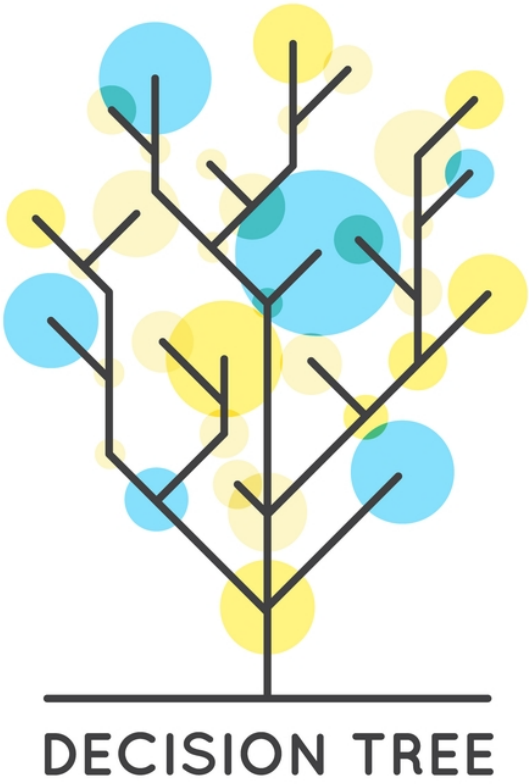
Fachhochschule Nordwestschweiz
Hochschule für Technik

# GDS2020 Bericht Flavio Müller
## Entscheidungsbäume




Flavio Müller

Windisch, 27.11.2020

Studiengang Data Science
Bahnhofstrasse, 5
5210, Windisch
T +41 76 802 30 10
flavio.mueller@students.fhnw.ch
www.fhnw.ch

# Inhaltsverzeichnis



# 1 Abkürzungsverzeichnis

CART ..... *Classification and Regression Trees*
DT ..... *Decision Tree / Decision Trees*
HP ..... *Hyperparameter*
ML ..... *Machine Learning*
SL ..... *Supervised Learning*
ZV ..... *Zielvariable*

# 2 Abstract

Entscheidungsbäume (englisch: Desicion Trees im nachfolgenden DT genannt, Abbildung 1) sind Supervised Machine Learning Algorithmen, welche vor allem wegen der Nachvollziehbarkeit ihrer Entscheidungen eingesetzt werden. Der vorliegende Bericht gibt eine grobe Übersicht dieses Algorithmus und beschreibt Einsatz und Anwendungsgebiete, sowie die Vor- und Nachteile und die Funktionsweise. Die Informationen basieren zu grossen Teilen auf dem 2011 von Gordon S. Linoff und Michael J. A. Berry geschriebenen Buch «Data mining techniques : for marketing, sales, and customer relationship management». [1]

# 3 Einsatzgebiet

Maschinelles Lernen (ML) wird grundsätzlich in drei verschiedene Kategorien unterteilt: Supervised Learning (deutsch: Überwachtes Lernen, SL), Unsupervised Learning (deutsch: Unüberwachtes Lernen) und Reinforced Learning (deutsch: bestärkendes Lernen). Die meisten Algorithmen, darunter auch DT werden hauptsächlich oder ausschliesslich in einem dieser drei Gebiete gebraucht. DT sind klar im SL anzusiedeln. Dabei können sie für Klassifikations- und Regressionsprobleme angewandt werden. Die Funktionsweise bleibt dabei die gleiche, es werden Regeln gesucht, welche die zur Verfügung stehenden Daten möglichst homogen trennen. Der Unterschied liegt nur in der Art der Zielvariable (ZV) (qualitativ oder quantitativ).
Ein weiteres Einsatzgebiet von DT liegt in der explorativen Datenanalyse. Dabei veranschaulicht ein DT die Abhängigkeiten der ZV zu Kombinationen von anderen Variablen. Die damit gewonnenen Erkenntnisse können in weiteren Analysen gezielt eingesetzt werden.

