene Methoden angepasst werden. Einfache Methoden erweitern künstlich die unterrepräsentierten oder reduzieren die überrepräsentierten Klassen. Hierbei wird direkt Einfluss auf die Daten genommen was zu einer Verfälschung der Ergebnisse führen kann. Um dies zu vermeiden, gibt es einige Möglichkeiten die Unausgewogenheit direkt durch das Klassifikationsverfahren auszugleichen. Neben kostensensiblen bzw. kostengewichteten Optimierungen und balancierten Random Forest Verfahren kann auch beim bagging eine Oversampling bzw. Undersampling zu der gewünschten Anpassung der Daten führen. Ziel der vorliegenden Arbeit ist nicht die optimalen Werte mit beiden Verfahren zu berechnen, sondern diese anhand ihrer Leistung zu bewerten. Es wird daher von einer Anpassung der Daten abgesehen, um die Ergebnisse nicht durch die Optimierungen zu verfälschen.

Alle Datensätze wurden zudem auf fehlende und falsche Werte untersucht. Da keine solche Werte identifiziert werden konnten, wurde dieser Schritt bei der Datenbereinigung nicht weiter beachtet. Bei der genaueren Untersuchung der Daten viel auf, dass bei 40 Datensätzen mindestens ein Feature in Form eines nominalen Datentyps vorliegt und im nächsten Schritt angepasst werden muss. Dies ist vor allem bei dem zu klassifizierenden Feature oft der Fall.

Anschließend wird für jeden Datensatz die Korrelation und der Informationsgewinn der einzelnen Feature berechnet. Dies ermöglicht eine Reduktion der Dimensionen der

[71] Breiman 1996.
[72] Breiman 2001.
[73] Han and Kim 2019.

vorliegenden Datensätze und damit eine genauere und schnellere Klassifikation. Feature mit sehr geringem Informationsgewinn (<0,01) werden für den Schritt der Datenaufbereitung zum Löschen markiert.

### 3.5.2.2 Verfahren spezifische und komplexe Datensätze

Für die Untersuchung der Auswirkungen der Datenstruktur auf die Performanz wurden die generierten sechs Datensätze genauer untersucht. Zwei der Datensätze zeigen eine Anordnung der Datenpunkte auf, die für jeweils eines der Verfahren bei der Split Berechnung vorteilhaft sein sollte (vgl. Abbildung 3.3 und 3.4). Die verbleibenden vier Datensätze weisen eine geometrische Struktur, mit einer hohen Anzahl an Instanzen auf. Bei drei dieser Datensätze ist ein geringes Rauschen und einer erkennbaren Trennung zu beobachten (vgl. Abbildung 3.5-3.7), wodurch beide Verfahren eine hohe Genauigkeit erzielen können. Hingegen ist bei einem Datensatz mit geometrischer Struktur ein gewisser Grad an Rauschen zu beobachten (vgl. Abbildung 3.8). Anzunehmen ist, dass dies zu einem starken Fall der Genauigkeit, im Vergleich zu den vorherigen Datensätzen, führen sollte. Spannend wird hier zu beobachten, welches der zwei Verfahren mit dieser Ausprägung besser umgehen kann.

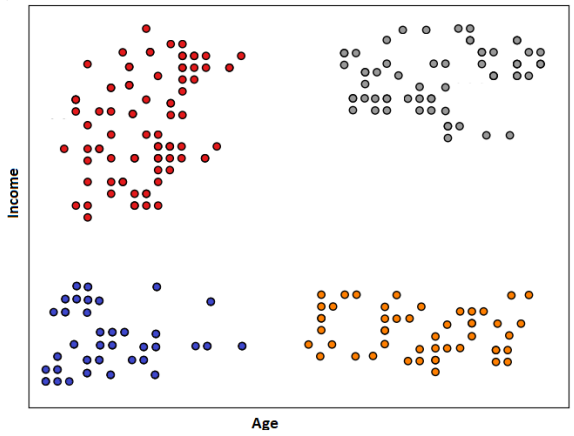



*Abbildung 3.4 Vorteil horizontal/vertikal Split*


*Abbildung 3.5 Vorteil schräger Split*

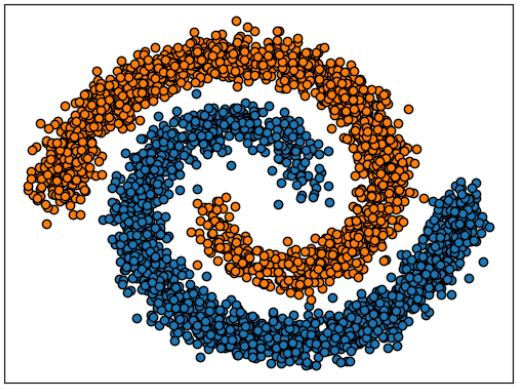


*Abbildung 3.6 geom. Struktur – Spirale*

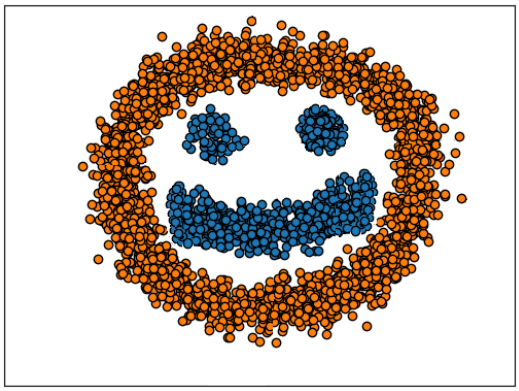


*Abbildung 3.7 geom. Struktur - Smiley*

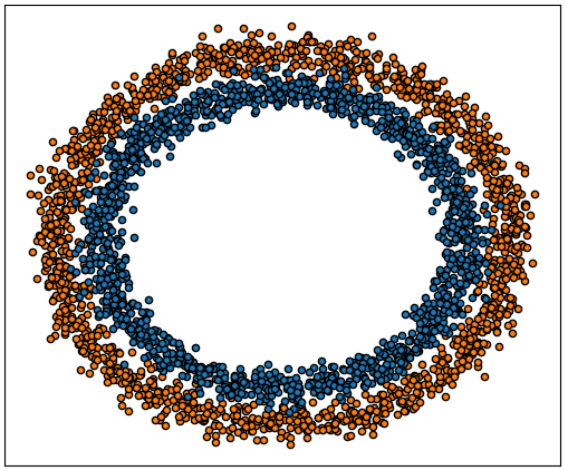


*Abbildung 3.8 geom. Struktur – Ringe geringem Rauschen*

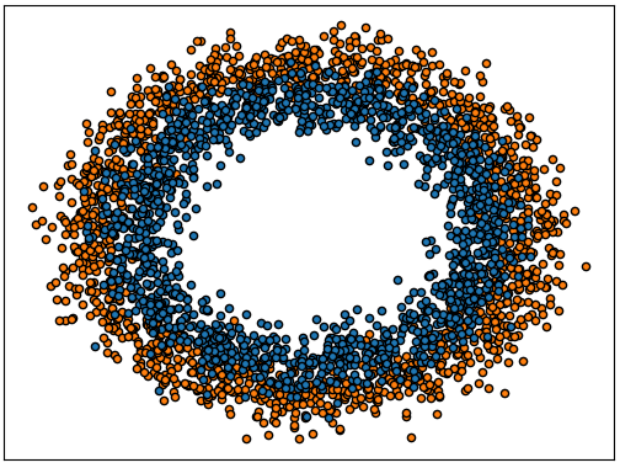


*Abbildung 3.9 geom. Struktur – Ringe mittlerem Rauschen*

### 3.5.3 Data Preparation

Mögliche Aufgaben und Schritte wurden in der Abbildung 3.9 skizziert. Im Rahmen der vorliegen Arbeit mussten nicht alle Teilaufgaben durchgeführt werden. Im vorherigen Schritt wurde bereits festgestellt, dass keine fehlenden Werte, Duplikate und Inkonsistenzen vorliegen, somit kann der Schritt der Bereinigung übersprungen werden. Die Datensätze konnten direkt in die Pythonskripte importiert werden, einer weiteren Bearbeitung für die Integration war somit nicht nötig. Es wurde zudem bereits angemerkt, dass einige Features in Form von Nominalen Daten vorliegen, diese wurden in numerische Daten überführt. Für die Testreihe in R wurden dem Feature für die Klassifikation ein entsprechendes Label beigefügt. Dieses ist nötig, um dem dort implementierten Verfahren zu signalisieren, dass es sich um eine Klassifikation handelt und nach welchem Feature klassifiziert werden soll. Wie in Kapitel 3.1 beschrieben, wurden für alle Datensätze Korrelationsanalyse und Berechnung des Informationsgewinn der Features angefertigt. Anhand der dort erhaltenen Ergebnisse wurde die Selektion der Features für die Verfahren optimiert.

