

Fakultät für Informatik

Studiengang B.Sc. Wirtschaftsinformatik

Evaluation von Decision Tree und Random Forest
Klassifikatoren in der Finanzdomäne

Bachelor Thesis

von

Felix Schuhbauer

am tt.mm.2020

Datum der Abgabe: tt.mm.jjjj

Erstprüfer: Prof. Dr. Marcel Tilly

Zweitprüfer: Prof. Dr. Andreas Krüger

ERKLÄRUNG

Ich versichere, dass ich diese Arbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Rosenheim, den tt.mm.jjjj

Felix Schuhbauer

Abstract

The random forest methodology has been successfully involved in various practical problems, including a data science hackathon on air quality prediction (<http://www.kaggle.com/c/dsg-hackathon>), chemoinformatics (Svetnik et al. 2003), ecology (Prasad et al. 2006; Cutler et al. 2007), 3D object recognition (Shotton et al. 2011), and bioinformatics (Díaz-Uriarte and de Andrés 2006), just to name a few.

Diese Arbeit untersucht, wie Random FOrests in der Finanzdomäne angewendet werden kann. Beispiele Stock Price Prediction und Credit Scoring. Unterfragen zeigen.

Die Untersuchungen zeigen, dass ERFOLGE.

Die Erfolge der intelligenten Computerprogramme Deep Blue, AlphaGo und AlphaGo Zero veranschaulichen den beeindruckenden ("reißerisch" wie in Papers? Oder hier nicht?) Fortschritt im Bereich des Machine Learnings. Sie geben dazu Anlass, Machine Learning Methoden in weiteren Domänen einzusetzen.

In der Finanzdomäne ist die Markteffizienzhypothese (EMH) seit Jahrzehnten eine grundlegende Theorie. Die EMH besagt, dass Aktienkurse einen Random Walk verfolgen und es daher nicht möglich ist, durch technische und fundamentale Analyse systematisch höhere Renditen zu erzielen als der Marktindex. Bisher konnte kein Mensch durch technische oder fundamentale Analyse diese Hypothese eindeutig beweisen oder widerlegen. Die Vermutung liegt nahe, dass intelligente Computerprogramme solche Analysen besser durchführen können und Muster erkennen können als der Mensch - wie auch in den hochkomplexen Brettspielen Schach und Go.

Diese Arbeit untersucht nun, inwiefern die EMH vor dem Hintergrund der steigenden Vorhersagegenauigkeit von Machine Learning Modellen noch Stand hält. Dazu werden Decision Tree und Random Forest Klassifikatoren zur Investitionsentscheidung von Aktien verwendet. Insbesondere Random Forest Modelle haben in der Literatur viel Aufmerksamkeit genossen und haben sich in verschiedenen Studien als zuverlässigster Klassifikator herausgestellt. Diese Arbeit evaluiert die Machine Learning-Methoden des Decision Trees und des Random Forests in Bezug auf Finanzdomäne. Die Markteffizienzhypothese (EMH) ist seit langem diskutiert. Es soll eruiert werden, inwiefern diese ML Methoden sinnvoll eingesetzt werden können, gemessen an Markttrenditen als Benchmark. Transaktionskosten werden dabei ignoriert. Dabei soll zwischen kurzfristigem (Zeithorizont < 6 Monate) und langfristigem Strategien (Zeithorizont >= 6 Monate) unterschieden werden. Dabei soll auch auf die Optimierung der Hyperparameter eingegangen werden. Zudem zeigt diese Arbeit, wie Hyperparameter sinnvoll gesetzt werden.

Die Untersuchungen ergeben, dass kurzfristige Strategien den Index - gemessen an der Rendite - nicht signifikant übertreffen können. Der Grund dafür sind unter anderem unvorhersehbare Ereignisse, wie Naturkatastrophen, Regulation oder öffentliche Äußerungen des Managements, die zu kurzfristigen Kursschwankungen führen aber durch die Modelle nicht erfasst werden.

Langfristige Strategien mit einem Zeithorizont von mindestens einem Jahr, hingegen, konnten unter der Verwendung eines Random Forest Klassifikators signifikant höhere Renditen

als der Index erzielen.

Schlagworte: Künstliche Intelligenz, Machine Learning, Klassifikation, Random Forest

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Historie der Künstlichen Intelligenz	2
1.3 Ziel der Arbeit	2
2 Theoretische Grundlagen	5
2.1 Machine Learning	5
2.2 Klassifikation	7
2.3 Decision Tree Klassifikator	8
2.3.1 Generelle Funktionsweise und Eigenschaften	8
2.3.2 Erstellung eines Decision Trees	10
2.3.3 Grundlegende Parameter	13
2.3.4 Optimierung von Decision Trees	14
2.4 Ensemble Methoden	14
2.4.1 Die Ensemble-Idee	14
2.4.2 Bagging und Boosting	15
2.4.3 Entscheidungsfindung im Ensemble	15
2.5 Random Forest Klassifikator	16
2.5.1 Generelle Funktionsweise und Eigenschaften	16
2.5.2 Erstellung eines Random Forests	17
2.5.3 Grundlegende (Hyper-)Parameter	18
2.6 Erfolgsmessung von Klassifikatoren	19
2.7 Underfitting, Overfitting und der Curse of Dimensionality	21
2.8 Besonderheiten bei der Klassifikation von Zeitreihen	22
2.9 Random Walk Theorie und Markteffizienzhypothese	23
2.10 Relevante Literatur	24
3 Methodik	27
3.1 Vorgehen	27
3.2 Tool-Stack und Bibliotheken	31
3.3 Datengrundlage	31
4 Ergebnisse	35
4.1 Erfolgsmessung der Klassifikatoren pro Zeithorizont	35
4.1.1 F-Maße	36
4.1.2 Precision und Recall	39
4.1.3 Analyse pro Aktiengruppe	40
4.2 Feature Importances in Abhängigkeit von dem Zeithorizont	44
4.3 Einfluss von Feature Extraction	46
4.4 Optimierung der Hyperparameter	47

5 Zusammenfassung	49
5.1 Zusammenfassung	49
5.2 Ausblick	49
A Anhang	51
A.1 Aktienverläufe absolut	51
A.2 Auswertungen der Klassifikatoren: F-Maße pro Aktie pro Horizont als Diagramme	53
A.3 Auswertungen der Klassifikatoren: Precision, Recall und F-Maß in Tabellen .	55
A.4 Auswertungen der Klassifikatoren pro Gruppe	58
A.5 Tuning Einfluss pro Gruppe ergänzen?	66
A.6 Jupyter Code? Vllt wichtigste Funktionen?	66
Literaturverzeichnis	67

Abbildungsverzeichnis

2.1	Beispielhafter Decision Tree für den Anwendungsfall der Kreditvergabe	9
2.2	Schematischer Ablauf einer Time Series Cross-Validation in den ersten vier Iterationen	22
3.1	Vorgehen in der Untersuchung vom Laden des Datensets bis zur Auswertung des Klassifikators	28
3.2	TBD	32
4.1	TBD	37
4.2	TBD	41
4.3	TBD	42
4.4	TBD	42
4.5	TBD	43
4.6	TBD	44
4.7	TBD	45
4.8	TBD	45
4.9	TBD	46
4.10	TBD	46
4.11	TBD	47
4.12	TBD	48
A.1	TBD	51
A.2	TBD	51
A.3	TBD	52
A.4	TBD	52
A.5	TBD	53
A.6	TBD	53
A.7	TBD	54
A.8	TBD	54
A.9	TBD	55
A.10	TBD	58
A.11	TBD	59
A.12	TBD	59
A.13	TBD	60
A.14	TBD	60
A.15	TBD	61
A.16	TBD	61
A.17	TBD	62
A.18	TBD	62
A.19	TBD	63
A.20	TBD	63
A.21	TBD	64

A.22 TBD	64
A.23 TBD	65
A.24 TBD	65

Tabellenverzeichnis

2.1 (Hyper-)Parameter eines Decision Trees und deren Bedeutung	13
2.2 (Hyper-)Parameter eines Random Forests und deren Bedeutung	19
2.3 Konfusionsmatrix	20
3.1 Die fünf zentralen Variablen für die Untersuchungen	27
3.2 TBD	32
3.3 TBD	33
4.1 Erfolgsmessung pro Klassifikator für die verschiedenen Zeithorizonte	36
4.2 Einteilung der Aktien in Gruppen zur weiteren Analyse	41
4.3 TBD	47
A.1 TBD	55
A.2 TBD	56
A.3 TBD	56
A.4 TBD	57
A.5 TBD	57
A.6 TBD	58

1 Einleitung

1.1 Motivation

Der Sieg des intelligenten Schachprogramms Deep Blue gegen den damaligen Schachweltmeister Garry Kasparov im Jahr 1997 gilt als Meilenstein in der Künstlichen Intelligenz. Das von IBM entwickelte Programm schaffte es erstmals, den Zustandsraum des Brettspiels Schach – mit einer Komplexität von 10^{50} – mittels Alpha-Beta-Suche auf dem Profi-Niveau in spiel-tauglicher Geschwindigkeit zu lösen. Bis zu 330 Millionen Spielpositionen konnte Deep Blue pro Sekunde auswerten. (REI?ERISCH OK?) Damit hat sich die Wahrnehmung von KI gewendet: von Science Fiction zur Realität. Aufgrund der weltweiten medialen Präsenz dieses Schach-Duell's hat KI in der Öffentlichkeit stark an Bekanntheit gewonnen. So veröffentlichte zum Beispiel in Großbritannien The Guardian den Artikel "Deep Blue win a giant step for computerkind"(<https://www.theguardian.com/theguardian/2011/may/12/deep-blue-beats-kasparov-1997>). In den USA berichteten unter anderem die New York Times ("Deep, Deeper, Deepest Blue", <https://www.nytimes.com/1997/05/18/weekinreview/deep-deeper-deepest-blue.html>) und das Wall Street Journal ("IBM's Winning 'Deep Blue' Is Still Product of Primates"<https://www.wsj.com/articles/SB863381328224169500>).

Ein weiterer Meilenstein fand im Jahr 2015 statt, als DeepMind's Programm AlphaGo den damaligen Go-Weltmeister Lee Sedol bezwang. Das asiatische Brettspiel Go weist einen Zustandsraum der Komplexität 10^{170} auf und konnte vor AlphaGo nicht auf Profi-Niveau und innerhalb der für einen Spielzug zulässigen Zeit gelöst werden. Maßgeblich war die Anwendung von Deep Learning in AlphaGo, um den Suchbaum effizienter zu durchsuchen (Silver u. a., 2017). Zwei Jahre später besiegte der Nachfolger, AlphaGo Zero, AlphaGo mit 100 zu Null. Der signifikante Fortschritt des Machine Learnings in den letzten 25 Jahren, oft anhand von Brettspielen veranschaulicht, regt Forscher und Unternehmen dazu an, diese Methoden auch in anderen Domänen zu erproben und einzusetzen.

Aufgrund ihrer Bedeutung für die Altersvorsorge sowie der monetären Anreize für einzelne Investoren, erfährt die Finanzdomäne Aufmerksamkeit in der KI-Forschung. Die Finanzdomäne umfasst institutionelle und private Kapitalgeber, die ihre angelegten finanziellen Mittel Kapitalnehmern, häufig Unternehmen oder Staaten, zur Verfügung stellen. Institutionelle Anleger, darunter Versicherungen, Stiftungen und Pensionskassen, investieren ihr Vermögen über Intermediäre, zum Beispiel Vermögensverwalter, in die Kapitalmärkte.

Der weltweit größte Vermögensverwalter BlackRock verwaltet ein Vermögen von über 6,5 Billionen US-Dollar(von Statista herausfinden, 2019) und setzt dabei KI ein. BlackRock hat eine Risiko- und Investmentplattform namens Aladdin entwickelt, die Daten analysiert und verarbeitet. In das Modell fließen über 2.000 täglich berechnete Faktoren ein, zum Beispiel Wechselkurse oder Zinssätze(BlackRock, 2019a). Laut dem Unternehmen werden über Aladdin "derzeit Anlagen von über 14 Billionen US-Dollar für mehr als 160 Kunden und insgesamt 30.000 Investmentportfolios verwaltet"(BlackRock, 2019b). Auf die Risikoplattform hat nicht nur BlackRock selbst Zugang, sondern auch andere Vermögensverwalter wie

Versicherer oder Vorsorgeeinrichtungen, die den Service entgeltlich erwerben können.

1.2 Historie der Künstlichen Intelligenz

- Dartmouth Konferenz (KI Als interdisziplinäres Forschungsgebiet, beschäftigt Forscher aus der Informatik, der Mathematik, der Neurowissenschaften, der Philosophie und weiterer Wissenschaften (Russell and Norvig, 2009, pp. 5-12). Die Fragestellung von KI wurde auf der Dartmouth Conference, 1956, formuliert als "how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves"(McCarthy u. a., 1955). KI beschäftigt sich also mit Programmen, die anhand von Daten Abstraktionen bilden und daraus lernen, um Fragen zu beantworten, die zuvor nur von Menschen gelöst werden konnten.)