# 4 Entwicklung für die Anwendung

Um ein DT-Modell zu entwickeln, kann wie in anderen Data Science Projekten nach dem CRISP-Data Mining Modell (Abbildung 2) vorgegangen werden. Dabei müssen zuerst Daten gesammelt, exploriert, gereinigt und transformiert werden. Am Ende dieser Prozesse sollten gereinigte und klassifizierte Daten vorliegen. Damit SL funktioniert, muss eine ZV definiert sein. Diese wird anhand des Anwendungsfalls definiert. Wenn zum Beispiel die Qualität eines Rotweines auf einer Skala (1-10) bewertet werden soll, müssen bereits Daten von Rotweinen vorliegen, welche nach dieser Skala bewertet wurden.
Sobald diese klassifizierten Daten vorliegen, werden sie in drei Gruppen aufgeteilt. Die erste und grösste Gruppe (meistens ca. 70%) bilden die Trainingsdaten. Aus ihnen wächst später der DT. Die übrigen Daten werden meist zu gleichen Teilen in Validations- und Testdaten aufgeteilt. Dabei werden die Validationsdaten gebraucht, um die Hyperparameter (HP) zu tunen oder aus verschiedenen ML-Modellen das Beste auszuwählen. Die Testdaten werden dem Modell nur einmal gezeigt, um die definitive Genauigkeit zu beurteilen.
Wenn das Modell trainiert, ein Optimum für die HP gefunden wurde und der Baum mittels den Testdaten getestet wurde, wird anhand der Genauigkeit und der Geschwindigkeit des Baumes beurteilt, ob er bereit ist für den Einsatz. Wenn dies der Fall sein sollte, wird das Modell in eine Pipeline oder eine Applikation integriert, um es automatisiert nutzen zu können. Während es im Einsatz ist, wird es dauernd überwacht, um es später mit den gesammelten Daten noch weiter zu verfeinern.
Für die Benutzung in der explorativen Datenanalyse müssen die Daten nicht unterteilt werden, da der DT zu diesem Zeitpunkt nicht für ML genutzt wird.

# 5 Funktionsweise des Algorithmus

Im Folgenden werde ich die Funktionsweise eines DT anhand der CART Algorithmus Familie aufzeigen.

## 5.1 Den Baum wachsen lassen

Vorausgesetzt, dass ein klassifiziertes Trainingsdatenset vorliegt, kann der Algorithmus anfangen die Daten zu teilen. Dabei werden alle Variablen als mögliche Teilvariable betrachtet.
Da die meisten CART Algorithmen nur binäre Teilungen machen können (genau 2 Kinder pro Teilung) müssen kategorische Variablen mit mehr als zwei Leveln gruppiert werden. Dabei werden alle möglichen Kombinationen gebildet und die Reinheit (Abbildung 3) in Abhängigkeit der ZV verglichen.
Bei numerischen Teilvariablen wird ein Schwellwert $S$ definiert, welcher alle Werte in zwei Gruppen aufteilt. Dabei ist $S$ immer grösser als der kleinste und kleiner als der grösste Wert. Nun wird für jeden möglichen Wert von $S$ die Reinheit der Kinder berechnet ($S$ wird immer nur einen einzigen Wert zwischen zwei Beobachtungen annehmen).