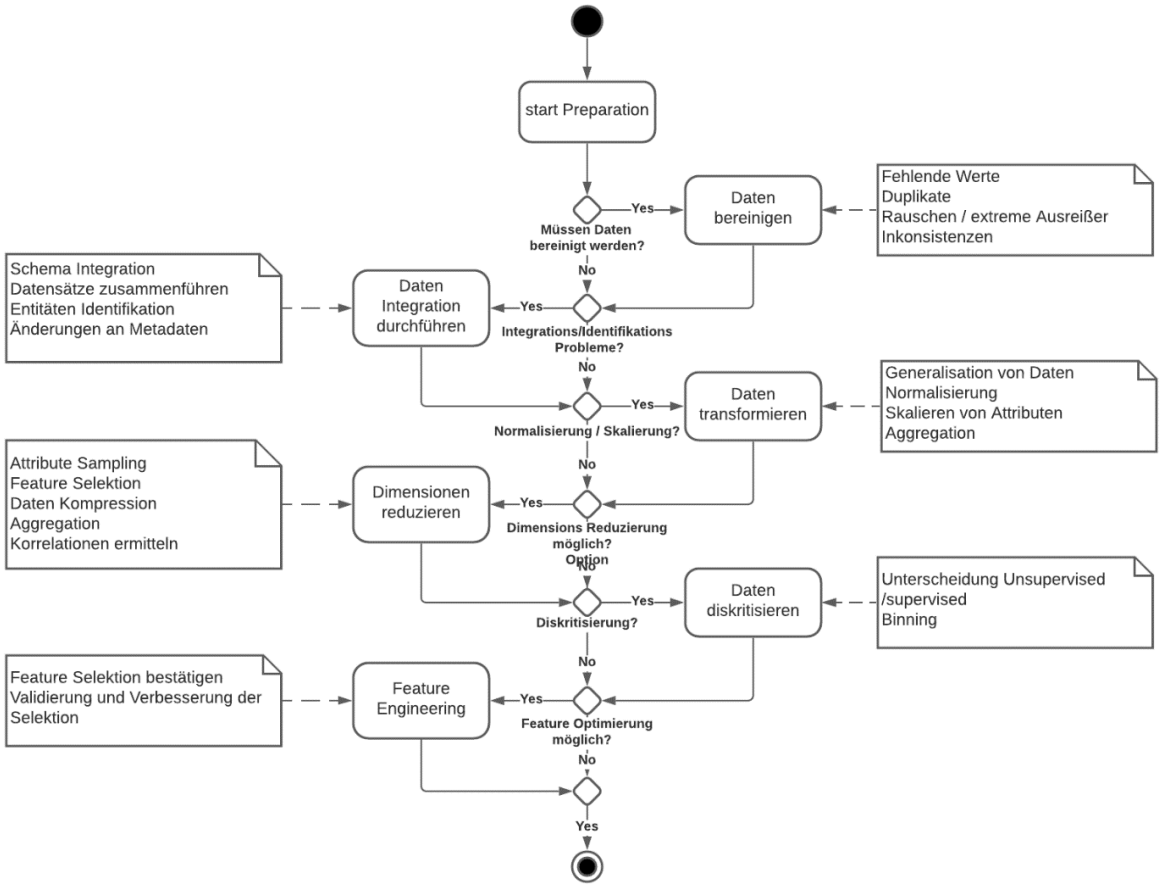



*Abbildung 3.10 Aktivitätendiagramm für die Phase der Data Preparation*

## 3.6 Einflussfaktoren

Die Kontrolle von Variablen macht ein Experiment im traditionellen Sinne zu einem großen Teil wissenschaftlich. Zwei Kategorien von Variablen, die kontrolliert werden müssen, sind interne Variablen und externe Variablen. Interne Variablen bestehen typischerweise aus den Variablen, die manipuliert und gemessen werden. Externe Variablen sind Faktoren, die außerhalb des Rahmens des Experiments liegen. Für den in Kapitel 3.1 skizzierten Ablauf wurden mehrere Faktoren identifiziert, die zu einer Änderung oder im schlimmsten Fall zu einer Verfälschung der Daten führen können. Im Folgenden wurden ausgiebige Testreihen durchgeführt, um für die Faktoren zu prüfen, inwieweit sich diese auf die Messwerte auswirken.

### 3.6.1 Programmiersprachen

Wie bereits in Kapitel 3.2 festgelegt wurde die Implementierung in der Programmiersprache Python realisiert. Alternativen in Java und R sind denkbar. Es wurde daher Stichprobenartig überprüft, ob es unter Verwendung von Implementierung in anderen Programmiersprachen

zu größeren Schwankungen in den Qualitätskriterien kommt. Realisierungen in Java und R zeigten einen großen Speicherbedarf. Größere Datensätze beanspruchten mit unter 10GB an Arbeitsspeicher. Eine vergleichbare Implementation in Python benutzte hingegen kaum Arbeitsspeicher wohingegen eine deutliche CPU-Belastung vor lag.

Alternativ konnten nur minimale Schwankungen bei den Qualitätskriterien beobachtet werden. Zu größeren Fluktuationen kam es überwiegend bei der Berechnungszeit. Wobei keine signifikanten Änderungen im Verhältnis der Ergebnisse der einzelnen Verfahren zueinander auftraten. Folglich lässt sich die Implementierung unter den getesteten Bedingungen als Einflussfaktor ausschließen.

### 3.6.2 Parallelität

Hochparallele Algorithmen zur Konstruktion von Entscheidungsbäumen für Klassifikationsprobleme sind wünschenswert, um große Datensätze in angemessener Zeit zu bearbeiten. Algorithmen zum Erstellen von Entscheidungsbäumen weisen eine natürliche Parallelität auf, sind jedoch aufgrund der inhärenten dynamischen Natur der Berechnung schwierig zu parallelisieren. Versuche den gängigen Random Forest Algorithmus mit CPU/GPU zu parallelisieren gestalteten sich schwierig. Erste Implementationen erzeugten einen deutlichen Overhead, sodass diese Betrachtung nicht weiterverfolgt wurde. Um eine ergiebige Parallelität zu gewährleisten, müsste auf einen Random Forest basierten parallel optimierten Algorithmus zurückgegriffen werden. Diese Betrachtung ist nicht im Umfang der vorliegenden Arbeit und wurde daher nicht weiterverfolgt.

### 3.6.3 Hardware

Für alle Berechnungen wurde, die in Kapitel 3.2 spezifizierte Hardware verwendet. Es wurde eine Testreihe durchgeführt, ob es zu Abweichungen der Ergebnisse kommt, wenn unterschiedliche Rechenkerne zur Berechnung verwendet werden oder anderweitig Last auf das System ausgeübt wird. Hierzu wurde Python auf unterschiedliche Rechenkerne eingeschränkt. Zudem wurden CPU und RAM Benchmarks betrieben während Python Random Forest Modelle berechnete. Auch hier konnten keine Abweichungen der Ergebnisse beobachtet werden, sodass weitere Einschränkungen bei der Versuchsdurchführung nicht weiter berücksichtigt wurden.

# 4 Versuchsdurchführung am Beispiel „Adult“

Das folgende Kapitel dient einer tieferen Betrachtung der Datenaufbereitung. Der Datensatz Adult, des UCI Archives für maschinelles Lernen, wird hierfür als Beispiel herangezogen. Es wird sich an das in Kapitel 3 vorgestellte Vorgehen gehalten, um dieses besser zu illustrieren. Die hier vorgestellte Aufbereitung ist größtenteils analog zu allen Datensätzen. Natürlich bestehen Unterschiede zwischen den Datensätzen, diese können zu leichten Abweichungen in der Durchführung des Vorgehens führen. Die in Kapitel 3.4.3 vorgestellten Schritte für die Datenaufbereitung werden für alle Datensätze betrachtet. Dabei hängt die Durchführung stehts von der Ausprägung der Daten und der Qualität des Datensatzes ab. Das Kapitel soll zudem einen Eindruck in den Umfang und die Wichtigkeit der Datenaufbereitung bieten.

## 4.1 Business Understanding

Die ausgeprägte Ungleichheit von Vermögen und Einkommen ist insbesondere in den Vereinigten Staaten ein großes Problem. Die Wahrscheinlichkeit eines Rückgangs der Armut ist ein triftiger Grund, die weltweit zunehmende wirtschaftliche Ungleichheit zu verringern. Das Prinzip der universellen moralischen Gleichheit sichert eine nachhaltige Entwicklung und verbessert die wirtschaftliche Stabilität einer Nation. Regierungen in verschiedenen Ländern haben ihr Bestes gegeben, um dieses Problem anzugehen und eine optimale Lösung bereitzustellen. Es wird versucht unter Einsatz von maschinellem Lernen und Data-Mining-Techniken eine Lösung aufzuzeigen. Auf der Grundlage von Volkszählungsdaten soll eine Vorhersage über das Einkommen der Personen getroffen werden.

## 4.2 Data Understanding

Bei der Untersuchung des Datensatzes vielen folgende Eigenschaften auf:

- Anzahl der Attribute beträgt 14
- Attribute liegen in kategorialer, diskreter, kontinuierlicher und numerischer Form vor
- Es gibt fehlende Werte bei dem Attribut *workclass*, *occupation* und *native-country*
- Fehlende Werte liegen bei 2399 Instanzen von 48842 vor
- Die Klassifikation von Einkommen <=50k und >50k zeigt eine deutliche Unausgewogenheit mit 24720 zu 7841 Instanzen auf
- Binäre Klassifikation <=50k und >50k

## 4.3 Data Preparation

### 4.3.1 Datenbereinigung

Bei dem Datenverständnis vielen 2399 fehlende Werte und eine Unausgewogenheit der Daten auf. Im Rahmen der Bereinigung muss entschieden werden, wie mit diesen Daten umgegangen werden soll. In Bezug zu den Daten sind die Methoden Reparieren, künstliche Beschriftung anhand anderer Merkmale oder Löschen der Instanz denkbar. Bei dem Datensatz liegen 48842 Instanzen vor, das Löschen von falschen/fehlenden Daten kann daher in Betracht gezogen werden, da dieses das Ergebnis nur minimal beeinflussen würde. Bei genauerer Betrachtung viel zudem auf, dass sich nur 329 aus den 2399 Instanzen mit fehlenden Werten in der unterrepräsentierten Klasse befinden. Das Löschen der Werte würde daher dieses Verhalten der Daten nicht weiter verschlimmern, sondern zu einer kleinen Anpassung der Ausgeglichenheit führen. Ansonsten liegen keine weiteren extremen Ausreißer, Duplikate oder Inkonsistenzen vor, die weiter behandelt werden müssten.

### 4.3.2 Daten Integration

Die Daten liegen in zwei Datendateien vor. Eine Datei enthält die reinen Daten, die zweite die nötigen Beschriftungen der Spalten. Die verwendete Python Implementation konnte beide Dateien direkt einlesen und verwenden. Es muss daher keine weitere Anpassung vorgenommen werden.

### 4.3.3 Daten Transformation

Bei der Betrachtung der Daten viel auf, dass Features in unterschiedlichen Formen, z.B. kategorial, vorliegen. Für die Verwendung der, in dieser Arbeit betrachteten Klassifikationsverfahren und der Art der Implementierung müssen diese in numerische Werte überführt werden. Die Anzahl der Kategorien viel für diesen Datensatz klein aus, wodurch eine direkte Transformation angewendet werden kann. Die Transformation hat dabei keinen weiteren Einfluss auf das Ergebnis. Bei einer großen Vielfalt an Kategorien wäre eine Kombination von einzelnen Ebenen oder hashing-Verfahren ratsam. Eine Normalisierung, z.B. anhand der Z-score oder Min-Max Methode, sowie eine Generalisierung der Daten ist in diesem Fall nicht nötig.