- Turing Test? DELETE - AI Winter - Erfolge dank Verfügbarkeit von Daten und Rechenleistung - Dadurch konnten intelligente Programme wie die erwähnten Deep Blue oder AlphaGo entstehen. - AI Winter war somit vorbei, seit Jahrtausendwende erfährt Künstliche Intelligenz und zieht hohe Forschungsgelder an (Quelle: TBD).

Um die Erforschung von KI transparent zu machen und zu analysieren, veröffentlicht das Human-Centered AI Institute (HAI) an der Stanford Universität den AI Index. In 2018 hat das Institut festgestellt, dass sich KI-bezogene Paper TBD mal schneller entwickeln als alle auf TBD veröffentlichten Pape. Während sich seit 1996 die Anzahl der wissenschaftlichen Paper – gemessen mit der Scopus-Datenbank – fast verdreifacht hat, hat sich die Anzahl der Paper mit KI-Bezug im selben Zeitraum verachtfacht(Shoham u. a., 2018). TBD: In China viele Paper mit Regierungsbezug; in Europa Fokus mehr auf Wirtschaf.

Die Bedeutung von KI schlägt sich nicht nur in der Wirtschaft, sondern auch in der Politik nieder. Staaten machen darauf aufmerksam, dass KI einen zunehmenden Einfluss auf das Zusammenleben der Gesellschaft hat, und versuchen, in diesem Bereich eine führende Rolle einzunehmen. So hat die Bundesministerin für Bildung und Forschung, Anja Karliczek, KI als "Treiber des Wandels, unseres Zusammenlebens und unserer Arbeit"(Quelle: <https://www.bundesregierung.de/breg-de/service/bulletin/rede-der-bundesministerin-fuer-bildung-und-forschung-anja-karliczek-1581192>) beschrieben und das Wissenschaftsjahr 2019 - Künstliche Intelligenz ausgerufen (Quelle: <https://www.bildungsforschung.digital/de/wissenschaftsjahr-2019—kuenstliche-intelligenz-offiziell-gestartet-2507.html>). DELETE: Der Zweck eines Wissenschaftsjahres ist es, den Dialog zwischen der Forschung und der Gesellschaft zu fördern; es soll "Wissenschaft erlebbar machen, Debatten anregen, Zukunftsfragen beantworten und diskutieren"(Quelle: <https://www.bmbf.de/de/die-wissenschaftsjahre-229.html>).

Der Teilbereich von KI, der sich mit dem Lernen von beschäftigt, nennt sich Machine Learning. Diese Arbeit beschäftigt sich mit Klassifikatoren und ist somit im Machine Learning Teilbereich angesiedelt.

1.3 Ziel der Arbeit

Random Forests gehören zu den mächtigsten Machine Learning Algorithmen (Gron, 2017). In verschiedenen Domänen wurde ihre praktische Anwendbarkeit unter Beweis gestellt. Einige Beispiele sind Anwendungen in der Chemoinformatik (Svetnik u. a., 2003), Ökologie (Prasad u. a. 2006; Cutler u. a. 2007), 3D-Objekterkennung (Shotton u. a., 2011) und Bioinformatik

(Diaz-Uriarte u. Alvarez, 2006) sowie ein Data Science Hackathon zur Luftqualitätsvorhersage (<http://www.kaggle.com/c/dsg-hackathon>) (Tang u. a., 2018). Howard u. Bowles (2012) bezeichnen Random Forests als den erfolgreichsten Allzweck-Algorithmus der neueren Zeit.

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

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

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

- Forschungsfrage: Können DT und RF Klassifikatoren sinnvoll (besser als Dummy Classifier oder besser als Markt Index Performance) in der Finanzdomäne eingesetzt werden?
- Unterfrage 1: Können DT und RF sinnvoll für Credit Scoring (Anwendung 1) und Aktienempfehlungen (Anwendung 2) eingesetzt werden?
- Unterfrage 2: Wie sollten die Hyperparameter jeweils für DT und RF pro Anwendungsfall gesetzt werden?
- Unterfrage 2.5: Kann Pruning in Anwendung 1 und Anwendung 2 zur Vermeidung von Overfitting des DT beitragen?
- Unterfrage 3: Wie unterscheiden sich DT und RF Algorithmen bzw. wann eignet sich welcher Klassifikator?
- Unterfrage 3.5: Ist Meta Random Forest sinnvoll? Soft Voting vs Hard Voting?
- Unterfrage 4: Lassen sich relevante Effekte aus der realen Welt (Christmas Effekt, Dividendausschüttung, Jahresabschluss-Veröffentlichung, etc.) in den Modellen wiederfinden?
- Unterfrage 5: Sind die Ergebnisse mit der Markteffizienzhypothese vereinbar?
- Unterfrage 6: Welche Einschränkungen gibt es bei der Anwendung von DT und RF Klassifikatoren in der Finanzdomäne?

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

Auch wenn diese zwei Modelle in der Forschung große Beachtung finden (wie später unter "Aktuelle Literatur" gezeigt), sind Decision Tree und Random Forest Klassifikatoren noch nicht ausreichend erforscht.

Vor allem der Einfluss verschiedener Parametrisierungen auf die Trefferquote wurde noch nicht im Detail behandelt. Eine Parametrisierung bezeichnet eine spezielle Zusammenstellung der beeinflussbaren Attribute von Klassifikatoren. Zum Beispiel kann bei einem DT ein

Parameter zur maximalen Tiefe des Baumes gesetzt werden. Speziell in der Finanzdomäne fehlt eine gründliche Evaluation, wie sich diese Klassifikatoren unter verschiedenen Parametrisierungen verhalten.

Eine zentrale Frage für die Anwendung der Modelle liegt in der Auswahl der optimalen Hyperparameter. Diese muss der Anwender im Vorhinein festlegen. Häufig werden suboptimale Standard-Werte benutzt. Wenn man diese Werte jedoch optimiert, findet man eine schnelle und einfach anwendbare Verbesserung für das Modell. Daher sollte die Setzung der Hyperparameter untersucht werden.

Parametrisierungen vergleichen, wie beeinflussen die Genauigkeit der Modelle?

Somit leistet diese Arbeit einen Beitrag zur Anwendung von DTs und RFs, insbesondere innerhalb der Finanzdomäne. Die Arbeit soll erläutern, inwiefern die Methoden zur Investitionsentscheidung an den Aktienmärkten verwendet werden kann. Vor dem Hintergrund der Markteffizienzhypothese (siehe Fama, 1965) untersucht die Arbeit, ob Muster erkennbar und zur systematischen Erwirtschaftung überdurchschnittlicher - im Vergleich zur gesamten Marktentwicklung im Index - Renditen anwendbar sind.

Weiterhin wird auf verbreitete Probleme wie Overfitting eingegangen.

Die Zeitreihen der Aktienkurse einiger ausgewählter Aktien aus dem S&P 500 dienen dabei als Datengrundlage.

2 Theoretische Grundlagen

2.1 Machine Learning

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

Machine Learning gliedert sich in die drei Bereiche des Supervised-, Unsupervised- und Reinforcement Learnings. Reinforcement Learning umfasst Modelle, die durch Feedback im Form von Belohnungen und Bestrafungen lernen (Russell u. Norvig, 2009). Beim Unsupervised Learning werden die Modelle mit Daten trainiert, die zuvor nicht mit einer Klasse betitelt wurden. Stattdessen werden Ähnlichkeiten unter den Datensätzen gesucht, um sie zum Beispiel in Cluster einzuteilen. Diese Arbeit beschäftigt sich ausschließlich mit dem dritten Bereich, dem Supervised Learning. Hier wird das Modell mit Datensätzen trainiert, die jeweils sowohl den Input als auch den gewünschten Output enthalten. Die Aufgabe des Modells ist es anschließend, für unbekannte Datensätze anhand des Inputs deren Output zu bestimmen.

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

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

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

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

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

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \quad (2.3)$$

der als Informationsgrundlage zur Bestimmung von \hat{y} dient. Die Variable \hat{y} bezeichnet das Ergebnis der Funktion für einen gegebenen Input-Vektor X . Die Optimierung von $f(X|\Theta)$ bezieht sich hier auf die Suche jener Parameter Θ , welche die genaueste Näherung für die tatsächliche Funktion $f^*(X)$ und somit auch für die tatsächliche Klasse y erzielen (Alpaydin, 2008). Ist die Variable \hat{y} numerisch, so spricht man von Regression. Ein Beispiel für eine Regression ist die Vorhersage der Temperatur in Grad Celsius. Dabei enthält die Ergebnismenge der Funktion unendlich viele Elemente, zum Beispiel die reellen Zahlen. Ist die Variable \hat{y} kategorisch, so spricht man von Klassifikation. Im Spezialfall einer binären Klassifikation gilt zudem

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

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

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

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

berechnet. Für die Diff-Funktion gibt es zahlreiche Methoden, deren Eignung vom konkreten Anwendungsfall abhängt (Alpaydin, 2008). Die in dieser Arbeit verwendeten Methoden sind in Kapitel 3, "Methodik", erläutert. Das finale Modell verwendet anschließend jene Funktion, die die geringsten Abweichungen auf χ_{val} erreicht, also die beste Generalisierung aufweist. Um die erwartete Fehlerrate des finalen Modells zu bestimmen, werden die Abweichungen auf einem Test-Datenset χ_{test} – wie in Abschnitt 2.6 – herangezogen. Die Aufgabe von Machine Learning ist es also, jene Parameter Θ_{opt} zu finden, welche die Fehlerfunktion minimieren, also

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

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

Generalisierungsfehler führen. Wenn die Datenmenge ceteris paribus steigt, nimmt der Generalisierungsfehler ab, und vice versa (Alpaydin, 2008).

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

2.2 Klassifikation

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

Das vorangegangene Beispiel beinhaltet typische Elemente einer Klassifikation: Instanzen, Features und Klassen. Eine Instanz ist im Beispiel ein Kunde. Jede Instanz wird durch dieselben Attribute – aber womöglich mit unterschiedlichen Ausprägungen – beschrieben. Diese Attribute werden als Features, Φ , bezeichnet. Die Ausprägungen der Features für eine gegebene Instanz dienen als Input-Vektor X für die Klassifizierung, wie in Gleichung 2.1 dargestellt. Aufgrund der grundlegenden Bedeutung der Features stellen deren Berechnung (Feature Extraction) und Auswahl (Feature Selection) zentrale Schritte im Machine Learning dar, den das Kapitel 3 vertieft. Die möglichen kategorischen Werte der Ergebnis-Variable y , genannt Klassen, könnten im Beispieldfall "Kunde mit hohem Risiko" und "Kunde mit geringem Risiko" sein. Das Ergebnis einer Klassifikation ist die Klasse der betrachteten Instanz, im Beispiel die Kreditwürdigkeit eines Kunden.

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

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

Basierend auf dem Gehalt des Kunden und der Kreditsumme, schätzt das Modell das Risiko der Kreditvergabe ein. In der Realität reichen diese zwei Kriterien allerdings häufig nicht aus, um eine fundierte Entscheidung zu treffen. Bankangestellte prüfen ebenfalls die Kredithistorie, die Lebenssituation, den Arbeitsvertrag, den persönlichen Eindruck sowie weitere – möglicherweise auch die Persönlichkeit betreffende – Faktoren. Sofern diese Faktoren als Features in den Daten vorhanden sind, eignet sich der Klassifikator diese in der Trainingsphase an und ergänzt sie in das Entscheidungsmodell. Bis zu einem gewissen Punkt erhöht das Hinzufügen von Features und Regeln den Erfolg des Modells, wie in Abschnitt 2.7 dargestellt, bevor die zunehmende Komplexität die Generalisierungsfähigkeit vermindert.

Diese Bachelorarbeit beschränkt sich auf binäre Klassifikatoren, also Klassifikatoren mit genau zwei Klassen. Konkret werden Decision Tree und Random Forest Klassifikatoren sowie

einige Abwandlungen dergleichen untersucht. Es folgen nun die theoretischen Grundlagen der Erstellung, Anwendung und Erfolgsmessung dieser Modelle.

2.3 Decision Tree Klassifikator

2.3.1 Generelle Funktionsweise und Eigenschaften

Die Induktion von Decision Trees (Quinlan, 1986) gehört zu den simpelsten aber erfolgreichsten Machine Learning Algorithmen (Russell u. Norvig, 2009). Gupta u. a. (2017) nennen unter anderem die medizinische Diagnostik, intelligente Fahrzeuge, das Credit Scoring (siehe Abschnitt 2.2) und die industrielle Qualitätskontrolle als Anwendungsbereiche. Außerdem bilden Decision Trees die Grundlage für das später behandelte Random Forest Ensemble. Ein Decision Tree funktioniert nach dem Teile-und-Herrsche Prinzip. Die Trainings-Stichprobe χ_{training} wird nach einem festgelegten Kriterium in Teilmengen geteilt. Die entstandenen Teilmengen wiederum werden nach der gleichen Prozedur aufgeteilt. Diese Rekursion findet so lange statt, bis ein definiertes Endkriterium erfüllt ist und der Decision Tree zur Klassifikation bereit ist. Nachfolgend wird zuerst der Aufbau eines Decision Trees erklärt, sowie anschließend dessen Erstellung und Optimierung.