### 5.1.1 Quantifizierung der Reinheit

Die Reinheit wird abhängig vom Typ der ZV (kategorisch oder numerisch) mit unterschiedlichen Algorithmen berechnet. Bei einer numerischen ZV muss eine Fehlerfunktion definiert werden. In unserem Beispiel nutzen wir die empirische Varianz (Formel 1). Dabei wird für beide Untergruppen der Mittelwert (Formel 2) berechnet und die Summe der quadratischen Abweichung zum jeweiligen Mittelwert durch die Anzahl Beobachtungen in der Untergruppe geteilt. Dies wird nun für alle möglichen Schwellwerte (numerische Teilvariable) bzw. Kombinationen von Klassifikationsgruppen (kategorische Teilvariable) berechnet. Dabei wird der Schwellwert / die Kombination als Teilkriterium gewählt, welche die kleinste Varianz erzeugt.
Bei einer kategorischen ZV wird häufig der Gini-Index (Formel 3) als Quantifizierung der Reinheit gewählt, da er einfach zu berechnen ist. Dafür wird bei beiden möglichen Kindern die summierte, quadrierte Wahrscheinlichkeit für das Vorkommen der jeweiligen ZV-Klasse berechnet. Danach werden diese Werte nach Anzahl Beobachtung pro Kind gewichtet, summiert und von 1 abgezogen. Der resultierende Wert wird zwischen 0 und 1 sein. Hier wird die Kombination bzw. der Schwellwert als Kriterium ausgewählt, welcher den Gini-Index minimiert, da ein tieferer Gini-Index eine höhere Reinheit der ZV impliziert.

## 5.2 Den Baum zurückschneiden (Pruning)

DT sind sehr anfällig für Überanpassung (Variance). Das heisst, wenn sie nicht zurückgehalten werden, passen sie sich so gut den Trainingsdaten an, dass dies negative Auswirkungen auf die Genauigkeit des Modells hat. Um dies zu verhindern gibt es zwei Ansätze: Prepruning = Hyperparametertuning und Postpruning, welche meist in Kombination gebraucht werden.
Beim Prepruning werden die zur Verfügung stehenden HP (maximale Tiefe, mindeste Anzahl von Beobachtungen pro Blatt etc.) vor dem Wachsen des Baumes so angepasst, dass eine möglichst hohe Genauigkeit erzielt wird. Dies geschieht anhand des Validation Datasets. Dabei wird das Modell mehrmals mit verschiedenen Werten als HP trainiert. Der Algorithmus, welcher die HP anpasst, wird dabei entweder vordefinierte Werte brauchen (Manualsearch, Gridsearch und Randomsearch) oder er nimmt sich die Bayessche Optimierung zu Hilfe.
Beim Postpruning wird entweder mittels Kreuzvalidierungsverfahren und Minimum Cost Complexity Pruning (MCCP) oder mit dem simpleren und schnelleren Reduced Error Pruning (REP) der DT ausgehend von den Blättern zurückgeschnitten. Weniger gebräuchliche Methoden, wie z.B. Pessimistic Error Pruning (PEP), können auch bei der Wurzel beginnen um so irrelevante Teilungen zu identifizieren und wegzuschneiden.

Jegliches Pruning macht den DT anfälliger für Bias, da er nun simpler ist und schwache Muster evtl. nicht wahrnimmt, dafür sinkt die Anfälligkeit von Variance. Das Handwerk des Data Scientist ist nun den richtigen Kompromiss zwischen Bias und Variance zu finden. (Abbildung 4)

### 5.3 Testen

Um die Genauigkeit des DT zu testen, wird die ZV des Testdatensatzes durch das trainierte Modell vorhergesagt. Danach wird bei Klassifikationsbäumen die Fehlerrate berechnet. Bei Regressionsbäumen wird wiederum ein Streumass (z.B. Varianz) berechnet, um die Fehlerrate zu quantifizieren. So können verschiedene Modelle und Algorithmen miteinander verglichen werden.

### 5.4 Vorhersagen

Um neue Werte vorherzusagen wird eine neue, nicht klassifizierte Beobachtung (ZV ist leer), ausgehend von der Wurzel des DT, durch den Baum gelassen, bis sie bei einem Blatt angekommen ist. Bei Regressionsbäumen wird die ZV der neuen Beobachtung den durchschnittlichen Wert der ZV von allen Beobachtungen dieses Blattes annehmen. Bei Klassifikationsbäumen wird die ZV die Klasse annehmen, welche im Blatt am meisten repräsentiert ist. Neue Beobachtungen beeinflussen weder die Struktur noch die Werte in den Blättern des DT. Dazu müsste er neu trainiert werden.