### 4.3.4 Daten Reduzierung

Datensätze können eine Vielzahl an Features aufweisen. Die Komplexität der Berechnung und die Aussagekraft des Ergebnisses wird dabei stark von der Anzahl beeinflusst. Es wird daher versucht die Feature Dimensionen so stark wie möglich einzuschränken, ohne das Ergebnis groß zu beeinflussen. Dies kann über verschiedene Methoden erreicht werden. Im Rahmen der Arbeit wurde der Informationsgewinn jedes Features betrachtet und die Korrelation zueinander.

Die kontinuierliche Variable *fnlwgt* stellt das Endgewicht dar, d.h. die Anzahl der Einheiten in der Zielpopulation, die die vorliegende Einheit repräsentiert. Die Variable *education_num* steht für die Gesamtzahl der Bildungsjahre, die eine kontinuierliche Darstellung der diskreten Variablen *education* darstellt. Die variable *relationship* repräsentiert die Rolle der entsprechenden Einheit in der Familie.

Der Informationsgewinn der Variable *fnlwgt* beträgt 0 und hat somit keinen Einfluss auf die Zielvariable. Die Gesamtzahl der Bildungsjahre kann durch den höchsten abgeschlossenen Bildungsabschluss repräsentiert werden. Die Rolle in der Familie kann anhand des Geschlechts und des Familienstands beurteilt werden. Somit können die folgenden 3 Variablen *education*, *relationship* und *fnlwgt* gelöscht werden.

## 4.4 Modeling und Auswertung

Basierend auf den angepassten Datensatz können die Klassifikationsverfahren angewendet werden. Anschließend werden anhand eines ersten Klassifikationsmodells die Hyperparameter angepasst. Mit Hilfe einer Rastersuche werden verschiedene Kombinationen und Parameterwerte getestet. Die Auswirkungen einer Anpassung von einzelnen Hyperparameter auf die Genauigkeit des Modells wurden in Abbildung 4.1 skizziert. Zu erkennen ist das durch eine genaue Auswahl der Hyperparameter das Modell um mehr als 1,5% verbessert werden konnte, dies zeigt auch wie wichtig es ist diese Optimierungen spezifisch auf jeden Datensatz durchzuführen.

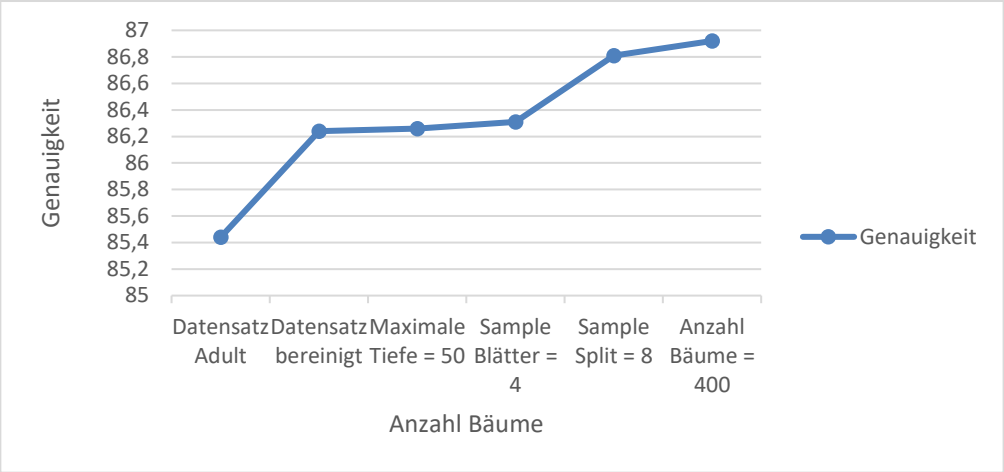



*Abbildung 4.1 Auswirkung Optimierung Hyperparameter*

# 5 Auswertung

Bevor die Klassifikationsergebnisse präsentiert werden, wird nochmal genauer der Einfluss des Hyperparameters $n$ = Anzahl Entscheidungsbäume untersucht. Abb. 5.1 zeigt die Genauigkeit und Abb. 5.2 die Berechnungszeit für den *miniboone* Datensatz für verschiedene Werte von n ∈ {100, 500, 1000, 1500, 2000}. Es lässt sich erkennen, dass mit zunehmender Anzahl an Bäumen, für die Modellerstellung des Random Forest (RF) und des Oblique Forest (OF), die Genauigkeit beschränkt monoton zunimmt. Erhöhung der Bäume über den Punkt der Sättigung hinaus hat entweder keine Auswirkungen oder beeinträchtigt sogar leicht die Leistung. Im Vergleich zeigt sich bei der Berechnungszeit eine stetig monotone Steigung. Wobei die gemessene Steigung abflacht und bei einem Wert $n > 1700$ für die Anzahl der Bäume eine lineare Form annimmt. Aus den Messwerten geht hervor, dass für $n = 1000$ die beste Genauigkeit im Bezug zu der Berechnungszeit vorliegt. Im Vergleich der Leistung beider Verfahren zeigt sich, dass der Oblique Forest deutlich stabiler auf Änderungen an dem Hyperparameter $n$ reagiert und bereits bei kleinen Werten eine gute Genauigkeit liefert. Der Random Forest hingegen zeigt sich für diesen Datensatz anfangs schwach nähert sich, mit höherer Rechenintensität an das Ergebnis des Oblique Forest an.

Im Rahmen der Optimierung der Hyperparameter für einen optimalen Wert der Anzahl der Bäume ließ sich beobachten, dass 30 Datensätze im Bereich 400 bis 500, 6 bei 700, 3 bei 800 und 31 im Bereich von 900 bis 1000 liegen. Anhand dieser Ergebnisse wurde für einen Teil der Datensätze im Bereich $n = 1000$ eine erweiterte Auswertung durchgeführt, um die dortige Genauigkeit genauer zu untersuchen (vgl. Tabelle 5.1). Es konnten keine signifikanten

Unterschiede bei der Qualität beobachtet werden, wobei es zu einer minimalen Verbesserung kommt.

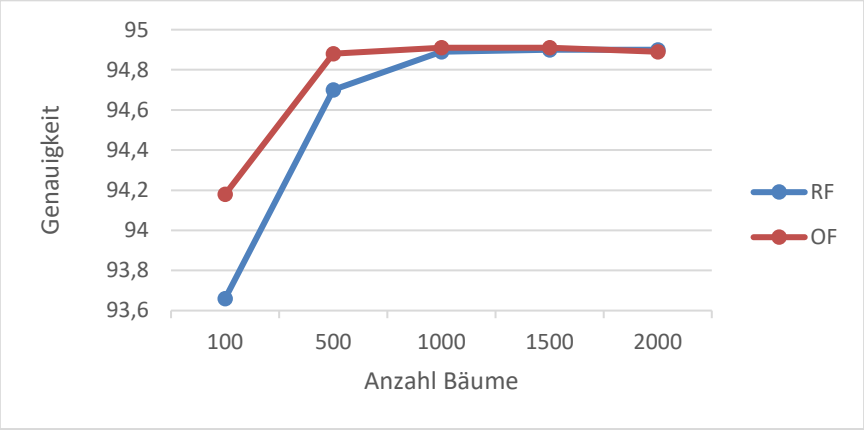



*Abbildung.5.1 Genauigkeit basierend auf n-Bäumen*


*Abbildung 5.2 Berechnungszeit basierend auf n-Bäumen*

| Datensatz | RF n=500 | OF n=500 | Ber. RF 500 | Ber. OF 500 | RF n=1000 | OF n=1000 | Ber. RF 1000 | Ber. OF 1000 |
|---|---|---|---|---|---|---|---|---|
| adult | 86,43 | 86,12 | 9,390 | 12,655 | 86,46 | 86,13 | 19,518 | 23,926 |
| balloons | 52,01 | 50,37 | 0,478 | 0,802 | 52,1 | 50,39 | 1 | 2,221 |
| ctg-10classes | 99,62 | 99,83 | 1,142 | 1,985 | 99,63 | 99,85 | 2,288 | 3,393 |
| credit-approval | 90,12 | 89,29 | 0,792 | 1,174 | 90,16 | 89,31 | 1,865 | 3,174 |
| gender by name | 61,407 | 62,71 | 47,419 | 52,156 | 61,415 | 62,81 | 95,281 | 101,336 |
| hill-valley | 59,86 | 58,79 | 1,127 | 1,475 | 59,86 | 58,84 | 2,719 | 3,765 |
| iris | 95,55 | 96,02 | 0,521 | 1,038 | 95,6 | 96,02 | 0,935 | 2,220 |
| lenses | 88,86 | 85,76 | 0,495 | 0,198 | 88,88 | 85,75 | 0,988 | 1,528 |
| letter | 96,38 | 96,28 | 8,398 | 9,372 | 96,4 | 96,3 | 16,199 | 17,764 |
| libras | 82,22 | 82,88 | 1,070 | 1,226 | 82,43 | 82,85 | 1,771 | 3,244 |
| miniboone | 94,7 | 94,88 | 241,658 | 313,812 | 94,89 | 94,91 | 483,410 | 556,191 |
| waveform | 86,34 | 85,52 | 4,091 | 4,829 | 86,4 | 85,58 | 7,44 | 8,819 |

*Tabelle 5.1 Selektion UCI Datensätze Genauigkeit u. Berechnungszeit Bäume n=1000*

Obwohl das No-Free-Lunch-Theorem besagt, dass kein einzelner Klassifikator immer der Beste ist, ist es intuitiv, den robustesten Klassifikator zu implementieren, der in den meisten Datensätzen besser verallgemeinert. Im Rahmen dieser Arbeit stehen für die Betrachtung nur zwei Verfahren zur Verfügung. Der Fokus liegt damit darauf Unterschiede im Bezug zu der Leistung dieser Verfahren aufzuzeigen.