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

Ein Decision Tree besteht aus der Wurzel, internen Knoten, Ästen und externen Knoten, genannt Blätter. Die Wurzel ist der einzige interne Knoten, der keinen Vorgänger hat. Ein interner Knoten ist ein Knoten, der Nachfolger hat. Interne Knoten testen jeweils ein bestimmtes Feature Φ der betrachteten Instanz X auf dessen Wert. Äste sind die Verbindungen zwischen Knoten. Jeder Ast bildet eine Wertemenge ab, die im vorgelagerten internen Knoten festgelegt wurde. Ein Blatt ist ein Knoten ohne Nachfolger. Blätter repräsentieren, im Fall der Klassifikation, die Klassen (Quinlan, 1986). In dieser Arbeit werden binäre Decision Trees betrachtet, die genau zwei Klassen kennen. Auch finden ausschließlich univariate Bäume Anwendung, das heißt interne Knoten testen jeweils nur auf ein Feature, und nicht auf mehrere.

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

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

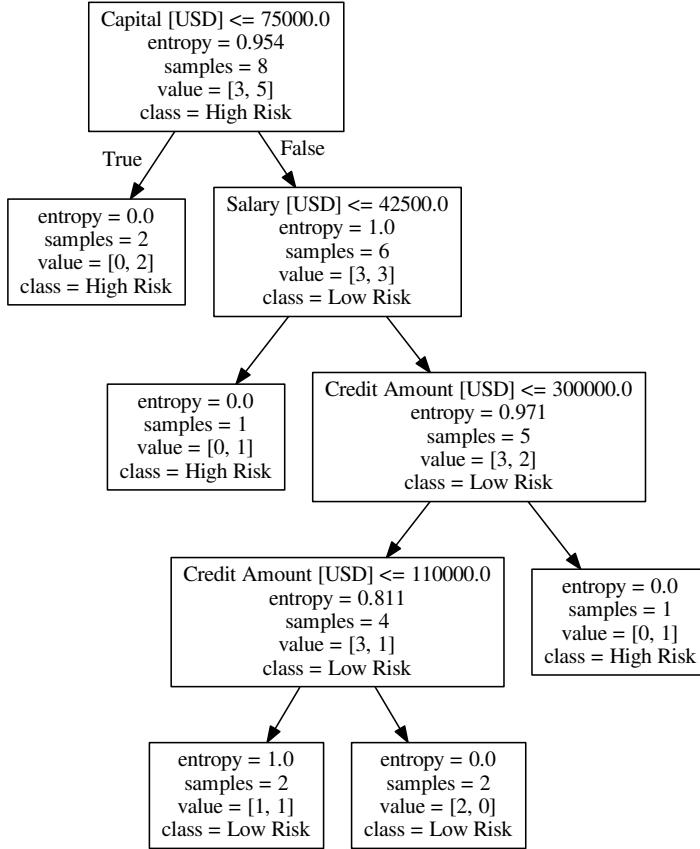


Abbildung 2.1 Beispielhafter Decision Tree für den Anwendungsfall der Kreditvergabe

alle Instanzen einem Blatt zugewiesen wurden und der Decision Tree fertig erstellt ist.

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

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

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

Im Gegensatz zu einigen statistischen Modellen wie der linearen oder exponentiellen Regression, gibt es im Funktionenraum F von Decision Trees stets eine Funktion $f(X|\Theta)$, die

die Klassen aller Trainings-Instanzen korrekt abbildet. Ein Decision Tree kann prinzipiell beliebig viele Regeln definieren und somit jede Instanz aus $\chi_{training}$ korrekt abbilden. Diese Eigenschaft bringt allerdings einen Nachteil mit sich. Decision Trees sind für Overfitting (siehe Abschnitt 2.7) anfällig. Denn falls der Decision Tree, ohne Kontrolle der Baumstruktur, für jede Instanz einen eigenen Blattknoten anlegt, erzielt er zwar auf $\chi_{training}$ ein optimales Ergebnis, ist jedoch nicht zur Generalisierung und somit nicht zum Lernen fähig. Entsprechende Gegenmaßnahmen, um die Baumstruktur zu kontrollieren, finden sich im Unterabschnitt 2.3.2.

2.3.2 Erstellung eines Decision Trees

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

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

$$v_{start} < s_1 < s_2 < v_{end}. \quad (2.9)$$

Die entstehenden Teilmengen wären dann

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

und würden zu drei entsprechenden Ästen führen. Der Spezialfall von genau einem Schwellenwert, wodurch sich zwei Teilmengen ergeben, wird als binäre Aufteilung bezeichnet (Alpaydin, 2008). Binäre Aufteilungen führen graphisch gesehen zu Hyperrechtecken, die die Instanzen voneinander abgrenzen.

Die Anzahl der möglichen Aufteilungen ist gleich der Anzahl der vorhandenen Features, m . Die möglichen Aufteilungen unterscheiden sich hinsichtlich ihres Einflusses auf den

finalen Decision Tree. Deshalb wird die Güte der möglichen Aufteilungen verglichen, um die sinnvollste Aufteilung zu identifizieren.

Als Kriterium für die Güte der Aufteilung verwendet der ID3-Algorithmus die aus ihr resultierende Änderung in Entropie, bezeichnet als Information Gain (*IG*) (Quinlan, 1986). Die Entropie H liegt per Definition zwischen Null und Eins und trifft eine Aussage über die Homogenität einer gegebenen Menge an n Instanzen. In der Informationstheorie bezeichnet die Entropie "die minimale Anzahl an Bits, die nötig sind, um die Klassifikationsgenauigkeit einer Instanz zu codieren" (Alpaydin, 2008). An jedem Knoten K wird die Entropie berechnet mit (Shannon, 1948)

$$H(K) = - \sum_w p_i \log_2(p_i), \quad (2.11)$$

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

$$IG(K_d, K_d|\phi_j) = H(K_d) - H(K_d|\phi_j). \quad (2.12)$$

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

$$H(K_d|\phi_j) = \sum_k P(\phi_j = v_k) H(K_d|\phi_j = v_k), \quad (2.13)$$

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

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

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

verglichen werden Gron (2017).

Russell u. Norvig (2009) weisen darauf hin, dass Features mit einem starken Verzweigungsfaktor nach dem ID3-Verfahren bevorzugt werden. Ein Beispiel dafür ist ein Zeitstempel, der für jede Instanz aus $\chi_{training}$ einzigartig ist. Dann würde eine Aufteilung anhand dieses Features stets zu einer erwarteten Entropie von Null führen, und würde somit stets den größtmöglichen IG erzielen. Der ID3-Algorithmus würde dieses Feature bevorzugen. Das würde zu starkem Overfitting führen, da das Modell nicht zur Generalisierung befähigt würde, also nicht lernt. Solchen stark-verzweigenden Features kann mit einer Bestrafung basierend auf dem Verzweigungsfaktor, zum Beispiel mithilfe des Gain Ratios, entgegengewirkt werden (Russell u. Norvig, 2009). Ein Nachfolger des ID3-Algorithmus namens C4.5 berücksichtigt den Gain Ratio bei der Erstellung des Decision Trees (Gupta u. a., 2017).

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

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

2.3.3 Grundlegende Parameter

Ein Machine Learning Algorithmus optimiert die Parameter Θ , um die Fehlerrate des Klassifikators zu minimieren (siehe Abschnitt 2.1). Die optimalen Werte, Θ_{opt} , resultieren also aus dem Trainingsprozess und sind erst im Nachhinein bekannt. Neben diesen Parametern gibt es Hyperparameter, die vom Anwender zu Beginn zu setzen sind. Einige dieser Hyperparameter beeinflussen die finalen Parameter Θ_{opt} , zum Beispiel indem sie den Wertebereich einschränken. Die Tabelle 2.1 orientiert sich an der Implementierung von Scikit-learn (Pedregosa u. a., 2011) und gibt einen Überblick über grundlegende Parameter (Typ in der Tabelle ist "P") sowie Hyperparameter (Typ in der Tabelle ist "H") von Decision Trees und deren Bedeutung.

2.3.4 Optimierung von Decision Trees

Nach Erstellung ist ein Decision Tree oft zu komplex und neigt zu Overfitting (Gron, 2017). Eine Methode zur Reduzierung von Overfitting ist das Pruning, also das Kürzen des Baumes. Es wird zwischen Prepruning und Postpruning unterschieden. Prepruning findet noch während der Erstellung des Decision Trees statt. Ein Ansatz für Prepruning ist es, eine Bedingung für Aufteilungen an jedem Knoten zu definieren. Dadurch soll verhindert werden, dass Entscheidungen, die auf zu wenigen Instanzen basieren, die Varianz des Klassifikators erhöhen und somit die Generalisierung verschlechtern (Alpaydin, 2008). So könnte man festlegen, dass eine weitere Aufteilung nur dann stattfindet, wenn mindestens fünf Prozent der Trainingsinstanzen aus $\chi_{training}$ zu diesem Knoten gelangt sind. Andernfalls wird dieser Knoten zu einem Blatt, betitelt mit der Mehrheitsklasse.

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

2.4 Ensemble Methoden

2.4.1 Die Ensemble-Idee

Nach dem No-free-lunch Theorem ist kein Lernalgorithmus besser als ein anderer, sofern die durchschnittliche Fehlerrate über alle möglichen diskreten Funktionen betrachtet wird (Whitley u. Watson, 2005). Es gibt also keinen einzigen Lernalgorithmus, der in jedem Anwendungsfall das genaueste Modell erzeugt (Alpaydin, 2008). Ist der beste Klassifikator in einer gegebenen Situation gesucht, so wird dieser beispielsweise mit einem Validierungs-Datenset χ_{val} ermittelt. Eine Alternative, die die Auswahl eines einzigen besten Klassifikators umgeht, ist das Ensemble Learning. Die Idee dabei ist es, die – nicht zu 100% ausmerzbaren – Fehler an einer Instanz X_i von einzelnen Klassifikatoren durch andere Klassifikatoren, die X_i korrekt abbilden, überzukompensieren (Russell u. Norvig, 2009). Die einzelnen Basisklassifikatoren verhalten sich dann komplementär zueinander. Das Ziel ist es, dadurch die Fehlerrate insgesamt im Ensemble zu reduzieren.

Ein Ensemble ist ein Klassifikator, der aus mindestens zwei individuellen Klassifikatoren besteht. Die individuellen Klassifikatoren werden als Basisklassifikatoren bezeichnet. Als Basisklassifikatoren eignen sich Modelle, die eine niedrigere Fehlerrate als zufälliges Raten erreichen. Ist dieses Kriterium erfüllt, so führt zum Beispiel das nachfolgend behandelte Boosting zu Ensembles mit "beliebig hoher Treffergenauigkeit" (Freund u. Schapire, 1997).

Um eine möglichst große Fehlerreduktion im Vergleich zu den einzelnen Basisklassifikatoren zu erreichen, ist es nötig, diese Basisklassifikatoren maximal unterschiedlich zu machen, sie also zu dekorrelieren. Zur Messung der Dekorrelation von Decision Trees schlägt Tin Kam Ho (1998) zum Beispiel die Metrik Tree Agreement vor. Für die Dekorrelation gibt es verschiedene Ansätze. So können zum Beispiel verschiedene Lernalgorithmen genutzt werden, dieselben Lernalgorithmen mit unterschiedlichen Hyperparametern belegt werden oder die Basisklassifikatoren mit unterschiedlichen Repräsentationen der Input-Daten trainiert werden (Alpaydin, 2008). Eine weitere Möglichkeit zur Dekorrelation der Basisklassifikatoren ist das Variieren der Trainingsdaten. Dazu gehören die Ansätze des Bagging und Boosting. Diese behandelt der nachfolgende Unterabschnitt 2.4.2.

2.4.2 Bagging und Boosting

Zwei verbreitete Methoden zur Dekorrelation von Basisklassifikatoren in Ensembles sind das Bagging und das Boosting. Beide Methoden können sowohl auf Klassifikatoren als auch auf Regressions-Modelle angewendet werden (Alpaydin, 2008). Bagging steht für Bootstrap Aggregation. Nach dieser Methode werden für jeden Basisklassifikatoren mit dem Bootstrap-Algorithmus zufällige Instanzen aus χ ausgewählt. Das heißt, der jeweilige Basisklassifikatoren β_i wird mit der Stichprobe $\chi_i \subset \chi$ trainiert, wobei $|\chi_i| = |\chi|$. Diese Auswahl erfolgt mit Wiederholung, eine Instanz aus χ kann also mehrmals in χ_i vorkommen oder auch gar nicht. Die Wahrscheinlichkeit, dass eine Instanz X nicht in die Bootstrap-Stichprobe χ_i kommt ist Alpaydin (2008)

$$P(X \in \chi_i) = \left(1 - \frac{1}{|\chi|}\right)^{|\chi|} \approx e^{-1} = 36,8\% \quad (2.14)$$

Das heißt im Umkehrschluss, dass χ_i ungefähr $100\% - 36,8\% = 63,2\%$ der Instanzen aus χ enthält. Im Gegensatz zum Boosting entstehen die Stichproben χ_i beim Bagging unabhängig voneinander.