## 6 Vor und Nachteile des Algorithmus

### 6.1 Vorteile

Der grösste Vorteil eines DT als ML-Algorithmus liegt in der Nachvollziehbarkeit seiner Vorhersagen. Denn diese basieren auf klar definierten und für Menschen verständlichen Regeln. Weiter sind DT sehr beliebt, weil sie für Anfänger und fachfremde Personen gut verständlich und anwendbar sind.
Zudem sind die trainierten DT-Modelle eher klein und dadurch sehr schnell darin, neue Daten zu klassifizieren. Während bei neuronalen Netzen Millionen von Parametern abgeglichen werden müssen, operieren DT nach ein paar definierten Regeln und sparen so an Laufzeit ein.

Ausserdem müssen die Daten vorgängig weder normalisiert noch skaliert werden, wie das bei anderen Algorithmen der Fall ist. Dies verkürzt die vorgängig nötigen Arbeitsschritte und spart Zeit. Zusätzlich wird der zeitliche Aufwand verkleinert, indem DT sehr gut mit fehlenden Werten umgehen können und diese nicht zwingend vorgängig bereinigt werden müssen.

### 6.2 Nachteile

Durch den simplen Aufbau eines DT, ist er anfällig für Bias. Da bei den meisten Algorithmen lediglich eine Variable für einen neuen Split in Betracht gezogen wird, können kombinierte Abhängigkeiten zur ZV nicht repräsentiert werden, was häufig in einem zu simplen Modell endet. Ein weiterer grosser Nachteil bei nicht korrekter Handhabung ist die Anfälligkeit für Variance oder Overfitting (durch Pruning zu verhindern). Ein weiterer Nachteil von DT liegt in der Stabilität. Eine kleine Änderung im Trainingsdatensatz könnte eine grosse Änderung in der Struktur des Baumes bewirken, was zu sehr unterschiedlichen Genauigkeiten führen kann.

### 6.3 Verbesserungen

Um die genannten Nachteile auszugleichen wurden Algorithmen entwickelt, welche zwar auf DT basieren, jedoch entscheidende Vorteile in der Genauigkeit liefern. Eine populäre Methode ist der Random Forest. Hierbei handelt es sich um eine Kombination von DT, welche zusammen

einen Wald repräsentieren. Dafür wird für jeden Baum im Wald nur ein Teil des Trainingsset verwendet. Zusätzlich wird beim Wachsen des Baumes bei jedem Split nur eine bestimmte Anzahl Variablen als mögliche Split-variable in Betracht gezogen. Dies hat den Effekt, dass auch kleinere und weniger ausgeprägte Strukturen in den Daten erkannt werden können.
Laut Oshiro, Pedro, Perez und Baranauskas von der Univerity of Sao Paulo [2] liegt die optimale Anzahl der DT in einem Random Forest zwischen 64 und 128. Da mehr DT die Komplexität erhöhen, sich aber die Genauigkeit (wenn überhaupt) nur gering verbessert. Um nun neue Daten zu klassifizieren, werden sie von allen Bäumen im Wald klassifiziert. Danach wird die Entscheidung mit den meisten Stimmen als Vorhersage des gesamten Waldes ausgegeben.

### 6.4 Vergleich mit anderen ML-Algorithmen

Im Vergleich mit anderen Algorithmen (Tabelle 1) des SL fällt auf, dass DT in keiner Kategorie sehr schlecht abschneiden. Dies ist auch ein Grund der hohen Beliebtheit dieses Verfahrens. Beim visuellen Vergleich (Abbildung 5) dieser vier Algorithmen fällt die stufenartige Trennung der Daten von DT ins Auge. Diese Stufen sind darauf zurückzuführen, dass jeweils nur eine Variable pro Split gebraucht wird. Daher wird ein DT bei solch einer Darstellung nie diagonale Linien erzeugen.