Anhand der Ergebnisse der UCI Datensätze lässt sich schnell eine große Ähnlichkeit der Messwerte erkennen. Unter Verwendung, der in Kapitel 3.1.1 erläuterten Hyperparameter wird ersichtlich, dass der Random Forest (RF Optimiert) bei 30 Datensätzen zu einem genaueren Ergebnis führt, wohingegen der Oblique Forest (OF Optimiert) bei 31 besser abschneidet (vgl. Abbildung 5.4). Bei den verbleibenden 9 Datensätzen kommt es zu keinem deutlichen Unterschied in den Ergebnissen, für die Betrachtung der Genauigkeit (vgl. Tabelle 5.2). Dies stellt eine deutliche Besserung zu den Messwerten von der Testreihe ohne optimierte Hyperparameter dar. Hier konnte sich der Random Forest (RF Standard) nur in 26 Datensätzen behaupten, wohingegen der Oblique Forest (OF Standard) bei den verbleibenden 44 zu besserer Werten führte (vgl. Abbildung 5.3).

| Datensatz | RF S | OF S | RF O | OF O | | RF S | OF S | RF O | OF O |
|---|---|---|---|---|---|---|---|---|---|
| abalone | 65,84 | 65,82 | 66,1 | 65,99 | lymphography | 85,74 | 85,08 | 87,01 | 85,78 |
| adult | 85,88 | 85,17 | 86,43 | 86,12 | magic | 83,87 | 85,72 | 85,97 | 86,44 |
| annealing | 77,81 | 77,97 | 78,06 | 78,05 | mammographic | 82,59 | 82,80 | 84,53 | 82,84 |
| arrhythmia | 82,01 | 81,22 | 82,31 | 82,11 | miniboone | 93,66 | 94,18 | 94,7 | 94,88 |
| audiology-std | 82,87 | 83,62 | 83,4 | 83,93 | mushroom | 100,00 | 99,36 | 100 | 100 |
| balloons | 50,76 | 49,70 | 52,01 | 50,37 | nursery | 99,53 | 99,08 | 99,71 | 99,66 |
| bank | 86,64 | 87,09 | 87,75 | 87,65 | oocMerl2F | 92,65 | 93,06 | 93,03 | 93,13 |
| breast-cancer | 78,14 | 78,29 | 79,61 | 79,26 | oocMerl4D | 77,13 | 80,90 | 78,36 | 81,89 |
| bc-wisc | 97,83 | 97,17 | 98,01 | 98,01 | oocTris2F | 78,96 | 83,35 | 80,67 | 83,46 |
| breast-tissue | 69,03 | 69,90 | 69,45 | 70,37 | oocTris5B | 91,78 | 93,17 | 92,46 | 93,3 |
| car | 98,15 | 98,54 | 98,71 | 98,96 | page-blocks | 98,05 | 97,81 | 98,54 | 98,64 |
| ctg-10classes | 99,44 | 99,03 | 99,62 | 99,83 | pima | 75,86 | 77,06 | 77,25 | 77,38 |
| ctg-3classes | 100,00 | 99,04 | 100 | 100 | planning | 72,73 | 73,08 | 74,44 | 73,33 |
| credit-approval | 89,21 | 88,56 | 90,12 | 89,29 | post-operative | 73,01 | 72,86 | 73,59 | 73,59 |
| cylinder-bands | 78,64 | 80,45 | 80,86 | 80,86 | primary-tumor | 56,24 | 56,45 | 57,91 | 57 |
| echocardiogram | 80,73 | 83,17 | 81,99 | 83,64 | ringnorm | 97,01 | 92,63 | 97,45 | 93,03 |
| ecoli | 83,94 | 85,67 | 85,93 | 85,93 | seeds | 93,20 | 94,61 | 94,18 | 94,83 |
| energy-y1 | 94,13 | 94,83 | 94,84 | 95,49 | semeion | 93,12 | 93,63 | 94,62 | 94,3 |
| energy-y2 | 87,89 | 87,25 | 88,67 | 88,02 | soybean | 39,64 | 37,25 | 40,06 | 37,61 |
| fertility | 90,32 | 91,29 | 91,23 | 92,23 | spambase | 94,47 | 95,51 | 95,22 | 95,61 |
| flags | 63,53 | 63,73 | 65,78 | 64,43 | steel plates | 78,36 | 78,33 | 80,11 | 78,98 |
| glass | 75,97 | 74,36 | 78,64 | 74,89 | synthetic-control | 97,99 | 98,55 | 98,48 | 98,87 |
| haberman-survival | 74,11 | 75,38 | 76,78 | 75,8 | teaching | 63,12 | 63,19 | 63,6 | 64 |
| hayes-roth | 86,78 | 88,99 | 89,8 | 89,8 | thyroid | 98,64 | 98,43 | 98,87 | 98,87 |
| hill-valley | 58,73 | 58,26 | 59,86 | 58,79 | tic-tac-toe | 98,78 | 97,71 | 98,93 | 97,77 |
| horse-colic | 63,98 | 63,46 | 66,54 | 63,94 | titanic | 76,34 | 77,77 | 77,99 | 78,46 |
| ilpd-indian-liver | 68,46 | 72,71 | 70,55 | 73,12 | twonorm | 96,89 | 97,19 | 97,58 | 97,85 |
| image-segmentation | 13,34 | 13,90 | 13,67 | 13,9 | vc-2classes | 82,67 | 83,08 | 83,14 | 83,8 |
| ionosphere | 91,17 | 93,45 | 91,9 | 93,72 | vc-3classes | 84,26 | 85,33 | 85,12 | 85,69 |
| iris | 94,76 | 95,35 | 95,55 | 96,02 | wall-following | 99,57 | 99,28 | 99,79 | 99,76 |
| led-display | 64,28 | 68,62 | 65,64 | 69,43 | waveform | 85,24 | 85,25 | 86,34 | 85,52 |
| lenses | 84,23 | 85,17 | 88,86 | 85,76 | waveform-noise | 85,28 | 84,92 | 85,28 | 85,89 |
| letter | 96,04 | 95,82 | 96,38 | 96,28 | wine | 98,36 | 98,40 | 98,73 | 98,83 |
| libras | 78,77 | 81,95 | 82,22 | 82,88 | yeast | 63,45 | 63,48 | 64 | 63,76 |
| lung cancer | 57,26 | 54,05 | 58 | 54,74 | zoo | 99,09 | 98,72 | 99,44 | 99,44 |

*Tabelle 5.2 Auszug UCI Datensätze Genauigkeit und Berechnungszeit für Bäume n=500*

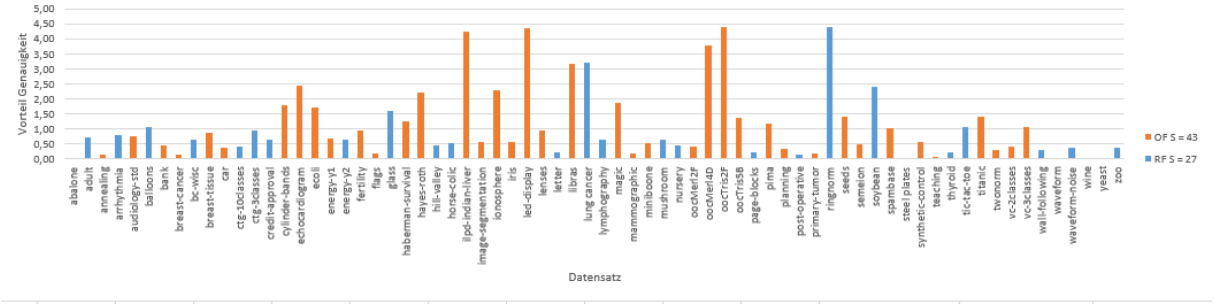



*Abbildung 5.3 Vergleich der Genauigkeit für die Standardparameter*

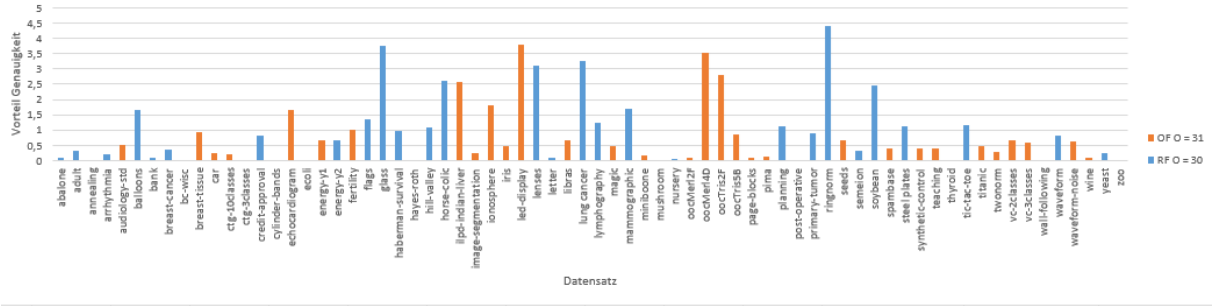



*Abbildung 5.4 Vergleich der Genauigkeit für die optimierten Parameter*

Für die 31 Datensätze mit Vorteil Random Forest liegt die Verbesserung gegenüber des Oblique Forest bei Durchschnittlich 1,11% mit einem minimalen Vorteil von 0,01% und einem Maximum von bis zu 4.42%. Vergleichsweise sieht dies beim Oblique Forest ähnlich aus mit einer durchschnittlichen Verbesserung von 0.92%, einem Minimum von 0.1% und einem Maximum von 3,79%. Die vollen Ergebnisse sind im Anhang einsehbar.

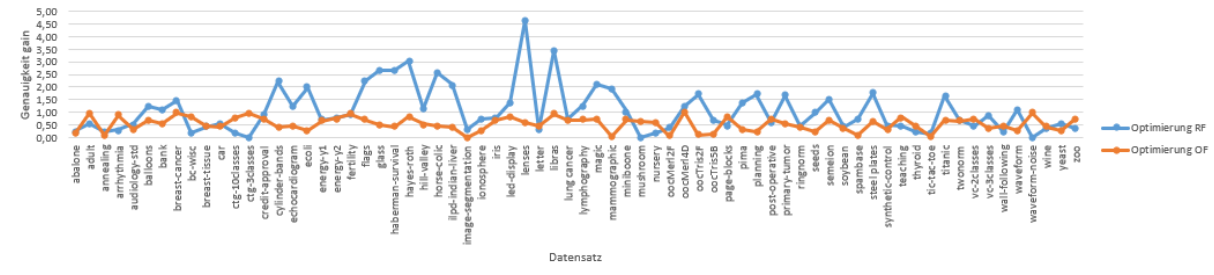



*Abbildung 5.5 Gewonnene Genauigkeit durch Parameteroptimierung*

Ein deutlicher Unterschied lässt sich vor allem in der Berechnungszeit beobachten. Random Forest führt bei 62 Datensätzen schneller zu einem Ergebnis. Dieser Unterschied mag anhand der vorliegenden Daten nicht so wichtig erscheinen, bei größeren Datensätzen oder einer großen Anzahl an Feature kann die Rechenzeit, zur Berechnung eines Klassifikators, schnell ausschlagkräftig werden. Wie am Datensatz *miniboone* ersichtlich, kann die Berechnung, bereits bei $n = 500$ Bäumen einen Unterschied von 72,154 Sekunden aufweisen. Bei $n = \sim 1000$, welcher bei einzelnen Datensätzen zu Verbesserung führte, betrug der Unterschied bereits fast vier Minuten, was einer Steigerung von 48,12% entspricht.