Boosting bedeutet Verstärken. Dabei sind die Stichproben nicht unabhängig voneinander, sondern werden der Reihe nach gebildet. Die Idee von Boosting ist es, die Stichprobe χ_{i+1} dahingehend anzupassen, dass sie jene Instanzen aus χ_i bevorzugt, welche von den darauf trainierten Basisklassifikatoren nicht erfolgreich gelernt wurden. Der Basisklassifikatoren β_{i+1} soll also von den Fehlern seines Vorgängers β_i lernen. Somit sollen die Basisklassifikatoren ihre Schwächen untereinander ausgleichen. Um den Bedarf nach einer sehr großen Datenmenge zum Boosting zu beseitigen, entwarfen Freund u. Schapire (1997) die Variante AdaBoost, oder ausgeschrieben Adaptive Boosting. Außerdem ist AdaBoost im Gegensatz zu seinen Vorgängern dazu in der Lage, beliebig viele Basisklassifikatoren zu kombinieren (Alpaydin, 2008).

2.4.3 Entscheidungsfindung im Ensemble

Die Entscheidung, welche Klasse einer neuen Instanz X zugewiesen wird, trifft ein Ensemble auf Basis der Entscheidungen seiner Basisklassifikatoren. Ein Ensemble E , bestehend aus den n Basisklassifikatoren $\{\beta_1, \dots, \beta_n\}$ unter Verwendung von Kombinationsmethode Ψ , kann als

Vorhersagefunktion für Instanz X als

$$E(X|\{\beta_1, \dots, \beta_n\}, \Psi) = \hat{y} \quad (2.15)$$

formalisiert werden. Für Ψ lassen sich einstufige und mehrstufige Kombinationsmethoden unterscheiden (Alpaydin, 2008). Diese Arbeit beschränkt sich auf einstufige Methoden. Eine einstufige Methode ist die nachfolgend beschriebene Voting-Methode.

Wenn $\{\hat{y}_1, \dots, \hat{y}_n\}$ die Ergebnisse von $\{\beta_1, \dots, \beta_n\}$ sind, dann ergibt sich nach der Voting-Methode Ψ_{voting} die Entscheidung des Ensembles \hat{y}_E allgemein (im Sinne von sowohl für Klassifikation als auch für Regression gültig) mit

$$\hat{y}_E = \Psi_{voting}(\hat{y}_1, \dots, \hat{y}_n). \quad (2.16)$$

Handelt es sich bei $\{\hat{y}_1, \dots, \hat{y}_n\}$ um die vorhergesagten Klassen, so nennt man dies Hard Voting. Handelt es sich dabei jedoch um die Wahrscheinlichkeiten für die positive Klasse, so spricht man von Soft Voting (Gron, 2017). Im Spezialfall, dass das Ensemble als Klassifikator verwendet wird, entspricht das finale Ergebnis der nächstgelegenen – gemessen an der Distanz zu \hat{y}_E – Klasse. Das heißt in einer binären Klassifikation mit den Klassen $y \in \{0, 1\}$ ergibt sich das Ergebnis \hat{y}_{Class} mit

$$\hat{y}_{Class} = \begin{cases} 1(\text{positiv}), & \text{wenn } 0 \leq \hat{y}_E < 0,5 \\ 0(\text{negativ}), & \text{wenn } 0,5 \leq \hat{y}_E \leq 1. \end{cases} \quad (2.17)$$

Die Annahme bezüglich Ψ_{voting} ist, dass alle Basisklassifikatoren $\{\beta_1, \dots, \beta_n\}$ gleich gewichtet sind, und zwar mit einem Gewicht von $\frac{1}{n}$. In der Literatur finden sich Vorschläge, wie man die Basisklassifikatoren gewichten kann. Ein solches Vorgehen ist es, die Basisklassifikatoren auf einem Validierungs-Datensatz χ_{val} zu bewerten, um sie im Ensemble dann anhand ihrer Treffgenauigkeit zu gewichten (Alpaydin, 2008).

An dieser Stelle sei, alternativ zur Voting-Methode, die geschachtelte Generalisierung von Wolpert (1992) erwähnt. Dessen Idee ist es, die Kombination von $\hat{y}_1, \dots, \hat{y}_n$ nicht durch eine (gewichtete) Summe zu vollziehen, sondern selbst wiederum durch eine lernende Funktion Ψ_{sg} zu entwerfen. Das finale Ergebnis eines Ensembles, das eine solche Kombinationsfunktion verwendet, lautet dann

$$\hat{y}_E = \Psi_{sg}(\hat{y}_1, \dots, \hat{y}_n | \Theta). \quad (2.18)$$

Ein Lernalgorithmus sucht die optimalen Parameter Θ_{opt} , sodass die Fehlerfunktion minimiert wird, wie in Abschnitt 2.1 beschrieben. Die Kombinationsfunktion Ψ_{sg} soll also die Schwächen der einzelnen Basisklassifikatoren lernen, um diese bei der finalen Kombination zu berücksichtigen (Alpaydin, 2008). Da nun sowohl die Base Classifier, die $\{\hat{y}_1, \dots, \hat{y}_n\}$ erzeugen, als auch die Kombinationsfunktion Ψ_{sg} selbst anhand von Daten lernen, findet eine geschachtelte Generalisierung statt.

2.5 Random Forest Klassifikator

2.5.1 Generelle Funktionsweise und Eigenschaften

Ein Random Forest ist ein Ensemble, das Decision Trees als Basisklassifikatoren nutzt. Trotz ihrer vergleichsweise simplen Struktur gehören Random Forests zu den mächtigsten Machine

Learning Algorithmen (Gron, 2017). Durch das Einführen von Zufälligkeit ist ein Random Forest robuster als ein einzelner Decision Tree. Die Zufälligkeit entsteht durch Bagging und durch zufällige Auswahl jener Attribute, die für einen Split überhaupt erst betrachtet werden. Diese Zufälligkeit hilft dabei, unkorrelierte Decision Trees zu generieren, die ihre Schwächen gegenseitig ausgleichen und somit komplementär zueinander sind.

Die Klassifikation durch einen Random Forest erfolgt nach der Voting-Methode (siehe Unterabschnitt 2.4.3). Die Decision Trees aus dem Random Forest klassifizieren die Instanz nach dem in Abschnitt 2.3.1 dargelegten Procedere. Die Ergebnisse fließen als gleich-gewichtete Stimmen in den Random Forest ein. Die am häufigsten vorkommende Klasse ist das Ergebnis der Klassifizierung.

Der Funktionenraum F eines Random Forests enthält – wie auch der von Decision Trees – stets eine Funktion f , die die Klassen aller Trainings-Instanzen korrekt abbildet. Der entscheidende Unterschied zu Decision Trees jedoch ist die Lösung des Overfitting-Problems. Mithilfe des Gesetzes der großen Zahlen hat Breiman (2001) bewiesen, dass bei Random Forests kein Overfitting auftritt. Der Generalisierungsfehler eines Random Forests hängt von der Güte der einzelnen Decision Trees ab als auch von der Korrelation dergleichen ab (Breiman, 2001).

Eine vorteilhafte Eigenschaft ergibt sich aus der Tatsache, dass der Random Forest Bagging verwendet. Wie in Kapitel 2.4.2 gezeigt, enthalten die Stichproben, auf denen die Basisklassifikatoren trainiert werden, durchschnittlich 63,2% der Instanzen aus χ . Die restlichen 36,8% der Instanzen aus χ werden als out-of-bag (OOB) bezeichnet. Diese OOB Instanzen können zur Erfolgsmessung verwendet werden, noch bevor ein Validierungs- oder Test-Datenset zum Einsatz kommt. Die gemessene Fehlerrate nennt sich in dem Fall OOB Error (Gron, 2017).

2.5.2 Erstellung eines Random Forests

Ein Random Forest entsteht durch das Trainieren von n Decision Trees auf den Trainingssets $\{\chi_1, \chi_n\}$, welche mittels Bagging (siehe Kapitel 2.4.2) aus χ generiert werden.

Eine Besonderheit dabei ist, dass nicht alle Features für die Aufteilung an Knoten K betrachtet werden. Stattdessen erwägt der Random Forest Algorithmus eine zufällige (engl. random), echte Untermenge der Features. Von den i Attributen der Daten wird eine Untermenge von j Attributen zufällig ausgewählt. Für diese j Attribute werden im nächsten Schritt jeweils die Information Gains berechnet, die aus ihren Aufteilungen (siehe Unterabschnitt 2.3.2) resultieren würden. Das Attribut mit dem höchsten Information Gain wird für den betrachteten Knoten als Aufteilungskriterium gewählt. Diese Zufälligkeit dekorreliert die Basisklassifikatoren, denn diese treffen ihre Entscheidungen basierend auf verschiedenen Features, und ermöglicht dadurch ein komplementäres Ensemble. Algorithmus 1 veranschaulicht die Erstellung eines Random Forests mittels Pseudocode.

Algorithm 1 Random Forest Pseudocode

Require: Trainingset $\chi_{training}$, Features Φ , Anzahl der Decision Trees n

```
1: function RandomForest ( $\chi_{random\_forest}$   $\Phi_{random\_forest}$ ,  $n$ )
2:    $T \leftarrow \emptyset$ 
3:   for  $i \in \{1, \dots, n\}$  do
4:      $\chi_i \leftarrow$  Bootstrapping-Set von  $\chi_{random\_forest}$ 
5:      $\Phi_i \leftarrow$  Randomisierte, echte Teilmenge von  $\Phi_{random\_forest}$ 
6:      $t_i \leftarrow$  DecisionTree( $\chi_i, \Phi_i$ )
7:      $T \cup t_i$ 
8:   end for
9:   return  $T$ 
10:  end function

11: function DecisionTree( $\chi_{decision\_tree}$   $\Phi_{decision\_tree}$ )
12:    $K \leftarrow$  Wurzel-Knoten  $k_1$ 
13:   while  $K$  enthält mindestens einen internen Knoten  $k_{intern\_neu}$  ohne Nachfolger do
14:      $\Phi_{opt} \leftarrow$  Feature mit höchstem Information Gain an Knoten  $k_{intern}$ 
15:     Teile Instanzen an  $k_{intern\_neu}$  entsprechend der Werte von  $\Phi_{opt}$  auf neue Knoten  $K_{neu}$  auf
16:     Weise  $k_{intern\_neu}$  alle Knoten aus  $K_{neu}$  als Nachfolger zu
17:     for  $K_{neu\_i} \in K_{neu}$  do
18:       if Instanzen an Knoten  $K_{neu\_i}$  haben alle dieselbe Klasse then
19:         Erkläre  $K_{neu\_i}$  als Blatt mit Mehrheitsklasse
20:       else
21:         Erkläre  $K_{neu\_i}$  als internen Knoten ohne Nachfolger
22:       end if
23:     end for
24:      $K \cup K_{neu}$ 
25:   end while
26:   return  $K$ 
27: end function
```

2.5.3 Grundlegende (Hyper-)Parameter

Zusätzlich zu den (Hyper-)Parametern seiner Basisklassifikatoren (siehe 2.3.3) besitzt der Random Forest als Ensemble noch weitere Einstellungen. Die Tabelle 2.2 orientiert sich an der Implementierung von Scikit-learn (Pedregosa u. a., 2011) und gibt einen Überblick über alle relevanten (Hyper-)Parameter.

Aufbauend auf der ursprünglichen Idee von Breiman (2001) entwickelten Forscher neue Random Forest Modelle mit dem Ziel, die Fehlerrate weiter zu reduzieren. Ebenso optimieren manche Ansätze den Random Forest für bestimmte Anwendungsbereiche, oder reduzieren die Trainings- und Klassifikationszeit. Diese Arbeit stellt einige dieser weitergehenden Random Forest Variationen im Unterabschnitt ?? vor.

Tabelle 2.2 (Hyper-)Parameter eines Random Forests und deren Bedeutung

Name	Typ	Beschreibung	Zweck (Beispiele)
Anzahl der Basisklassifikatoren	H	Anzahl der Decision Trees innerhalb des Random Forests	Steigerung der Treffergenauigkeit durch mehr komplementäre Basisklassifikatoren
Bootstrap-Sampling	H	Bestimmung, ob Basisklassifikatoren auf den gesamten Daten oder auf Bootstrap-Stichproben trainiert werden	Steigerung der Zufälligkeit bei Training der Basisklassifikatoren durch Veränderung der Trainingsstichproben
OOB Treffergenauigkeit	P	Schätzung der Treffergenauigkeit, gemessen an den out-of-bag Instanzen der jeweiligen Decision Trees	Beurteilung der Treffergenauigkeit noch vor Evaluierung auf einem Validierungs- oder Testdatenset
Alle (Hyper-) Parameter von Decision Trees	H/P	Die (Hyper-)Parameter der Basisklassifikatoren werden über den Random Forest zentral definiert	Steigerung der Trefferrate des Random Forests durch Optimierung der Hyperparameter seiner Decision Trees

2.6 Erfolgsmessung von Klassifikatoren

Nach der Erstellung eines oder mehrerer Klassifikatoren ist häufig die Güte des Modells von Interesse, beispielsweise gemessen an der Fehlerrate. Fehlerraten werden unter anderem verglichen, um den geeigneten Klassifikator für einen gegebenen Anwendungsfall zu bestimmen. Außerdem ist die erwartete Fehlerrate auf neuen Daten außerhalb $\chi_{training}$ häufig von Interesse. Das ist insbesondere der Fall, wenn es für die Fehlerrate eine – selbstaufgerlegte oder fremdbestimmte – harte Obergrenze von $p\%$ gibt, die nachgewiesen werden muss, bevor der Klassifikator in der Realität angewendet werden darf.