## 7 Anwendungsgebiete

### 7.1 Optimale Anwendungsgebiete

Durch ihre Schlichtheit kommen DT überall da zum Einsatz, wo die Vorhersage erklärbar und nachvollziehbar sein muss. Ein mögliches Einsatzszenario ist z.B. im Marketing, wenn vorhergesagt werden soll, ob ein Kunde anfällig für einen Wechsel zur Konkurrenz ist. Weiter finden DT (meist als Random Forst o.ä.) eine gute Anwendung bei der Vorhersage von Versicherungsansprüchen. Generell sind DT überall dort stark, wo Genauigkeit nicht oberste Priorität hat.

### 7.2 Nicht optimale Anwendungsgebiete

Ein weniger optimaler Anwendungsfall ist z.B. die Klassifizierung von Objekten in Bildern. Da der grösste Vorteil von DT ihre Schlichtheit ist, macht es wenig Sinn, Pixelwerte von Bildern als Ausgangsdaten zu brauchen, da diese von Menschen nur sehr schwer, wenn überhaupt interpretiert werden können. Weiter sind alle Anwendungsfälle, in denen man sehr genaue Vorhersagen braucht, schlecht für DT geeignet. So zum Beispiel in der Medizin oder in der Pharmaindustrie. Dort können für einige Fallstudien auch DT angewendet werden, wenn jedoch die Genauigkeit oberste Priorität hat, ist man mit einem neuronalen Netz meist besser bedient.

## 8 Fazit

DT sind einfache und nachvollziehbare ML-Algorithmen, welche Regressions- und Klassifikationsprobleme im SL lösen. Nebst dieser Anwendung glänzen sie in der explorativen Datenanalyse. Im Vergleich zu anderen Algorithmen sind sie einfach anzuwenden, da die Daten weder normalisiert noch skaliert werden müssen. DT als ML-Algorithmen können auch wie andere Algorithmen mit dem CRISP Datamining Modell gebraucht werden. Sie gehen nach dem «Teile & Herrsche» Prinzip vor, wobei bei jeder neuen Teilung die Option gewählt wird, welche die grösste Reinheit in den geteilten Daten bewirkt. Da DT eine eher schlechte mittlere Genauigkeit aufweisen, gibt es diverse Verbesserungen, welche auf DT aufbauen, so z.B. Random Forests. Optimale Anwendungsgebiete sind alle, in welchen die Genauigkeit eine untergeordnete Rolle spielt, und die Vorhersage nachvollziehbar sein muss.

# 9 Quellenverzeichnis


[1] G. S. Linoff and M. J. A. Berry, Data mining techniques : for marketing, sales, and customer relationship management, 3rd Edition ed., Indianapolis, IN: Wiley Publishing, Inc., 2011.

[2] Oshiro, Thais, Pedro, Perez, Baranauskas und José, «How Many Trees in a Random Forest?,» *Lecture notes in computer science,* Bd. 7376, Nr. 07, 2012.

[3] D. Perruchoud, «Data Science ist hot!,» Fachhochschule Nordwestschweiz Hochschule für Technik, 2020.

[4] J. Akinsola , F. Osisanwo, O. Awodele, J. O. Hinmikaiye , O. Olakanmi and J. Akinjobi , "Supervised Machine Learning Algorithms: Classification and Comparison," *International Journal of Computer Trends and Technology (IJCTT),* vol. 48, no. 3, pp. 128-138, 2017.

[5] s. V. Wikipedia-Autoren, «Empirische Varianz,» Wikipedia, Die freie Enzyklopädie., 21 03 2020. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Empirische_Varianz&oldid=197960147. [Zugriff am 26 11 2020].

[6] S. Tahsildar, «Gini Index For Decision Trees,» QuantInsti, 18 04 2019. [Online]. Available: https://blog.quantinsti.com/gini-index/. [Zugriff am 26 11 2020].

[7] S. Ponmani und R. Samuel, «Classification Algorithms in Data Mining - A Survey,» 2017.

[8] T. Hastie, «Trees, Bagging, Random Forests and Boosting,» Stanford University.