Zu beachten ist jedoch, dass die Berechnung von Hyperparametern mit Hilfe der Rastersuche eine ungemeine Rechenzeit beansprucht. Anhand der vorherigen Ergebnisse ersichtlich, schneidet das Oblique Forest verfahren bereits mit Standardparametern bzw. gut geschätzten Parametern deutlich besser ab. In Abb. 5.6 wurde die Berechnungszeit zur Ermittlung von Hyperparametern aufgezeigt. Die Rechenzeiten wurden hierbei in Stunden angegeben. Ersichtlich wird, dass selbst für kleinere Datensätze mit einer Größenordnung von 1-4 Stunden gerechnet werden muss. Für größere Datensätze, wie z.B. dem *miniboone* können Zeiten für die Berechnungen mit bis zu 37 Stunden auftreten.

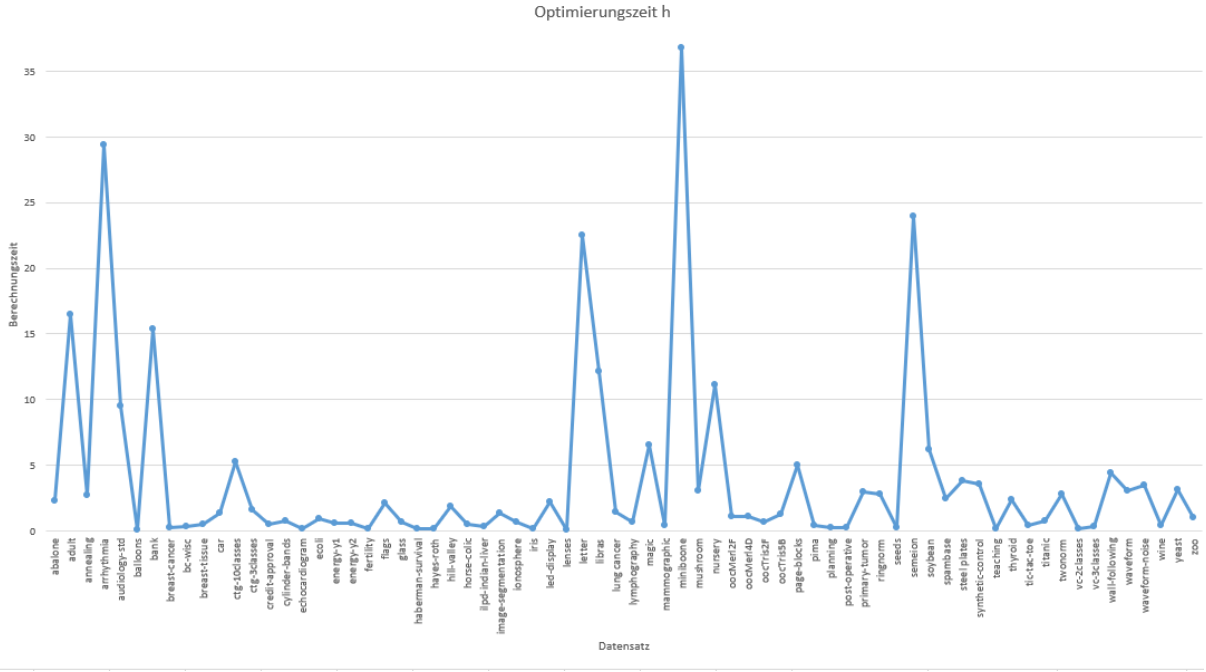



*Abbildung 5.6 Berechnungszeit zur Findung besserer Hyperparameter*

Bei der Betrachtung der künstlich generierten Datensätzen mit geometrischer Form, wurde für die Auswertung mehr auf die Tiefe der generierten Bäume geachtet und weniger auf die Bearbeitungszeit. Am ersten Beispiel für den Datensatz, der einen horizontalen Split bevorzugen sollte, ist dieses Verhalten klar erkennbar (vgl. Abbildung 5.7 und 5.8). Folglich lässt sich das Ergebnis leichter an dem visualisierten Modell nachvollziehen. Dasselbe Verhalten ist für den Datensatz für den schrägen Split zu beobachten. Es lässt sich dadurch bestätigen, dass die geometrische Ausrichtung der Daten bei der Wahl des zu verwendeten Verfahrens zu beachten ist. Ein gutes Indiz hierfür liefert die erreichte Tiefe des Baums, bereits anhand der Trainingsdaten kann diese leicht ermittelt werden. Für die Entscheidung, im Schritt des Datenverständnis, kann die Baumtiefe als ein Grund für die Verwendung eines der Verfahren berücksichtigt werden.

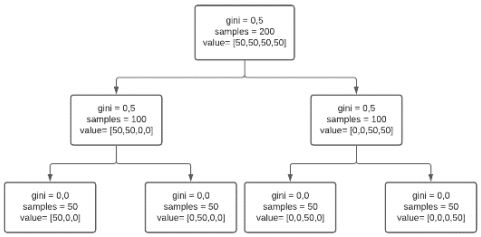



*Abbildung 5.7 Entscheidungsbaum Tiefe 3 Datensatz horizontal Split mit Random Forest*

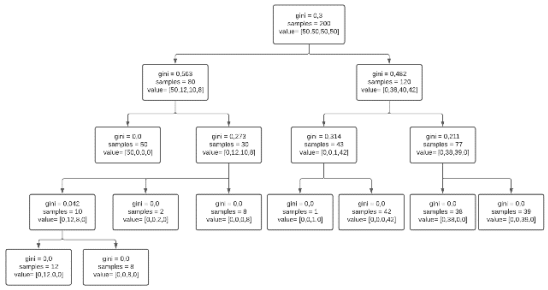


*Abbildung 5.8 Entscheidungsbaum Tiefe 5 Datensatz horizontal Split mit Oblique Forest*

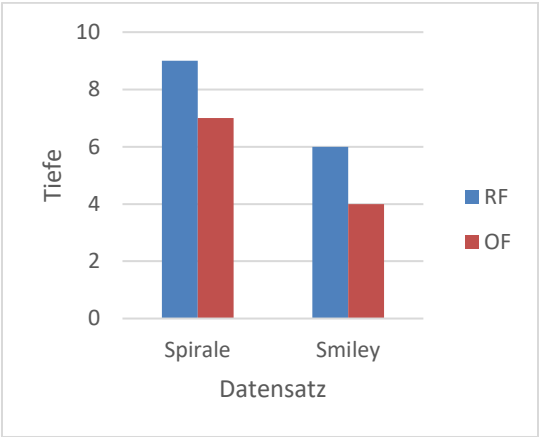



*Abbildung 5.9 Erreichte Tiefe für Split Datensätze*

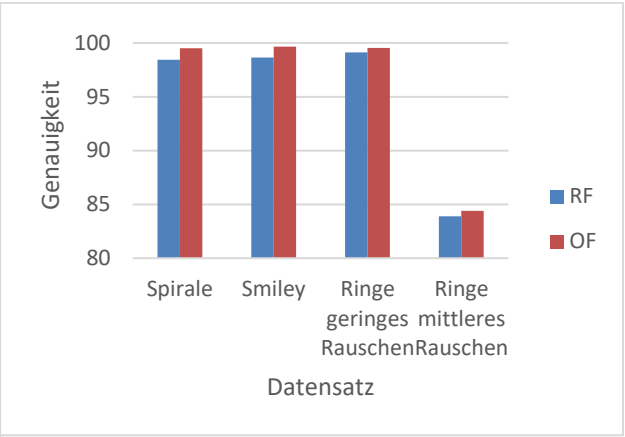



*Abbildung 5.10 Genauigkeit der Datensätze mit geom. Form*

Datensätze, wie in Kapitel 3.4.2.2 vorgestellt, die eine geometrische Struktur aufweisen, stellen für beide Verfahren kein Problem dar (vgl. Abbildung 5.10). Deutliche Einbußen in der Genauigkeit können beobachtet werden, wenn ein größerer Grad an Rauschen bei den Datenpunkten vorliegt. Dies lässt sich durch die Berechnung des Splits für die Bildung der Teilmengen erklären. Ein Rauschen in den Daten führt dabei bei beiden Verfahren zu einer größeren Anzahl benötigten Splits und somit deutlich tieferen und komplexeren Bäumen.

# 6 Schlussbetrachtung

Im Rahmen dieser Arbeit wurden die Klassifikationsverfahren Random Forest und Oblique Forest auf ihre Leistung hin untersucht. Es zeigt sich das Oblique Forest in den meisten Fällen zu einem genaueren Ergebnis führte. Nachteilig war eine erhöhte Rechenzeit für die Erstellung der Klassifikationsmodelle zu beobachten. Dies lässt sich anhand der Komplexität der Aufteilung erklären. Für Random Forest wächst die Suche nach der „optimalen“ orthogonalen Aufteilung, im schlimmsten Fall linear in der Anzahl der Trainingsmuster an jedem Knoten und der Quadratwurzel der Merkmalsdimension. Im Fall vom Oblique Forest Verfahren ist die Suche nach der „optimalen“ Schrägaufteilung NP-schwer wobei die Komplexität durch die Anzahl an Schrägaufteilungen beschränkt ist.

Die Untersuchung nach den Auswirkungen der geometrischen Form zeigten, dass der Oblique Forest in der Lage ist, Informationen von Variablen zu berücksichtigen, die für univariate Verfahren nicht von Bedeutung sind.

Grundsätzlich lässt sich festhalten, dass als Out-of-the-box-Lösung der Oblique Forst Klassifikator zu genaueren Ergebnissen führen kann. Einschränken finden sich hingegen in vorhandenen Implementationen und verfügbaren Bibliotheken. Zu beachten ist zudem das System, auf dem das Modell generiert werden soll. Unter Windows bedarf es einem deutlichen Zeitaufwand, um eine lauffähige Implementation zu realisieren. Weiterhin bieten diese nur eine begrenzte Möglichkeit der Optimierung durch Hyperparameter. Für Random Forest hingegen stehen viele Bibliotheken für unterschiedliche Programmiersprachen und Systeme zur Verfügung. Eine grobe Implementierung ist binnen weniger Minuten realisierbar und einsatzbereit. Spezielle Bibliotheken bieten dem Anwender deutlich größeren Spielraum seitens der Optimierung von Hyperparametern, wodurch sich die Ergebnisse gut an die des Oblique Forest annähern lassen.