Das Training, die Validierung und das Testen erfolgen jeweils auf den separaten Datensets $\chi_{training}$, $\chi_{validierung}$ und χ_{test} , wobei gilt

$$\chi_{training} \cap \chi_{validierung} = \emptyset; \chi_{validierung} \cap \chi_{test} = \emptyset \text{ und } \chi_{training} \cap \chi_{test} = \emptyset. \quad (2.19)$$

$\chi_{training}$ dient der Optimierung der Parameter eines Klassifikators, $\chi_{validierung}$ dem Tuning von Hyperparametern und χ_{test} der Erhebung des erwarteten Fehlers auf neuen Daten. Jedes dieser Datensets soll die Grundgesamtheit, aus der die Stichproben stammen, repräsentativ darstellen. Alpaydin (2008) schlägt vor, ein Drittel der vorhandenen Daten für χ_{test} zu verwenden und die anderen zwei Drittel auf $\chi_{training}$ und $\chi_{validierung}$ zu verteilen. Für diese Einteilung zwischen $\chi_{training}$ und $\chi_{validierung}$ gibt es mehrere Varianten. Eine davon ist die Kreuzvalidierung.

In der Kreuzvalidierung werden aus χ k Stichproben, $\{\chi_{-1}, \dots, \chi_k\}$ mit $|\chi_{-1}| = \dots = |\chi_{-k}|$ generiert. Dann erfolgen k Durchläufe, genannt Folds, wobei der Klassifikator in $Fold_i$ auf

$\chi_1 \cup \dots \cup \chi_{i-1} \cup \chi_i + 1 \cup \dots \cup \chi_k$ trainiert wird und auf χ_i getestet wird. Alpaydin (2008) merkt an, dass für jede Teilstichprobe χ_i die relative Häufigkeit jeder Klasse y gleich sein sollte, und, dass diese wiederum den relativen Häufigkeiten von y in χ gleichen sollten. Dieser Vorgang nennt sich Stratifizierung. Ein Spezialfall der Kreuzvalidierung ist die Leave-One-Out Methode Alpaydin (2008). Dabei wird χ in $k = n$ Stichproben aufgeteilt, mit $n = |\chi_{\text{training}} \cup \chi_{\text{validierung}}|$. Jede Trainingsstichprobe χ_i besteht also aus einer einzigen Instanz. In diesem Spezialfall ist Stratifizierung nicht möglich. Die Leave-One-Out Methode wird zum Beispiel in der medizinischen Diagnostik verwendet, wenn nur sehr wenige relevante Daten vorhanden sind Alpaydin (2008). Eine weitere Variante zur Einteilung der Datensets χ_{training} , $\chi_{\text{validierung}}$ und χ_{test} ist die Bootstrap Methode. Diese wurde bereits in Kapitel 2.4.2 behandelt.

Um die Güte eines Klassifikators zu bestimmen und mit anderen zu vergleichen, bieten sich verschiedene Kennzahlen an. Jede Klassifizierung fällt in eines der vier Felder aus der Konfusionsmatrix in Tabelle 2.3.

Tabelle 2.3 Konfusionsmatrix

		\hat{y} (Ergebnis der Klassifikation)	
		1	0
y	1	WP (Wahre Positive)	FN (Falsche Negative)
	0	FP (Falsche Positive)	WN (Wahre Negative)

Die Fehlerrate E berechnet sich dann mit

$$E = \frac{|FP| + |FN|}{N}, \quad (2.20)$$

wobei $N = |WP| + |FP| + |FN| + |WN|$.

Im Falle von verzerrten Datensets, wenn die Klassen sehr ungleich verteilt sind, eignet sich diese simple Fehlerrate alleine nicht zur Erfolgsmessung; stattdessen bieten sich weitergehende Metriken aus der Konfusionsmatrix an (Gron, 2017). Precision misst den Erfolg des Klassifikators auf den positiven Instanzen und ergibt sich aus

$$\text{Precision} = WP / (WP + FP). \quad (2.21)$$

Precision alleine reicht jedoch nicht aus. Ein Klassifikator könnte von n wahren Positiven nur einen einzigen, sicheren als positiv bestimmen, wobei $1 << n$. Damit wäre die Precision $\frac{1}{(1+0)} = 100\%$, jedoch zu Lasten von $n - 1$ falschen Negativen. Um diesen Trade-Off zu erkennen, bietet sich der Recall als weitere Metrik an. Dieser folgt aus

$$\text{Recall} = WP / (WP + FN). \quad (2.22)$$

Das F-Maß kombiniert schließlich Precision und Recall als harmonischer Durchschnitt (Gron, 2017)

$$F - \text{Maß} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = 2x \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (2.23)$$

Das F-Maß bestraft zwar Klassifikatoren, die sehr ungleiche Werte für Precision und Recall erzielen. Gleichzeitig bevorzugt es jedoch Klassifikatoren, die für Precision und Recall ähnliche Werte aufweisen. Das ist nicht immer erstrebenswert. Je nach Anwendungsfall kann entweder Precision oder Recall relevanter sein (Russell u. Norvig, 2009). Handelt es sich um die Klassifikation zur Bombendetektion, so würde man vermutlich Fehlalarme einer entgangenen Bombe vorziehen. Dann wäre Recall wichtiger als Precision. Allgemein geht eine höhere Precision mit niedrigerem Recall einher, und vice versa (Gron, 2017). Dieser Precision-Recall Trade-Off ist eine zentrale Fragestellung im Machine Learning und ist für jeden Anwendungsfall einzeln zu lösen.

Es sei erwähnt, dass die Konfusionsmatrix nicht das einzige Kriterium zur Erfolgsmesung eines Klassifikators sein sollte. Turney (2002) ergänzt zum Beispiel die Speicher- und Laufzeit-Komplexität sowie die Interpretierbarkeit als weitere Bewertungskriterien, die in der Forschung noch nicht genug Beachtung gefunden hätten.

2.7 Underfitting, Overfitting und der Curse of Dimensionality

Um die bestmögliche Generalisierung zu erreichen, vergleicht man die Komplexität der erlernten Funktion f mit der den Daten zugrundeliegenden Funktion (Alpaydin, 2008). Ist die Komplexität von f niedriger als die der approximierten Funktion, wird dies als Underfitting bezeichnet. Dies ist zum Beispiel der Fall, wenn man versucht, eine Gerade auf einen Datensatz anzupassen, der von einem Polynom dritten Grades stammt (Alpaydin, 2008). Die Erhöhung der Komplexität führt bei Underfitting zu einer Verbesserung der Vorhersagegenauigkeit beziehungsweise zu einer Reduktion des Vorhersagefehlers. Ist die Komplexität von f höher als die der approximierten Funktion, so wird dies als Overfitting bezeichnet. Ein Beispiel ist der oben erwähnte, nicht-kontrollierte Decision Tree, der für jede Instanz einen eigenen Blattknoten anlegt. Es folgt die Definition von Overfitting. Dabei bezeichnet $E(\cdot)$ die Fehlerfunktion. Angenommen, der Lernalgorithmus zieht die Funktion $f_1(X)$ aus F einer Funktion $f_2(X)$ vor, weil auf χ_{training} gilt:

$$E(f_1(x)) < E(f_2(x)). \quad (2.24)$$

Gleichzeitig aber gilt auf der gesamten Verteilung, aus der χ_{training} stammt:

$$E(f_1(X)) > E(f_2(X)). \quad (2.25)$$

Dann wird die Funktion $f_1(X)$ als overfitted bezeichnet (Mitchell, 1997).

Was genau die Komplexität ausmacht, hängt vom betrachteten Klassifikator ab. In Decision Trees leitet sich die Komplexität aus der Anzahl an Knoten und Blättern ab. Eine große Anzahl an Knoten und Blättern bedeutet hohe Komplexität, was wiederum zu Overfitting führen kann, und vice versa. Für Decision Tree Ensembles, wie dem Random Forest, kommt zusätzlich noch die Anzahl der einzelnen Basisklassifikatoren als Faktor für die Komplexität hinzu. Hierbei ist anzumerken, dass eine größere Anzahl an Basisklassifikatoren aber nicht unbedingt zu Overfitting führt. Random Forests profitieren – im Widerspruch zu Ockhams Rasiermesser – von einer steigenden Anzahl an Basisklassifikatoren, solange diese hinreichend unabhängig voneinander sind (Gron, 2017).

Eine weitere, allgemeine Herausforderung – unabhängig vom konkreten Anwendungsfall – im Machine Learning ist der Curse of Dimensionality. Dieses Phänomen bezeichnet die exponentielle Steigung der benötigten Features, die bei Erhöhung der Dimensionen nötig ist (Verleysen u. François, 2005). Wenn beispielsweise ein Klassifikator in einer Dimension mit 10 Instanzen trainiert wird, so sind in zwei Dimensionen 100 und in drei Dimensionen 1.000 Trainings-Instanzen nötig, um den gleichen Lernerfolg zu erzielen (Verleysen u. François, 2005). Obwohl Random Forests dazu in der Lage sind, den Curse of Dimensionality zu reduzieren, sind diese dennoch von einer steigenden Anzahl an Features negativ betroffen, da sich die Trainingszeit verlängert (Li, 2016). Li (2016) schlug einen mittels Hadoop MapReduce parallelisierten Random Forest vor und zeigte, dass dieser verglichen mit einem nicht-parallelisierten Random Forest ein besseres Ergebnis sowie eine um 40% verkürzte Rechenzeit erreicht.

2.8 Besonderheiten bei der Klassifikation von Zeitreihen

Für den Spezialfall der Zeitreihen-Klassifikation bieten sich angepasste Methoden für die Messung der Fehlerrate an. Es ist zum Beispiel möglich, dass zwischen den Instanzen aus χ zeitliche Abhängigkeiten bestehen. So könnte in einer Stichprobe über die Jahre j_1 bis j_n ein Muster ab Jahr j_s auftreten, wobei $j_1 < j_s < j_n$. Ein Beispiel ist die Einführung eines neuen Gesetzes in Jahr j_s , das die betrachtete Zeitreihe ab dann beeinflusst. In diesem Fall sollte der Klassifikator das Muster nur auf Instanzen aus den Jahren $[j_s, j_n]$, nicht jedoch aus der Vorzeit $[j_1, j_s]$ anwenden.

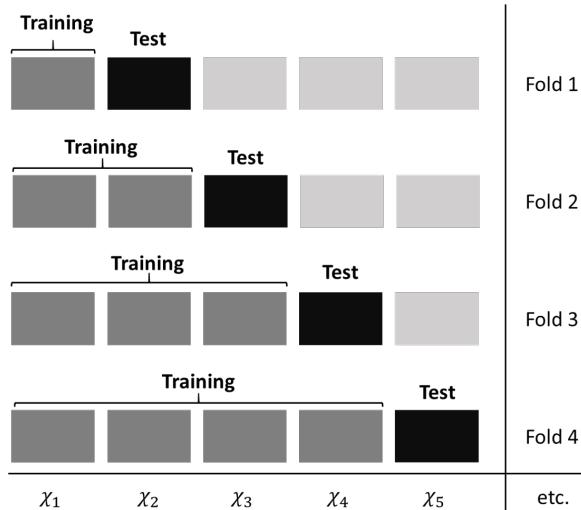


Abbildung 2.2 Schematischer Ablauf einer Time Series Cross-Validation in den ersten vier Iterationen

Ein Ansatz, um zeitlich-conditionierte Muster zu entdecken, ist die Blocked Form Cross-Validation, auch bekannt als Time Series Cross-Validation. Bergmeir u. Benitez (2011) haben empirisch ermittelt, dass diese Methode für Zeitreihen-Klassifikatoren genauere Ergebnisse erzielt als herkömmliche Cross-Validation Varianten wie die erwähnte Kreuzvalidierung. Der schematische Ablauf ist in Abbildung 2.2 dargestellt. Ausgehend von der Trainings-Stichprobe $\chi_{training}$ bildet die Time Series Cross-Validation k Teilstichproben, $\{\chi_1, \dots, \chi_k\}$,

wobei $|\chi_{-1}| = \dots = |\chi_{-k}|$. Dann erfolgen $i = k - 1$ Epochen, wobei der Klassifikator in $Epoche_i$ jeweils auf $\chi_1 \cup \dots \cup \chi_i$ trainiert wird und auf χ_{i+1} getestet wird. Alternativ kann $\chi_i \cup \dots \cup \chi_k$ als Testset verwendet werden. Eine weitere Variation ist es, nicht auf den Daten aller vorangegangener Zeitabschnitte, sondern lediglich auf jenen des letzten Zeitabschnittes, also auf χ_i , zu trainieren (Cerqueira u. a., 2019).