[9] J. Han, "Paper Summary: A Few Useful Things to Know About Machine Learning," Jekyll, 27 12 2016. [Online]. Available: https://otzslayer.github.io/machine-learning/domingos2012/. [Accessed 26 11 2020].

[10] J. Starmer, «Regression Trees, Clearly Explained!!!,» StatQuest, 20 08 2019. [Online]. Available: https://www.youtube.com/watch?v=g9c66TUylZ4. [Zugriff am 26 11 2020].

[11] J. Starmer, «StatQuest: Decision Trees,» StatQuest, 22 01 2018. [Online]. Available: https://www.youtube.com/watch?v=7VeUPuFGJHk. [Zugriff am 26 11 2020].

[12] J. Starmer, «StatQuest: Decision Trees, Part 2 - Feature Selection and Missing Data,» StatQuest, 29 01 2018. [Online]. Available: https://www.youtube.com/watch?v=wpNl-JwwplA. [Zugriff am 26 11 2020].

[13] J. Starmer, «StatQuest: Random Forests Part 1 - Building, Using and Evaluating,» StatQuest, 05 02 2018. [Online]. Available: https://www.youtube.com/watch?v=J4Wdy0Wc_xQ. [Zugriff am 26 11 2020].

[14] J. Mingers, «An Empirical Comparison of Pruning Methods for Decision Tree Induction,» *Machine Learning,* Bd. 4, pp. 227-243, 1989.

[15] H. Singh, «A Complete Guide to Building Decision Trees,» Medium, 18 10 2019. [Online]. Available: https://medium.com/analytics-vidhya/trees-part-i-a019fed0499e. [Zugriff am 26 11 2020].