# IV. Quellenverzeichnis

2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011). 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Colorado Springs, CO, USA, 6/20/2011 - 6/25/2011: IEEE / Institute of Electrical and Electronics Engineers Incorporated.

A. Chriminasi, S. Bucvciarelli (2009): Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes author={A. Criminisi and S. Bucciarelli}, year={2009} }.

Amit, Yali; Geman, Donald (1997): Shape Quantization and Recognition with Randomized Trees. In *Neural Computation* 9 (7), pp. 1545–1588. DOI: 10.1162/neco.1997.9.7.1545.

Andriy Mulyar (2020): Oblique Decision Trees in Python. A python interface to oblique decision tree implementations. github. Available online at https://github.com/AndriyMulyar/sklearn-oblique-tree.

Ben-Gal, Irad; Dana, Alexandra; Shkolnik, Niv; Singer, Gonen (2014): Efficient Construction of Decision Trees by the Dual Information Distance Method. In *Quality Technology & Quantitative Management* 11 (1), pp. 133–147. DOI: 10.1080/16843703.2014.11673330.

Breiman, Leo (1996): Bagging predictors. In *Mach Learn* 24 (2), pp. 123–140. DOI: 10.1007/BF00058655.

Breiman, Leo (2001). In *Mach Learn* 45 (1), pp. 5–32. DOI: 10.1023/A:1010933404324.

Cisco Systems, Inc (2021): Global_2021_Forecast_Highlights. VNI Complete Forecast Highlights. Cisco Systems, Inc. Available online at https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf.

Colin Shearer (2000): CRISP-DM The New Blueprint for Data Mining. In *Journal of Data Warehousing* Volume 5 (Number 4), pp. 13–22.

DASH, M.; LIU, H. (1997): Feature selection for classification. In *Intelligent Data Analysis* 1 (1-4), pp. 131–156. DOI: 10.1016/S1088-467X(97)00008-5.

Dietterich, Thomas G. (2000). In *Mach Learn* 40 (2), pp. 139–157. DOI: 10.1023/A:1007607513941.

Dua, Dheeru; Graff, Casey (2017): UCI Machine Learning Repository. Available online at http://archive.ics.uci.edu/ml.

Gareth, James (2017): An Introduction to Statistical Learning: with Applications in R. With applications in R. Corrected at 8th printing. New York, Heidelberg, Dordrecht, London: Springer Verlag (Springer texts in statistics).

Gérard Biau; Luc Devroye; Gábor Lugosi (2008): Consistency of Random Forests and Other Averaging Classifiers. In *J. Mach. Learn. Res.* 9, pp. 2015–2033. Available online at https://dl.acm.org/citation.cfm?id=1442799.

Gey, Servane; Poggi, Jean-Michel (2015): Discussion of 'Parallel construction of decision trees with consistently non-increasing expected number of tests'. In *Appl. Stochastic Models Bus. Ind.* 31 (1), pp. 79–80. DOI: 10.1002/asmb.2080.

Han, Sunwoo; Kim, Hyunjoong (2019): On the Optimal Size of Candidate Feature Set in Random forest. In *Applied Sciences* 9 (5), p. 898. DOI: 10.3390/app9050898.

Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009): The elements of statistical learning, second edition. Data mining, inference, and prediction. 2nd ed. New York: Springer (Springer Series in Statiscal).

Ho, Tin Kam (1995): Random decision forests. In : Third International Conference on Document Analysis and Recognition (Volume 2), Montréal, Canada, 14.08-15.08.1995. 3rd International Conference on Document Analysis and Recognition. Montreal, Que., Canada, 14-16 Aug. 1995. International Conference on Document Analysis and Recognition. [S.l.]n[s.n.]: IEEE Comput. Soc. Press, pp. 278–282.

Ho, Tin Kam (1998): The random subspace method for constructing decision forests. In *IEEE Trans. Pattern Anal. Machine Intell.* 20 (8), pp. 832–844. DOI: 10.1109/34.709601.

Hothorn, Torsten; Hornik, Kurt; Zeileis, Achim (2006): Unbiased Recursive Partitioning: A Conditional Inference Framework. In *Journal of Computational and Graphical Statistics* 15 (3), pp. 651–674. DOI: 10.1198/106186006X133933.

Hyafil, Laurent; Rivest, Ronald L. (1976): Constructing optimal binary decision trees is NP-complete. In *Information Processing Letters* 5 (1), pp. 15–17. DOI: 10.1016/0020-0190(76)90095-8.

Jiang, Hongying; Deng, Youping; Chen, Huann-Sheng; Tao, Lin; Sha, Qiuying; Chen, Jun et al. (2004): Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. In *BMC bioinformatics* 5, p. 81. DOI: 10.1186/1471-2105-5-81.

Katuwal, Rakesh; Suganthan, P. N.; Le Zhang (2020): Heterogeneous oblique random forest. In *Pattern Recognition* 99, p. 107078. DOI: 10.1016/j.patcog.2019.107078.

Kemmner, Götz-Andreas (1991): Anwenderorientierte Dezentralisierung von PPS-Systemen. Berlin, Heidelberg: Springer Berlin Heidelberg (Forschung für die Praxis, Berichte aus dem Forschungsinstitut für Rationalisierung (FIR) und dem Lehrstuhl und Institut für Arbeitswissenschaft (IAW) der Rheinisch-Westfälischen Technischen Hochschule Aachen, 39).

Kleinberg, E. M. (1990): Stochastic discrimination. In *Ann Math Artif Intell* 1 (1-4), pp. 207–239. DOI: 10.1007/BF01531079.

Kleinberg, E. M. (1996): An overtraining-resistant stochastic modeling method for pattern recognition. In *Ann. Statist.* 24 (6). DOI: 10.1214/aos/1032181157.

Kleinberg, E. M. (2000): On the algorithmic implementation of stochastic discrimination. In *IEEE Trans. Pattern Anal. Machine Intell.* 22 (5), pp. 473–490. DOI: 10.1109/34.857004.

Lin, Yi; Jeon, Yongho (2006): Random Forests and Adaptive Nearest Neighbors. In *Journal of the American Statistical Association* 101 (474), pp. 578–590. DOI: 10.1198/016214505000001230.

Max Bramer (2007): Principles of Data Mining. London: Springer London.

Menze, Bjoern H.; Kelm, B. Michael; Masuch, Ralf; Himmelreich, Uwe; Bachert, Peter; Petrich, Wolfgang; Hamprecht, Fred A. (2009): A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. In *BMC bioinformatics* 10, p. 213. DOI: 10.1186/1471-2105-10-213.

Menze, Bjoern H.; Petrich, Wolfgang; Hamprecht, Fred A. (2007): Multivariate feature selection and hierarchical classification for infrared spectroscopy: serum-based detection of bovine spongiform encephalopathy. In *Analytical and bioanalytical chemistry* 387 (5), pp. 1801–1807. DOI: 10.1007/s00216-006-1070-5.

Murthy, S. K.; Kasif, S.; Salzberg, S. (1994): A System for Induction of Oblique Decision Trees. In *jair* 2, pp. 1–32. DOI: 10.1613/jair.63.

Murthy, Sreerama K. (1998). In *Data Mining and Knowledge Discovery* 2 (4), pp. 345–389. DOI: 10.1023/A:1009744630224.

Norouzi, Mohammad; Collins, Maxwell D.; Fleet, David J.; Kohli, Pushmeet (2015): CO2 Forest: Improved Random Forest by Continuous Optimization of Oblique Splits. Available online at http://arxiv.org/pdf/1506.06155v2.

Novakovic, Jasmina; Strbac, Perica; Bulatovic, Dusan (2011): Toward optimal feature selection using ranking methods and classification algorithms. In *Yugosl J Oper Rres* 21 (1), pp. 119–135. DOI: 10.2298/YJOR1101119N.

Organisation for Economic Co-operation and Development (2019): Making decentralisation work. A handbook for policy-makers. Paris: OECD (OECD Multi-level Governance Studies).

Pal, M. (2005): Random forest classifier for remote sensing classification. In *International Journal of Remote Sensing* 26 (1), pp. 217–222. DOI: 10.1080/01431160412331269698.

Pete Chapman (1999): The CRISP-DM User Guide. Brussels SIG Meeting. Brussels, 1999. Available online at https://s2.smu.edu/~mhd/8331f03/crisp.pdf.

Piryonesi, S. Madeh; El-Diraby, Tamer E. (2020): Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems. In *J. Transp. Eng., Part B: Pavements* 146 (2), p. 4020022. DOI: 10.1061/JPEODX.0000175.

Provost, Foster; Fawcett, Tom (2013): Data science for business. What you need to know about data mining and data-analytic thinking. 1st ed. Sebastopol, Calif.: O'Reilly. Available online at

http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=619895.

Quinlan, J. R. (1986): Induction of decision trees. In *Mach Learn* 1 (1), pp. 81–106. DOI: 10.1007/BF00116251.

Reilly Meinert (2019): Optimizing Hyperparameters in Random Forest Classification. What hyperparameters are, how to choose hyperparameter values, and whether or not they're worth your time. Available online at https://towardsdatascience.com/optimizing-hyperparameters-in-random-forest-classification-ec7741f9d3f6.

Rokach, Lior; Maimon, Oded (2008): Data mining with decision trees. Theory and applications. Singapore: World Scientific (Series in machine perception and artificial intelligence, v. 69). Available online at http://site.ebrary.com/lib/academiccompletetitles/home.action.

Ruiz, Roberto; Riquelme, José C.; Aguilar-Ruiz, Jesús S. (2003): Fast Feature Ranking Algorithm. In Vasile Palade, Robert J. Ed Howlett, L. C. Jain (Eds.): Knowledge-based intelligent information and engineering systems. 7th International conference, vol. 2773. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 325–331.

Schmoldt, A.; Benthe, H. F.; Haberland, G. (1975): Digitoxin metabolism by rat liver microsomes. In *Biochemical pharmacology* 24 (17), pp. 1639–1641.

Scornet, Erwan; Biau, Gérard; Vert, Jean-Philippe (2015): Consistency of random forests. In *Ann. Statist.* 43 (4). DOI: 10.1214/15-AOS1321.