2.9 Random Walk Theorie und Markteffizienzhypothese

Da sich diese Arbeit mit Anwendungsfällen aus der Finanzdomäne beschäftigt, folgen zwei grundlegende Theorien aus diesem Gebiet. Nach der Random Walk Theorie (Fama, 1965) sind sämtliche Analysen zur Vorhersage von Aktienkursen wertlos. Das gilt sowohl für die technische Analyse, die annimmt, dass es in Aktienkursen sich wiederholende Muster gibt, als auch für die Fundamentalanalyse, die den intrinsischen Wert eines Unternehmens anhand von Finanzkennzahlen bewertet. Die technische Analyse bezieht sich ausschließlich auf die Zeitreihen eines Wertpapiers. Kennzahlen über die prozentuale oder absolute Veränderung des Kurses sowie statistische Werte bilden dabei die Grundlage für Kauf- und Verkaufentscheidungen. Ein Beispiel aus der technischen Analyse ist der gleitende Durchschnitt der letzten d Tage, wobei d beliebig variiert wird. Die Fundamentalanalyse hingegen basiert auf den Finanzkennzahlen des Emittenten des Wertpapiers, und nicht auf dem Kurs des Wertpapiers. Beispiele für solche Kennzahlen sind der Umsatz, der Jahresüberschuss oder auch die Eigenkapitalrendite des Unternehmens. Ebenso betrachtet die Fundamentalanalyse ökonomische Faktoren wie den Leitzins, gesellschaftliche Trends oder das Management des Emittenten.

Analysen basieren stets auf Informationen, die den Marktteilnehmern bekannt sind. Nach der Markteffizienzhypothese sind all diese Informationen jedoch bereits im Preis des Wertpapiers abgebildet. Dann kann die Analyse der Informationen keine überproportionalen Renditen ermöglichen. Nach Fama (1965) ist ein effizienter Markt ein Markt, in dem sehr viele rationale, Gewinn-maximierende Akteure konkurrieren, wobei jeder von diesen Zugriff auf alle Informationen hat und versucht, die Kursbewegungen gewinnbringend vorherzusagen. Somit spiegelt der Preis eines Wertpapiers in einem effizienten Markt alle vorhandenen Informationen wieder. Sowohl Ereignisse aus der Vergangenheit als auch zukünftig erwartete Ereignisse, über die sich die Marktteakteure einig sind, bestimmen die Preisbildung. Der Marktpreis stimmt also mit dem inneren Wert des Wertpapiers überein. In einem solchen Markt "hat eine Zeitreihe von Aktienkursen kein Gedächtnis" (Fama, 1965); es ist nicht möglich von historischen Beobachtungen auf neue Kursbewegungen zu schließen. Erkennbare, gewinnbringende Muster würden von der Masse der rationalen Akteure sofort ausgenutzt, bis sie im Marktpreis berücksichtigt und somit nutzlos wären (Lendasse u. a., 2002). Also können nur neue Informationen zu einer Änderung des inneren Preises eines Wertpapiers führen. Und neue Informationen können per Definition nicht vorhergesagt werden (Lendasse u. a., 2002).

In der Realität treffen einige Annahmen der Markteffizienzhypothese allerdings nicht immer zu. Zum einen handeln die Akteure am Aktienmarkt, also Käufer und Verkäufer, nicht immer rational. Auch wenn Computer einen wachsenden Teil der Aktienkäufe und -Verkäufe tätigen, sind emotional geleitete Menschen am Markt aktiv. Auch hat nicht

jeder Marktteilnehmer die gleichen Informationen. Zum Beispiel nutzen die Akteure unterschiedliche Datenquellen, die Informationen verzerrn oder zu verschiedenen Zeitpunkten veröffentlichen. Selbst wenn die Informationsbasis aller Akteure gleich wäre, würde jeder von diesen – aufgrund einmaliger Erfahrungen und Fähigkeiten eines jeden Menschen – zu unterschiedlichen Ergebnissen kommen.

Weiterhin ist es möglich, dass die handelnden Computerprogramme selbst, durch ihre automatischen Kauf- und Verkaufsaktionen bei Unter- oder Überschreitung gewisser Preispunkte, Muster in den Aktienkursen erzeugen. Dann könnte ein Klassifikator, wie der Decision Tree, eben diese Muster erkennen und Regeln bilden, um sie systematisch auszunutzen.

Diese über fünfzig Jahre alten Theorien wurden in jüngerer Zeit kritisch hinterfragt. Vor dem Hintergrund steigender Rechenleistung von Computern und gehäufter Erfolge im Machine Learning hat man mit verschiedenen Klassifikatoren versucht, signifikant höhere Renditen zu erzielen als der Marktdurchschnitt und die Random Walk Theorie zu widerlegen. Ausgewählte Arbeiten zu diesem Thema finden sich in Unterabschnitt ??.

2.10 Relevante Literatur

In den letzten Jahren haben Forscher die Anwendung von Machine Learning und speziell von Random Forest Modellen zur Vorhersage von Aktienkursen unter verschiedenen Ansätzen untersucht. Es folgt ein Überblick relevanter Veröffentlichungen mit deren Ansätzen, Vorgehen und Ergebnissen. Dabei werden auch bisher nicht- oder unzureichend erforschte Fragen identifiziert.

Sadia u. a. (2019) haben Random Forests und Support Vector Machines verglichen und deren Vorhersagegenauigkeiten von Aktienkursen auf einem Datensatz von Kaggle gemessen. Insgesamt lagen 121.608 Datensätze verschiedener Aktien vor, die bereinigt und im Verhältnis 80% zu 20% in Trainings- und Testset aufgeteilt wurden. Das Trainingsset wiederum wurde mittels Cross-Validation aufgeteilt, um die Hyperparameter zu optimieren. Mit einer Genauigkeit von 0,808 hat der Random Forest ein besseres Ergebnis erzielt als die State Vector Machine mit 0,787. Da die Chronologie der Daten bei der Cross-Validation nicht eingehalten wird, sind mögliche zeitliche Abhängigkeiten nicht berücksichtigt worden. Außerdem ist die 80% zu 20% Aufteilung nur einmalig erfolgt. Im Gegensatz zu Cross-Validation liegt hier eine lokale Verzerrung vor, da kein Mittelwert über verschiedene Aufteilungen berechnet wird. Die Anwendung der Time Series Cross-Validation reduziert die beiden genannten Schwächen und kann somit zu aussagekräftigeren Ergebnissen führen.

Alavi u. a. (2015) haben Random Forests mit State Vector Machines und K-Nearest Neighbor Modellen verglichen. Als Datengrundlage dienten Zeitreihen iranischer Aktien im Zeitraum von 2002 bis 2012. Als zusätzliche Features, neben den Aktienkursen selbst, wurden Indikatoren aus der technischen Aktienanalyse berechnet. Die Untersuchungen wurden in MATLAB durchgeführt. Nach dem Hyperparameter-Tuning erreichte der Random Forest mit 0,919 das beste F-Maß, gefolgt von State Vector Machines (0,860) und K-Nearest Neighbors (0,820). Zur Erfolgsmessung haben die Autoren eine einmalige Aufteilung in Trainings- und Testset vorgenommen; ein auf Zeitreihen spezialisiertes Verfahren hat nicht

stattgefunden. Auch die Auswirkungen der zusätzlichen technischen Features wurde nicht eruiert.

Pasupulety u. a. (2019) haben einen Baum-basierten Klassifikator mit Features zur Erfassung der öffentlichen Wahrnehmung einer Aktie angereichert, um Aktienkurse des indischen IT-Unternehmens Infosys Limited vorherzusagen. Dabei konnten sie jedoch nur eine vernachlässigbare Verbesserung der Genauigkeit feststellen. Die Umsetzung erfolgte mit der Scikit-learn Bibliothek in Python. Die Erfolgsmessung fand nach der Time Series Cross-Validation statt, um zeitliche Abhängigkeiten in den Daten zu berücksichtigen.

Eine Alternative zur binären Klassifikation auf den Aktienmärkten, mit den Klassen Äktie steigt und Äktie fällt, schlugen Lohrmann u. Luukka (2019) vor. Mit den vier Klassen stark positiv; schwach positiv; schwach negativ und stark negativ konnten sie bessere Ergebnisse erzielen, als mit zwei Klassen. Aktienkurse zwischen 2010 und 2018 von der Yahoo Finance API waren dabei die Datengrundlage. Der Random Forest Klassifikator erreichte in den MATLAB-Simulationen die höchste Genauigkeit. Eine Untersuchung verschiedener Zeithorizonte fand nicht statt.

Khaidem u. a. (2016) haben bei der Klassifikation nach Zeithorizont unterschieden und erschlossen, dass längere Zeithorizonte zu einer höheren Treffergenauigkeit führen. Für Datensätze von Apple, Microsoft und Samsung berichten die Autoren Treffergenauigkeiten zwischen 85% und 95% für einen Horizont von zehn Tagen, was eine Steigerung gegenüber der Bezugsliteratur (Di, 2014) darstellt, in der Werte zwischen 70% und 80% erreicht wurden. Beide Veröffentlichungen . Die Vorhersagegenauigkeit von Machine Learning Modellen kann abhängig vom Zeithorizont stark schwanken. So ist es möglich, dass es für die Vorhersagen ein Jahr in die Zukunft erkennbare Muster gibt, während Vorhersagen einen Tag in die Zukunft dem Zufall unterliegen. Einige der genannten Veröffentlichungen haben den Zeithorizont der Vorhersage gar nicht beachtet, andere haben sich auf maximal drei Monate beschränkt, dafür jedoch die kurzen Horizonte von wenigen Tagen vernachlässigt. Vorhersagen über mehr als drei Monate hinweg wurden nicht betrachtet. Auch ist noch erforscht, inwiefern sich die Gewichte der Features mit den Zeithorizonten verändern.

Zusammenfassend lässt sich feststellen, dass der Vergleich der Modelle zwischen den Aktien – und nicht nur im Gesamtmarkt –, der Einfluss von technischen Features auf die Vorhersagegenauigkeit sowie die Veränderung der Gewichtungen der Features mit unterschiedlichen Zeithorizonten noch nicht ausreichend erforscht sind.

3 Methodik

3.1 Vorgehen

Nachfolgend wird das Vorgehen zur Untersuchung der Forschungsfragen dargelegt. Um den Dummy Klassifikator, Decision Tree und Random Forest zu vergleichen werden die in Tabelle 3.1 aufgeführten fünf Variablen eingeführt.

Tabelle 3.1 Die fünf zentralen Variablen für die Untersuchungen

Variable	Mögliche Werte
Klassifikator	{Dummy Klassifikator, Decision Tree, Random Forest}
Datenset (Tickersymbol der Aktie)	{AAPL, AMZN, CSCO, GE, GOOGL, HP, IBM, INTC, MSFT, WU, XRX}
Feature Engineering	{Ja, Nein}
Zeithorizont	{1d, 5d, 10d, 20d, 65d, 250d} mit den Indizes {0, 1, 2, 3, 4, 5}
Hyperparameter-Tuning	{Ja, Nein}

Die erste Variable ist der Klassifikator selbst. Er nimmt stets eine der drei genannten Ausprägungen – Dummy Klassifikator, Decision Tree oder Random Forest – an. Die zweite Variable ist das Datenset. Die elf ausgewählten Aktien-Datensets stammen allesamt aus dem Technologie Sektor und werden in Abschnitt 3.3 im Detail vorgestellt. Die dritte Variable ist das Feature Engineering. Um die Auswirkungen der zusätzlichen technischen Features zu erfassen, werden die Klassifikatoren im ersten Durchgang ohne diese trainiert und getestet, und im zweiten Durchgang mit ihnen. Die vierte Variable ist der Zeithorizont, der zwischen einem Handelstag (1d) und einem Handelsjahr (250d) liegt. Die fünfte Variable ist das Hyperparameter-Tuning. Dadurch soll eruiert werden, wie sich die Optimierung der Hyperparameter in verschiedenen Szenarien auf die Treffergenauigkeit des Modells auswirkt. Das allgemeine Vorgehen ist so, dass vier dieser Variablen konstant gehalten werden und die fünfte variiert. Anschließend werden Rückschlüsse gezogen, wie diese fünfte Variable das Ergebnis beeinflusst. Für jede Kombination dieser fünf Variablen wird dasselbe Prozedere angewendet, um sie zu bewerten und mit den anderen Kombinationen zu vergleichen.