# 10 Anhänge

## 10.1 Abbildungen

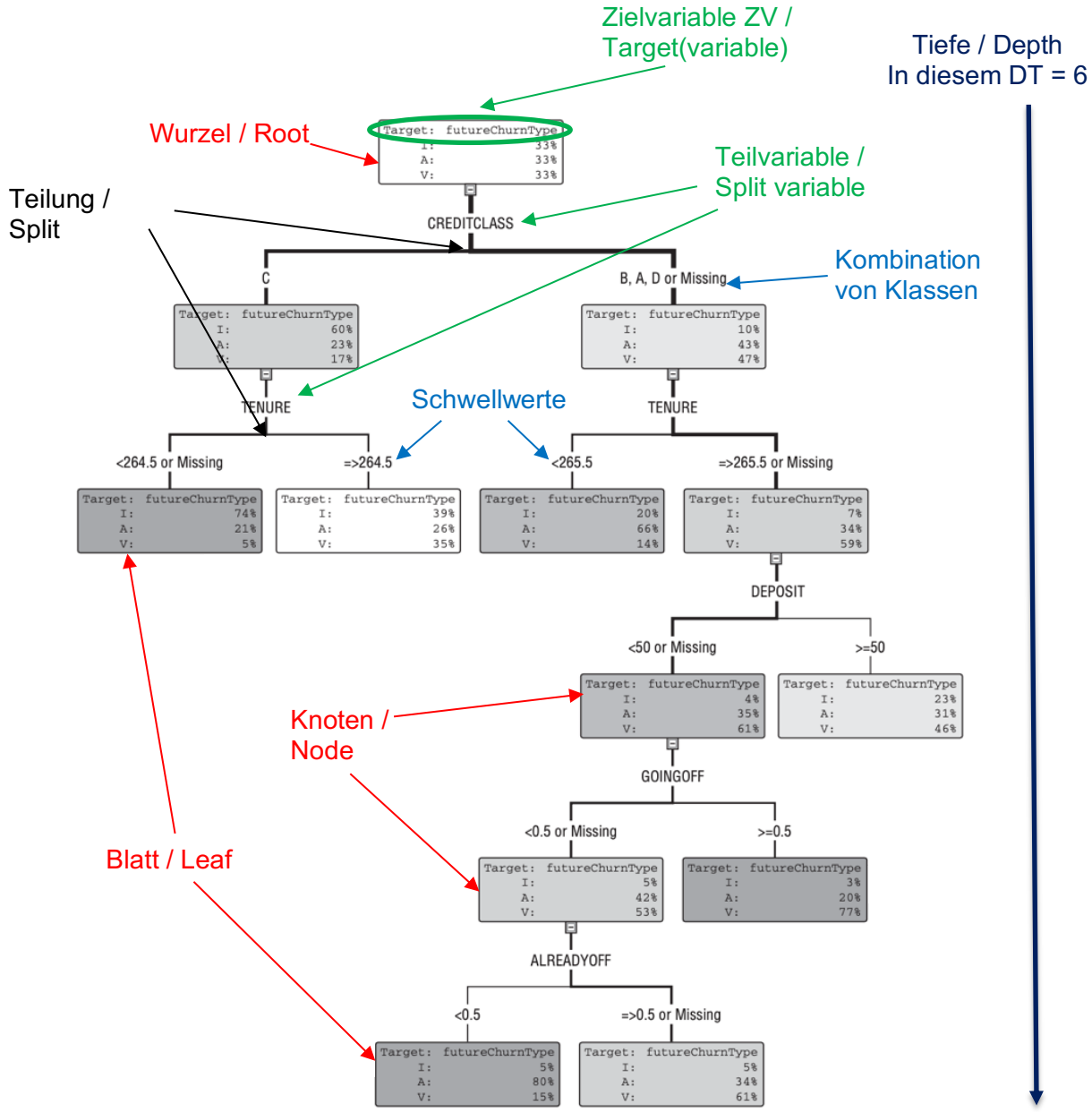



Abbildung 1: Graphische Darstellung eines Entscheidungsbaumes mit Beschriftung (Auszug aus [1])

Knoten = rot
Beobachtungsvariablen = grün
Modellparameter = blau
Hyperparameter = dunkelblau

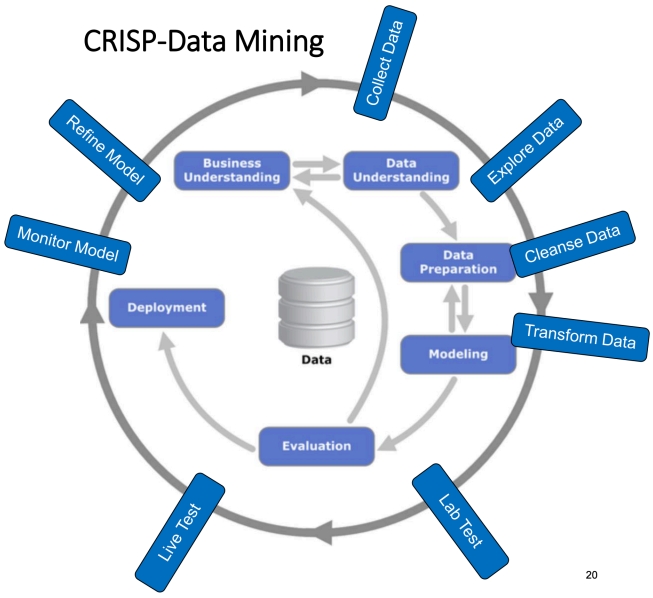



Abbildung 2: Graphische Darstellung CRIPS-Data Mining Modell (Auszug aus [3])

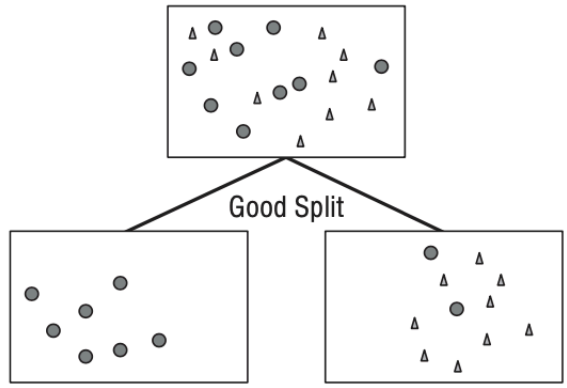



Abbildung 3: Graphische Darstellung eins guten Splits mit einer hohen Reinheit der Kinder (Auszug aus [1])