Shalev-Shwartz, Shai; Ben-David, Shai (2019): Understanding machine learning. From theory to algorithms. 12th printing. Cambridge: Cambridge University Press.

Shen, Kai-Quan; Ong, Chong-Jin; Li, Xiao-Ping; Hui, Zheng; Wilder-Smith, Einar P. V. (2007): A feature selection method for multilevel mental fatigue EEG classification. In *IEEE transactions on bio-medical engineering* 54 (7), pp. 1231–1237. DOI: 10.1109/TBME.2007.890733.

Strobl, Carolin; Malley, James; Tutz, Gerhard (2009): An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. In *Psychological methods* 14 (4), pp. 323–348. DOI: 10.1037/a0016973.

Svetnik, Vladimir; Liaw, Andy; Tong, Christopher; Culberson, J. Christopher; Sheridan, Robert P.; Feuston, Bradley P. (2003): Random forest: a classification and regression tool for

compound classification and QSAR modeling. In *Journal of chemical information and computer sciences* 43 (6), pp. 1947–1958. DOI: 10.1021/ci034160g.

Vanderplas, Jacob T. (2017): Python data science handbook. Essential tools for working with data / Jake VanderPlas. Beijing: O'Reilly.

Witten, I. H.; Frank, Eibe; Hall, Mark A. (2011): Data mining. Practical machine learning tools and techniques / Ian H. Witten, Eibe Frank, Mark A. Hall. 3rd ed. Burlington, MA: Morgan Kaufmann ([Morgan Kaufmann series in data management systems]).

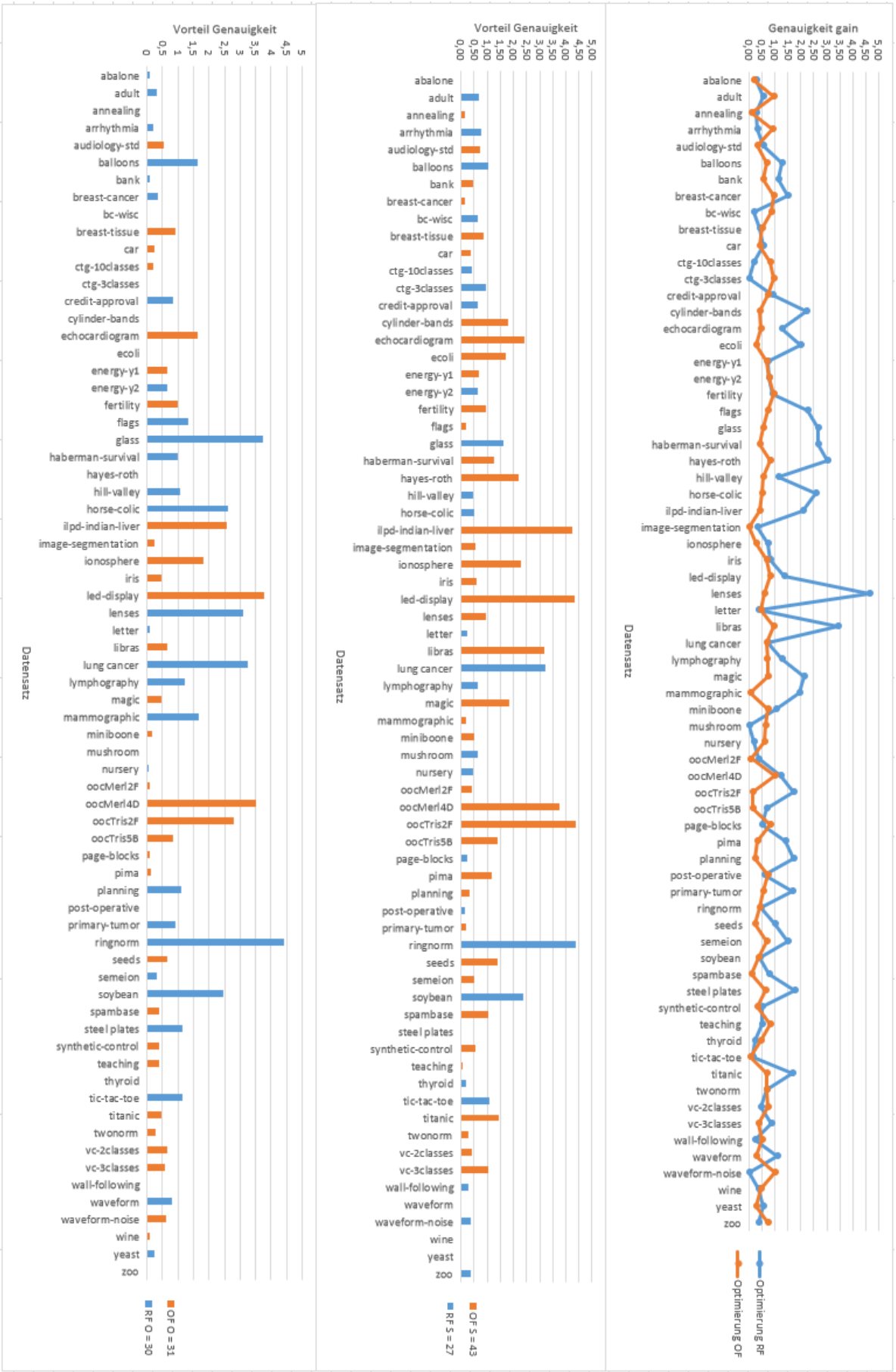
Xiaogang Su; Chih-Ling Tsai; Hansheng Wang; David M. Nickerson; Bogong Li (2009): Subgroup Analysis via Recursive Partitioning. In *Journal of Machine Learning Research* 10 (5), pp. 141–158. Available online at http://jmlr.org/papers/v10/su09a.html.

# V. Anhang

Alle Python und R Scripte können unter der folgenden Adresse eingesehen werden:

https://github.com/Kunahn/Master-Thesis

| Datensatz | Instanzen | Features | Klassen | RF n=100 | OF n=100 | Ber. RF 100 | Ber. OF 100 | RF n=500 | OF n=500 | Ber. RF 500 | Ber. OF 500 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| abalone | 4177 | 8 | 3 | 65,84 | 65,82 | 0,237 | 0,276 | 66,1 | 65,99 | 0,771 | 1,443 |
| adult | 48842 | 14 | 2 | 85,88 | 85,17 | 1,983 | 2,620 | 86,43 | 86,12 | 9,390 | 12,655 |
| annealing | 798 | 38 | 6 | 77,81 | 77,97 | 0,217 | 0,297 | 78,06 | 78,05 | 0,985 | 1,649 |
| arrhythmia | 452 | 262 | 13 | 82,01 | 81,22 | 3,433 | 3,814 | 82,31 | 82,11 | 16,785 | 18,076 |
| audiology-std | 226 | 59 | 18 | 82,87 | 83,62 | 0,257 | 0,345 | 83,4 | 83,93 | 1,039 | 1,600 |
| balloons | 16 | 4 | 2 | 50,76 | 49,70 | 0,125 | 0,175 | 52,01 | 50,37 | 0,478 | 0,802 |
| bank | 45211 | 17 | 2 | 86,64 | 87,09 | 1,760 | 2,317 | 87,75 | 87,65 | 9,284 | 11,584 |
| breast-cancer | 286 | 9 | 2 | 78,14 | 78,29 | 0,149 | 0,125 | 79,61 | 79,26 | 0,568 | 1,002 |
| bc-wisc | 699 | 9 | 2 | 97,83 | 97,17 | 0,179 | 0,195 | 98,01 | 98,01 | 1,205 | 1,150 |
| breast-tissue | 106 | 9 | 6 | 69,03 | 69,90 | 0,132 | 0,152 | 69,45 | 70,37 | 0,385 | 0,902 |
| car | 1728 | 6 | 4 | 98,15 | 98,54 | 0,182 | 0,253 | 98,71 | 98,96 | 0,910 | 1,880 |
| ctg-10classes | 2126 | 21 | 10 | 99,44 | 99,03 | 0,239 | 0,148 | 99,62 | 99,83 | 1,142 | 1,985 |
| ctg-3classes | 2126 | 21 | 3 | 100 | 99,04 | 0,170 | 0,232 | 100 | 100 | 0,735 | 1,875 |
| credit-approval | 690 | 15 | 2 | 89,21 | 88,56 | 0,196 | 0,161 | 90,12 | 89,29 | 0,792 | 1,174 |
| cylinder-bands | 512 | 35 | 2 | 78,64 | 80,45 | 0,620 | 0,637 | 80,86 | 80,86 | 0,988 | 1,669 |
| echocardiogram | 131 | 10 | 2 | 80,73 | 83,17 | 0,137 | 0,146 | 81,99 | 83,64 | 0,593 | 1,067 |
| ecoli | 336 | 7 | 8 | 83,94 | 85,67 | 0,153 | 0,123 | 85,93 | 85,93 | 0,742 | 0,655 |
| energy-y1 | 768 | 8 | 3 | 94,13 | 94,83 | 0,172 | 0,199 | 94,84 | 95,49 | 0,883 | 1,375 |
| energy-y2 | 768 | 8 | 3 | 87,89 | 87,25 | 0,174 | 0,196 | 88,67 | 88,02 | 0,491 | 1,369 |
| fertility | 100 | 9 | 2 | 90,32 | 91,29 | 0,118 | 0,119 | 91,23 | 92,23 | 0,371 | 1,047 |
| flags | 194 | 28 | 8 | 63,53 | 63,73 | 0,147 | 0,165 | 65,78 | 64,43 | 0,504 | 1,249 |
| gender by name | 147270 | 4 | 2 | 61,29 | 62,58 | 9,745 | 11,654 | 61,407 | 62,71 | 47,419 | 52,156 |
| haberman-survival | 306 | 3 | 2 | 74,11 | 75,38 | 0,146 | 0,153 | 76,78 | 75,8 | 0,697 | 0,631 |
| hayes-roth | 132 | 3 | 3 | 86,78 | 88,99 | 0,122 | 0,121 | 89,8 | 89,8 | 0,595 | 0,525 |
| hill-valley | 606 | 100 | 2 | 58,73 | 58,26 | 0,138 | 0,134 | 59,86 | 58,79 | 1,127 | 1,475 |
| horse-colic | 300 | 25 | 2 | 63,98 | 63,46 | 0,146 | 0,144 | 66,54 | 63,94 | 0,271 | 1,380 |
| ilpd-indian-liver | 583 | 9 | 2 | 68,46 | 72,71 | 0,152 | 1,169 | 70,55 | 73,12 | 0,423 | 1,044 |
| image-segmentation | 210 | 19 | 7 | 13,34 | 13,90 | 0,127 | 0,126 | 13,67 | 13,9 | 0,286 | 1,037 |
| ionosphere | 351 | 33 | 2 | 91,17 | 93,45 | 0,132 | 0,141 | 91,9 | 93,72 | 0,532 | 0,738 |
| iris | 150 | 4 | 3 | 94,76 | 95,35 | 0,120 | 0,127 | 95,55 | 96,02 | 0,521 | 1,038 |
| led-display | 1000 | 7 | 10 | 64,28 | 68,62 | 0,174 | 0,188 | 65,64 | 69,43 | 0,668 | 1,185 |
| lenses | 24 | 4 | 3 | 84,23 | 85,17 | 0,054 | 0,085 | 88,86 | 85,76 | 0,150 | 0,198 |
| letter | 20000 | 16 | 26 | 96,04 | 95,82 | 1,578 | 2,493 | 96,38 | 96,28 | 8,398 | 9,372 |
| libras | 360 | 90 | 15 | 78,77 | 81,95 | 0,191 | 0,151 | 82,22 | 82,88 | 1,070 | 1,226 |
| lung cancer | 32 | 56 | 3 | 57,26 | 54,05 | 0,138 | 0,127 | 58 | 54,74 | 1,037 | 1,055 |
| lymphography | 148 | 18 | 4 | 85,74 | 85,08 | 0,113 | 0,135 | 87,01 | 85,78 | 0,215 | 0,918 |
| magic | 19020 | 10 | 2 | 83,87 | 85,72 | 1,550 | 1,735 | 85,97 | 86,44 | 7,805 | 8,744 |
| mammographic | 961 | 5 | 2 | 82,59 | 82,80 | 0,164 | 0,169 | 84,53 | 82,84 | 0,426 | 0,860 |
| miniboone | 130064 | 50 | 2 | 93,66 | 94,18 | 48,716 | 66,812 | 94,7 | 94,88 | 241,658 | 313,812 |
| mushroom | 8124 | 21 | 2 | 100 | 99,36 | 0,255 | 0,369 | 100 | 100 | 1,733 | 2,619 |
| nursery | 12960 | 8 | 5 | 99,53 | 99,08 | 1,268 | 1,219 | 99,71 | 99,66 | 6,762 | 5,951 |
| oocMerl2F | 1022 | 25 | 3 | 92,65 | 93,06 | 0,315 | 0,352 | 93,03 | 93,13 | 1,257 | 1,972 |
| oocMerl4D | 1022 | 25 | 3 | 77,13 | 80,90 | 0,328 | 0,305 | 78,36 | 81,89 | 1,136 | 1,684 |
| oocTris2F | 912 | 25 | 2 | 78,96 | 83,35 | 0,262 | 0,237 | 80,67 | 83,46 | 1,026 | 1,123 |
| oocTris5B | 912 | 32 | 3 | 91,78 | 93,17 | 0,272 | 0,289 | 92,46 | 93,3 | 1,229 | 1,943 |
| page-blocks | 5473 | 10 | 5 | 98,05 | 97,81 | 0,670 | 0,721 | 98,54 | 98,64 | 2,920 | 4,108 |
| pima | 768 | 8 | 2 | 75,86 | 77,06 | 0,157 | 0,165 | 77,25 | 77,38 | 0,572 | 0,676 |
| planning | 182 | 12 | 2 | 72,73 | 73,08 | 0,552 | 0,130 | 74,44 | 73,33 | 0,191 | 1,199 |
| post-operative | 90 | 8 | 3 | 73,01 | 72,86 | 0,803 | 0,090 | 73,59 | 73,59 | 0,673 | 1,155 |
| primary-tumor | 330 | 17 | 15 | 56,24 | 56,45 | 0,139 | 0,154 | 57,91 | 57 | 1,083 | 1,199 |
| ringnorm | 7400 | 20 | 2 | 97,01 | 92,63 | 1,206 | 0,962 | 97,45 | 93,03 | 3,765 | 4,999 |
| seeds | 210 | 7 | 3 | 93,2 | 94,61 | 0,241 | 0,127 | 94,18 | 94,83 | 0,650 | 1,268 |
| semeion | 1593 | 256 | 10 | 93,12 | 93,63 | 0,317 | 0,238 | 94,62 | 94,3 | 1,179 | 1,135 |
| soybean | 307 | 35 | 18 | 39,64 | 37,25 | 0,836 | 0,174 | 40,06 | 37,61 | 1,038 | 1,427 |
| spambase | 4601 | 57 | 2 | 94,47 | 95,51 | 0,461 | 0,639 | 95,22 | 95,61 | 2,251 | 3,078 |
| steel plates | 1941 | 27 | 7 | 78,36 | 78,33 | 0,268 | 0,278 | 80,11 | 78,98 | 1,278 | 1,812 |
| synthetic-control | 600 | 60 | 6 | 97,99 | 98,55 | 0,173 | 0,176 | 98,48 | 98,87 | 1,202 | 1,049 |
| teaching | 151 | 5 | 3 | 63,12 | 63,19 | 0,106 | 0,104 | 63,6 | 64 | 0,791 | 1,181 |
| thyroid | 3772 | 21 | 3 | 98,64 | 98,43 | 0,249 | 0,243 | 98,87 | 98,87 | 0,962 | 1,258 |
| tic-tac-toe | 958 | 9 | 2 | 98,78 | 97,71 | 0,199 | 0,183 | 98,93 | 97,77 | 0,469 | 0,958 |
| titanic | 2201 | 3 | 2 | 76,34 | 77,77 | 0,219 | 0,245 | 77,99 | 78,46 | 1,327 | 1,786 |
| twonorm | 7400 | 20 | 2 | 96,89 | 97,19 | 0,881 | 1,011 | 97,58 | 97,85 | 4,294 | 5,445 |
| vc-2classes | 310 | 6 | 2 | 82,67 | 83,08 | 0,124 | 0,133 | 83,14 | 83,8 | 0,633 | 1,202 |
| vc-3classes | 310 | 6 | 3 | 84,26 | 85,33 | 0,122 | 0,135 | 85,12 | 85,69 | 0,570 | 1,356 |
| wall-following | 5456 | 24 | 4 | 99,57 | 99,28 | 0,787 | 1,076 | 99,79 | 99,76 | 3,983 | 4,431 |
| waveform | 5000 | 21 | 3 | 85,24 | 85,25 | 0,700 | 0,973 | 86,34 | 85,52 | 4,091 | 4,829 |
| waveform-noise | 5000 | 40 | 3 | 86,28 | 84,92 | 0,928 | 1,261 | 87,11 | 85,89 | 4,429 | 6,251 |
| wine | 179 | 13 | 3 | 98,36 | 98,40 | 0,125 | 0,120 | 98,73 | 98,83 | 1,051 | 0,629 |
| yeast | 1484 | 8 | 10 | 63,45 | 63,48 | 0,185 | 0,174 | 64 | 63,76 | 1,267 | 1,635 |
| zoo | 101 | 16 | 7 | 99,09 | 98,72 | 0,134 | 0,134 | 99,44 | 99,44 | 0,328 | 0,401 |


Vorteil Genauigkeit
0 0,5 1 1,5 2 2,5 3 3,5 4 4,5 5
abalone adult annealing arrhythmia audiology-std balloons bank breast-cancer bc-wisc breast-tissue car ctg-10classes ctg-3classes credit-approval cylinder-bands echocardiogram ecoli energy-y1 energy-y2 fertility flags glass haberman-survival hayes-roth hill-valley horse-colic ilpd-indian-liver image-segmentation ionosphere iris led-display lenses letter libras lung cancer lymphography magic mammographic miniboone mushroom nursery oocMerl2F oocMerl4D oocTris2F oocTris5B page-blocks pima planning post-operative primary-tumor ringnorm seeds semeion soybean spambase steel plates synthetic-control teaching thyroid tic-tac-toe titanic twonorm vc-2classes vc-3classes wall-following waveform waveform-noise wine yeast zoo
Datensatz
RF O = 30
OF O = 31
Vorteil Genauigkeit
0,00 0,50 1,00 1,50 2,00 2,50 3,00 3,50 4,00 4,50 5,00
abalone adult annealing arrhythmia audiology-std balloons bank breast-cancer bc-wisc breast-tissue car ctg-10classes ctg-3classes credit-approval cylinder-bands echocardiogram ecoli energy-y1 energy-y2 fertility flags glass haberman-survival hayes-roth hill-valley horse-colic ilpd-indian-liver image-segmentation ionosphere iris led-display lenses letter libras lung cancer lymphography magic mammographic miniboone mushroom nursery oocMerl2F oocMerl4D oocTris2F oocTris5B page-blocks pima planning post-operative primary-tumor ringnorm seeds semeion soybean spambase steel plates synthetic-control teaching thyroid tic-tac-toe titanic twonorm vc-2classes vc-3classes wall-following waveform waveform-noise wine yeast zoo
Datensatz
RF S = 27
OF S = 43
Genauigkeit gain
0,00 0,50 1,00 1,50 2,00 2,50 3,00 3,50 4,00 4,50 5,00
abalone adult annealing arrhythmia audiology-std balloons bank breast-cancer bc-wisc breast-tissue car ctg-10classes ctg-3classes credit-approval cylinder-bands echocardiogram ecoli energy-y1 energy-y2 fertility flags glass haberman-survival hayes-roth hill-valley horse-colic ilpd-indian-liver image-segmentation ionosphere iris led-display lenses letter libras lung cancer lymphography magic mammographic miniboone mushroom nursery oocMerl2F oocMerl4D oocTris2F oocTris5B page-blocks pima planning post-operative primary-tumor ringnorm seeds semeion soybean spambase steel plates synthetic-control teaching thyroid tic-tac-toe titanic twonorm vc-2classes vc-3classes wall-following waveform waveform-noise wine yeast zoo
Datensatz
Optimierung OF
Optimierung RF

# Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben kann.

_____ _____

Jena, 09.08.2021 Dario Dubberstein

# Veröffentlichung der Arbeit

Ich stimme der Veröffentlichung der Arbeit im Rahmen von Forschung und Lehre an der Friedrich-Schiller-Universität Jena zu.

_____ _____

Jena, 09.08.2021 Dario Dubberstein