Wie in Abbildung 3.1 dargestellt, ist der erste Schritt das Laden eines Datensets, das im Format .CSV (comma-separated values) vorliegt. Eine Vorstellung der ausgewählten Datensets folgt in Abschnitt 3.3. Das Ergebnis ist eine Datenstruktur, die die Zeitreihen eines Aktienkurses enthält. Nach dem Laden stehen für jeden enthaltenen Tag das Datum ("date"), der Name der Aktie ("name"), der Eröffnungspreis ("open"), das Tageshoch ("high"), das Tagestief ("low"), der Schlusspreis ("closing") und das gehandelte Volumen des Tages ("volume") zur Verfügung. Ein Vorteil von Decision Trees und Random Forests ist es, dass die Werte der Features keine Skalierung benötigen (Gron, 2017). Somit entfällt dieser Schritt.

Auch die Auswahl der wichtigsten Features ist im Gegensatz zu anderen Machine Learning Modellen nicht nötig. Die Bäume selektieren nämlich während ihrer Erstellung automatisch die wichtigsten Features insofern, als deren Aufteilungen die Entropie der Klassen am stärksten reduzieren (siehe Abschnitt 2.3.2).

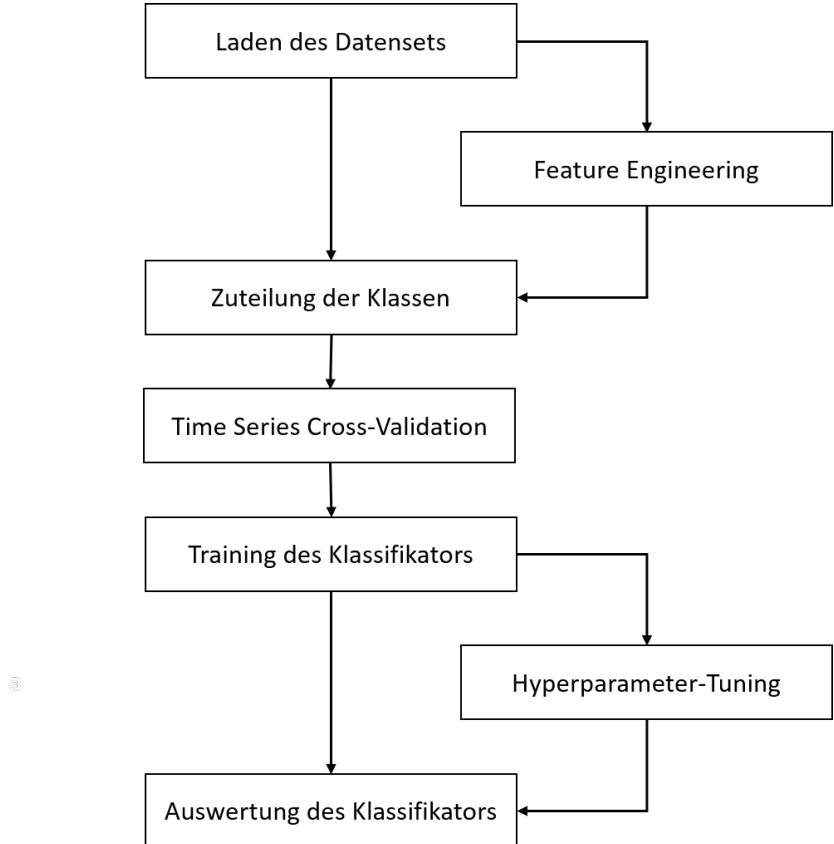


Abbildung 3.1 Vorgehen in der Untersuchung vom Laden des Datensets bis zur Auswertung des Klassifikators

Im zweiten Schritt werden weitere Features aus der technischen Analyse berechnet. Einige, in der Literatur weit verbreitete, technische Features sind der gleitende Durchschnitt, die Volatilität und das Momentum (Drakopoulou, 2016). Der gleitende Durchschnitt der Schlusspreise, c_i , der letzten k Tage, benannt als $ma_{-k>d}$, ergibt sich zum Zeitpunkt t aus

$$ma_{-k>d} = \frac{\sum_{i=t-k+1}^t c_i}{k}. \quad (3.1)$$

Die Volatilität der letzten k Tage, $volatility_{-k>d}$, zum Zeitpunkt t wird mit der Standardabweichung

$$volatility_{-k>d} = \sqrt{\frac{\sum_{i=t-k+1}^t (c_i - \bar{x})^2}{k-1}} \quad (3.2)$$

gemessen. Das Momentum der letzten k Tage, $momentum_{-k>d}$, bezeichnet die absolute Veränderung der Schlusspreise bis zum Zeitpunkt t :

$$momentum_{-k>d} = c_t - ct - k. \quad (3.3)$$

Ebenso wird die prozentuale Veränderung der Schlusspreises der letzten k Tage, $return_past_{-} < k > d$, als Feature ergänzt:

$$return_past_{-} < k > d = \frac{c_t}{c_{t-k}} - 1. \quad (3.4)$$

Für k werden jeweils die betrachteten Zeithorizonte h_i aus Tabelle 3.1 eingesetzt.

Der dritte Schritt ist die Zuteilung der Klassen. Bisher enthält die Datenstruktur lediglich Features. Um die Auswirkungen unterschiedlicher Klassifikationshorizonte zu untersuchen, wird pro Zeithorizont h_i eine eigene Klassenspalte erstellt. Dazu wird zuerst berechnet, ob der Schlusspreis in den nächsten h_i Tagen steigt oder fällt. Falls der Kurs steigt, wird der Instanz die Klasse 1 (positiv) zugeteilt. Andernfalls lautet die Klasse 0 (negativ). Nach dieser Logik zählt auch ein konstanter Kurs, dessen Werte zu Beginn und am Ende von h_i identisch sind, zur negativen Klasse. Der Klassifikator soll in diesem Fall also keine Kaufempfehlung abgeben. Zeithorizonte werden stets in Handelstagen angegeben, weshalb der maximale Horizont über ein Jahr 250 Tage besitzt. So gibt zum Beispiel die Klasse $class_10d$ der Instanz X vom 01.01.2015 an, ob der Kurs der Aktie bis zum 15.01.2015 steigt oder fällt. Nachdem die Klassen erstellt sind, werden alle Spalten mit Zwischenergebnissen, die zur Erstellung der Klasse verwendet wurden, aus der Datenstruktur entfernt. Dadurch ist gewährleistet, dass der Klassifikator nicht von illegal voraussehenden Features profitiert, die die Erfolgsmessung positiv verfälschen würden.

Im vierten Schritt findet die Time Series Cross-Validation statt. Diese ist für die vorliegenden Datensets besser geeignet als andere Cross-Validation Ansätze, da sie die Chronologie der Instanzen beibehält (siehe Abschnitt 2.8). Nicht nur technisch, sondern auch fachlich entspricht das der Problemstellung. Denn der Anwender des Klassifikators trainiert diesen auf historischen Daten, bis zum aktuellen Zeitpunkt, und wendet die Klassifikation dann auf den Zeithorizont h_i an, um eine Investitionsentscheidung zu treffen. In der realen Anwendung stehen nicht, wie es zum Beispiel bei der k-fold Cross-Validation der Fall wäre, Datensätze aus der Zukunft, die zeitlich hinter h_i liegen, zur Verfügung. Deshalb findet in dieser Arbeit die Time Series Cross-Validation Anwendung. Für die Anzahl der Epochen wird 10 gewählt; dadurch bestehen die Testsets pro Datensatz aus näherungsweise $\frac{1}{10} * 1259 = 126$ Instanzen. Mehr als zehn Epochen würden zu sehr kleinen Testsets führen, und weniger als zehn Epochen würden die Anzahl der Ergebnisse stärker einschränken.

Schritt fünf ist das Training des Klassifikators. Das Training erfolgt auf jenen Daten, die in der aktuellen Epoche als Trainingsdaten indiziert sind. Pro vergangener Epoche stehen ungefähr 126 zusätzliche Trainingsdaten zur Verfügung, die in der jeweils letzten Epoche als Testset dienten. Das bedeutet, dass in der ersten Epoche die wenigsten, und in der letzten Epoche die meisten Trainingsdaten vorhanden sind. Es werden in dieser Arbeit drei Klassifikatoren verglichen. Ein Dummy Classifier dient als Vergleichsmaßstab. Um einen Klassifikator sinnvoll in der Realität einsetzen zu können, muss er eine höhere Trefferrate als der Dummy Classifier erreichen. Der Dummy Classifier betrachtet nur die Verteilung der Klassen in den Trainingsdaten, nicht jedoch die sonstigen Features. Dann errechnet er die Wahrscheinlichkeit $P(y_i)$ pro Klasse y_i . Bei der Klassifikation unbekannter Instanzen erteilt er jeder Instanz dann mit Wahrscheinlichkeit $P(y_i)$ die Klasse y_i . Der Dummy Classifier nimmt an, dass die Klassenverteilung auf den Trainingsdaten stets mit der Verteilung auf den Testdaten übereinstimmt. Ausgehend von den Treffergenauigkeiten dieses simplen

Klassifikators lassen sich andere Modelle vergleichen. Die weiteren zwei Klassifikatoren sind der der Decision Tree (siehe Abschnitt 2.3) und der Random Forest (siehe Abschnitt 2.5).

Der sechste Schritt ist das Hyperparameter-Tuning, mit dem Decision Trees und Random Forests verbessert werden können. Für jeden der beiden Klassifikatoren werden zuerst zentrale Hyperparameter aus den Tabellen 2.1 und 2.2 ausgewählt. Dann werden für diese festgelegte Werte auf einem Validierungsset ausprobiert und verglichen, um die beste Kombination zu ermitteln. Wie auch bei der Bewertung der Klassifikatoren wird die Güte hier mit dem F-Maß bestimmt. Für den Hyperparameter der maximalen Anzahl an betrachteten Features ist die Wurzel der Anzahl an Trainingsinstanzen ein gängiger Standardwert (Probst u. a., 2018). In dieser Arbeit werden beim Hyperparameter-Tuning Werte bis maximal zu diesem Standardwert verglichen, um das Overfitting zu reduzieren. Auch für die anderen Hyperparameter der maximalen Tiefe, der minimalen Anzahl an Instanzen für die Aufteilung und der minimalen Anzahl an Instanzen für ein neues Blatt werden die zu vergleichenden Werte bis maximal zu diesem Standardwert gewählt. Die Anzahl der Basisklassifikatoren im Random Forest stellt eine Ausnahme dar, da sie nicht gleichermaßen durch Tuning optimierbar ist, sondern lediglich hinreichend hoch gesetzt werden muss (Probst u. a., 2018). Deshalb wird dieser Hyperparameter nach dem Tuning separat untersucht.

Neben dieser sogenannten Grid Search, bei der die zu vergleichenden Werte im Vorhinein vom Anwender festzulegen sind, gibt es auch noch eine zufallsgesteuerte Alternative, die Randomized Grid Search. Dann werden die zu vergleichenden Hyperparameter-Werte nicht explizit vom Anwender angegeben. Stattdessen werden zufällige Kombinationen solange betrachtet, bis eine vorher festgelegte maximale Anzahl zu vergleichender Kombinationen erreicht ist (Feurer u. Hutter, 2019). Die Randomized Grid Search ist besonders erfolgreich, wenn es nur wenige Features gibt, die mit sehr hohen Gewichtungen die wichtigsten sind (Feurer u. Hutter, 2019). Das Hyperparameter-Tuning wird in jeder der zehn Epochen aus der Time Series Cross-Validation separat ausgeführt. Ein einmaliges Tuning auf dem gesamten Datenset ist nicht möglich, da der Klassifikator sonst bereits Testdaten gesehen hätte und die Trefferrate positiv verfälscht würde.

Der letzte Schritt ist schließlich die Auswertung des Klassifikators in der jeweiligen Kombination der fünf Untersuchungsvariablen. Die Vorhersagen des trainierten, und gegebenenfalls optimiertem, Klassifikators für das unbekannte Testset werden mit den richtigen Klassen anhand einer Confusion Matrix verglichen. Die Erfolgsmessung durch Precision, Recall und das F-Maß erfolgt wie in Abschnitt 2.6 beschrieben. Ebenso werden die Gewichtungen der einzelnen Features, nachfolgend als Feature Importances bezeichnet, analysiert. Im Falle von Decision Trees und Random Forests ergeben sich diese Feature Importances aus der relativen Reduktion eines Unreinheitsmaßes, die der Baum durch Aufteilungen mittels des jeweiligen Features erreicht.

Diese sieben Schritte werden mehrmals durchgeführt, um die Ergebnisse je nach Kombination der Untersuchungsvariablen zu vergleichen und die Forschungsfragen zu beantworten. Für jedes Datenset (11 Möglichkeiten), mit oder ohne Feature Engineering (2 Möglichkeiten), mit einer der Klassen (6 Möglichkeiten), werden drei Klassifikatoren (3 Möglichkeiten), mit oder ohne Hyperparameter-Tuning (2 Möglichkeiten), trainiert und getestet. Insgesamt ergeben sich theoretisch $11 \times 2 \times 6 \times 3 \times 2 = 792$ mögliche Kombinationen. Von diesen werden im Ergebnisteil allerdings nur die für die Forschungsfragen relevantesten Kombinationen direkt miteinander verglichen. Ebenso werden Durchschnitte über eine der fünf

Dimensionen, wie beispielsweise über alle elf Datensets, berechnet, um übersichtliche und aussagekräftige Ergebnisse zu erhalten.