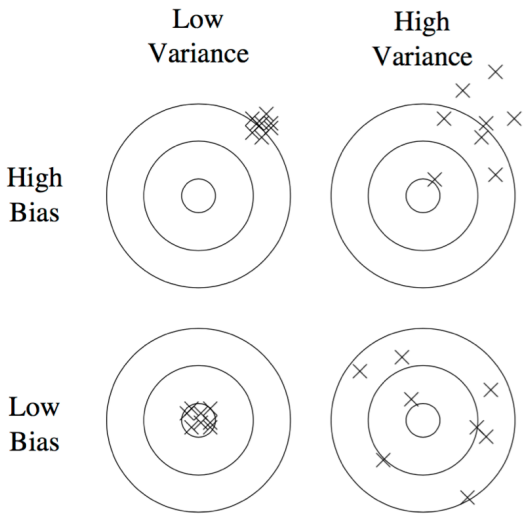



Abbildung 4: Graphische Darstellung Bias und Variance (Auszug aus [4])

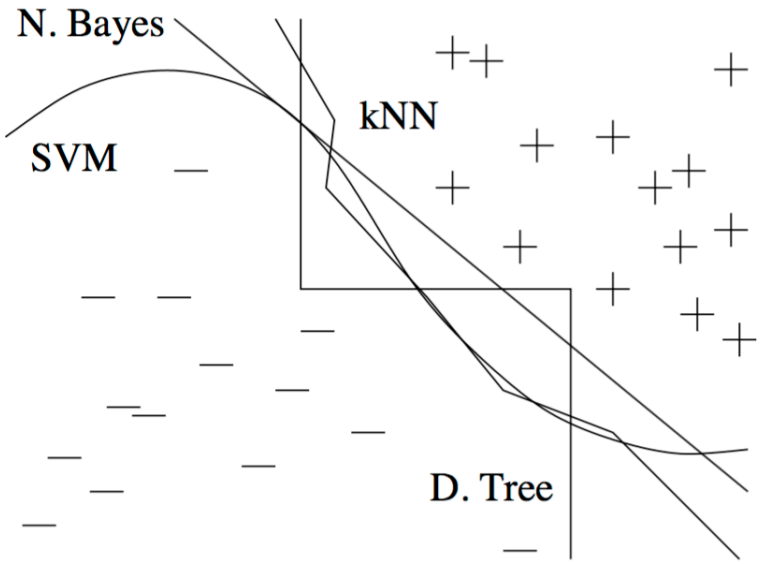



Abbildung 5: Graphische Darstellung verschiedener SL Algorithmen (Auszug aus [4])

## 10.2 Tabellen

| Kriterium | Entscheidungsbaum | Naiver Bayes-Klassifikator | Support Vektor Maschine | K-Nächster Nachbar |
|---|---|---|---|---|
| Genauigkeit generell | 2 | 1 | 4 | 2 |
| Geschwindigkeit trainieren | 3 | 4 | 1 | 4 |
| Geschwindigkeit Klassifizieren | 4 | 4 | 4 | 1 |
| Toleranz gegenüber fehlenden Werten | 3 | 4 | 2 | 1 |
| Toleranz gegenüber irrelevanten Attributen | 3 | 2 | 4 | 2 |
| Umgang mit der Gefahr von Überanpassung | 2 | 3 | 2 | 3 |
| Erklärbarkeit | 4 | 4 | 1 | 3 |

Tabelle 1: Vergleich SL Algorithmen (Teilauszug und Weiterverarbeitung aus [4]) Skala: 1 (schlecht) bis 4 (sehr gut)

## 10.3 Formeln

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Formel 1: empirische Varianz [5]

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Formel 2: Mittelwert [5]

$$G_i = 1 - \sum_{i=1}^{n}(P_i)^2$$

Formel 3: Gini-Index [6]