3.2 Tool-Stack und Bibliotheken

Die Untersuchungen in dieser Arbeit werden in einem Jupyter Notebook in der Python-Version 3.6.6 umgesetzt. Jupyter ist ein Open-Source Projekt mit dem Ziel, eine interaktive, Web-basierte und sprachenunabhängige Entwicklungsumgebung für Data Science-Anwendungen bereitzustellen. Ein Vorteil von Jupyter ist, dass sich Code, Graphiken sowie Markup-Texte in einer gemeinsamen Datei befinden, und man als Anwender die Ergebnisse somit direkt verwerten kann. Die Programmierung findet auf Azure Notebooks, einem Microsoft-Service für gehostete Jupyter Notebooks, statt. Als Kernel dient Python 3.6.6 aus der Anaconda Distribution, die bereits viele Machine Learning-relevante Pakete enthält.

Diese Arbeit nutzt die Pakete Scikit-learn, pandas, Numpy, Matplotlib, Seaborn und Graphviz. Die Open-Source Bibliothek Scikit-learn stellt Implementierungen für eine Vielzahl von Machine Learning Algorithmen, wie dem Decision Tree oder dem Random Forest, bereit (Pedregosa u. a., 2011). Auch nützliche Methoden rund um den Machine Learning Prozess deckt Scikit-learn ab, so zum Beispiel die Aufteilung von Daten in Training- und Testsets oder das Hyperparameter-Tuning. pandas bietet Werkzeuge zur Verarbeitung und Analyse von Daten an, wie zum Beispiel das zweidimensionale DataFrame. NumPy liefert eine effiziente Array-Implementierung und wird für mathematische Operationen, zum Beispiel zur Mittelwertberechnung oder Sortierung, verwendet. Matplotlib ist eine verbreitete Bibliothek zur Visualisierung in Python, die verschiedene Diagrammtypen unterstützt. Seaborn ist eine Erweiterung von Matplotlib und ist auf attraktive und informative Graphiken spezialisiert. Zur Visualisierung von Decision Trees wird Graphviz verwendet.

Das öffentliche GitHub Repository <https://github.com/feschu/BSc-Thesis-ML> enthält das Jupyter Notebook, die Datensets sowie die generierten Graphiken aus dieser Arbeit. Um diese Arbeit reproduzierbar zu halten, finden sich in Sektion 6 des Jupyter Notebooks die Zellen, mit welchen sämtliche Ergebnisse und Graphiken generiert wurden.

3.3 Datengrundlage

Elf Datensets bilden die Grundlage der Untersuchungen. Es handelt sich um die Aktienkurse der Unternehmen Apple (AAPL), Amazon (AMZN), Cisco (CSCO), General Electric (GE), Google (GOOGL), Hewlett-Packard (HP), IBM (IBM), Intel (INTC), Microsoft (MSFT), Western Union (WU) und Xerox (XRX) im Zeitraum vom 08.02.2013 bis zum 07.02.2018. Pro Aktie stehen somit 1.259 Handelstage zur Verfügung. Bei der Auswahl der Aktien wurde darauf geachtet, dass sie sich in ihrem Verlauf den fünf Jahren unterscheiden. So folgte AMZN einem steigenden Trend, HP einem ausgeglichenen Trend und GE einem fallenden Trend, wie in Abbildung ?? zu sehen.

Alle elf Unternehmen sind im Technologie-Sektor angesiedelt, teilweise jedoch mit unterschiedlichen Schwerpunkten. Während Google auf Software spezialisiert ist und Umsatz

größtenteils mit Werbung generiert, ist Cisco zum Beispiel auf Infrastruktur fokussiert. Die Datensets stammen aus dem Kaggle-Datenset SS&P 500 stock data" <https://www.kaggle.com/camnugent/sandp500>. Kaggle ist eine Plattform, die Machine Learning Wettbewerbe veranstaltet und es den Nutzern unter anderem ermöglicht, Datensets auszutauschen und die Anwendung von Machine Learning Algorithmen auf diesen zu diskutieren.

Abbildung 3.2 zeigt den relativen Verlauf der elf Aktien, ausgehend vom 08.02.2018. Die absoluten Werteverläufe der Aktien sind im Anhang zu finden.

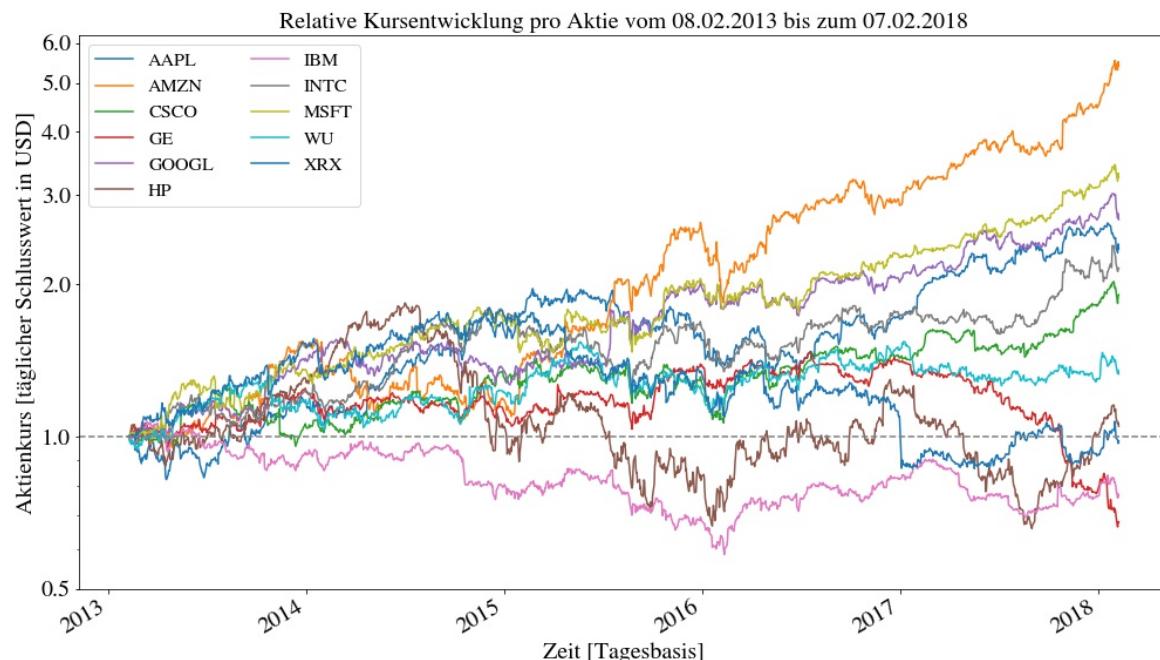


Abbildung 3.2 TBD

Beispieldaten sind in Tabelle 3.2 zu sehen. TBD: Beschreiben der Sopalten. Note: Month wurde in eine Zahl übersetzt, damit de rAlgo es als Feature akzeptiert.

Tabelle 3.2 TBD

date	open	high	low	close	volume	name	month	...
2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	2	...
2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	2	...
2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	2	...
2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	2	...
2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL	2	...

Tabelle 3.3 TBD

...	class_1d	class_5d	class_10d	class_20d	class_65d	class_250d
...	1	0	0	0	0	1
...	0	0	0	0	0	1
...	0	0	0	0	0	1
...	0	0	0	0	0	1
...	0	0	0	0	0	1

Zeigen: Klassenverteilung bei 6 Horizonten. Immer mehr unbalanced. Deshlab Confusion Matrix anstatt Accuracy. -> Es ist ersichtlich, dass Daten verschiedene Trends verfolgen.

4 Ergebnisse

Nachfolgend findet pro Fragestellung zuerst eine Beschreibung der Ergebnisse statt. Dann werden diese jeweils kritisch analysiert, und mit der Theorie und der aktuellen Literatur in Bezug gesetzt. Diagramme und Tabellen verschaffen einen schnellen und strukturierten Überblick und unterstreichen die wichtigsten Erkenntnisse. Aufgrund der hohen Anzahl untersuchter Variablen stehen zu viele Ergebnisse bereit, als dass diese im Kapitel eingebunden werden könnten. Nur die zentralen, häufig über mehrere Variablen aggregierenden Diagramme sind zu sehen. Die detaillierteren Ergebnisse, meistens in höherer Granularität wie beispielsweise pro Aktienset, befinden sich im Anhang. Limitierungen der Aussagekraft sollen, wo zutreffend, aufgezeigt werden. Im Anschluss werden weitergehende Ideen sowie aufgekommene Teilfragen präsentiert, die Ausgangspunkte für weitere Forschung darstellen können.

4.1 Erfolgsmessung der Klassifikatoren pro Zeithorizont

Der erste Ergebnisabschnitt beschäftigt sich mit der Treffergenauigkeit der Klassifikatoren auf den Datensets, gemessen mit dem F-Maß. Zusätzlich sind die zugrundeliegenden Precision- und Recall-Werte angegeben. Die Tabelle 4.1 zeigt die drei Metriken in Abhängigkeit von dem Zeithorizont für alle betrachteten Klassifikatoren. Die Werte stellen die Durchschnitte über alle elf Datensets dar.

Tabelle 4.1 Erfolgsmessung pro Klassifikator für die verschiedenen Zeithorizonte

Durchschnitt über alle Datensets							
Horizont		1d	5d	10d	20d	60d	250d
Precision	Dummy	0,519	0,531	0,538	0,540	0,582	0,743
	DT	0,514	0,553	0,551	0,540	0,632	0,845
	RF	0,515	0,553	0,571	0,542	0,658	0,885
	TunedDT	0,515	0,547	0,556	0,550	0,637	0,836
	TunedRF	0,521	0,539	0,556	0,550	0,618	0,879
Recall	Dummy	0,543	0,576	0,570	0,589	0,633	0,741
	DT	0,476	0,509	0,463	0,497	0,611	0,828
	RF	0,493	0,505	0,501	0,503	0,686	0,832
	TunedDT	0,544	0,488	0,438	0,508	0,662	0,835
	TunedRF	0,507	0,548	0,507	0,519	0,632	0,845
F-Maß	Dummy	0,531	0,552	0,554	0,563	0,607	0,742
	DT	0,495	0,530	0,503	0,518	0,621	0,836
	RF	0,504	0,528	0,534	0,522	0,672	0,858
	TunedDT	0,529	0,516	0,490	0,528	0,649	0,835
	TunedRF	0,514	0,543	0,530	0,534	0,625	0,862

4.1.1 F-Maße

Die erste Beobachtung bezüglich des F-Maßes ist, dass sich die Werte für den kürzesten und den längsten Zeithorizont stark unterscheiden. Die F-Maße für den 250-Tage-Horizont liegen deutlich über jenen für den 1-Tages-Horizont. Das trifft auf alle fünf Klassifikatoren zu. Im Schnitt liegen die Genauigkeiten bei 0,827 (250d) bzw. 0,515 (1d), mit einer Differenz von 0,312.

Um die Forschungsfrage zu beantworten, ob Decision Trees und Random Forests zur Aktienklassifikation sinnvoll eingesetzt werden können, werden zuerst die zwei Extrempunkte betrachtet: 1d und 250d. Für den kürzesten Horizont erzielen der Decision Tree und Random Forest Klassifikator F-Maße von circa 0,5. Sofern die beiden Klassen gleich wahrscheinlich sind, entsprechen diese Werte zufälligem Raten. Verglichen mit dem Ergebnis von 0,531 des Dummy Klassifikators – der sich als Vergleichsmaßstab besser eignet, da er die Verteilungsfunktion der Klassen beachtet – sind die erzielten F-Maße niedriger. Auch wenn die Decision Tree und Random Forest Klassifikatoren mit Tuning bessere Ergebnisse erreichen als ohne Tuning, ist der Dummy in den Horizonten 1d bis 20d erfolgreicher. Daraus folgt, dass die Decision Tree und Random Forest Klassifikatoren bei kurzfristigen Horizonten von bis zu 20 Tagen zur Aktenvorhersage nicht sinnvoll einsetzbar sind.

Betrachtet man nun die längerfristigen Horizonte 60d und 250d, so lässt sich feststellen, dass der Dummy Klassifikator allen anderen Klassifikatoren deutlich unterlegen. Im 60-Tage-Horizont sind zwar alle Decision Tree und Random Forest Varianten genauer als der Dummy, teilweise jedoch nicht signifikant. Zum Beispiel ist der Decision Tree um 0,014 besser, oder mit Tuning um 0,042. Der Random Forest übertrifft den Dummy um 0,065, oder mit Tuning um 0,018. Das Hyperparameter Tuning führt also einmal zu besseren, und das andere mal zu schlechteren Ergebnissen. Genauere Untersuchungen dazu folgen in Abschnitt 4.4. Für den 60-Tage-Horizont lässt sich somit keine eindeutige Aussage treffen. Im 250-Tage-Horizont

jedoch liegt der Dummy bei 0,742, während die komplexeren Klassifikatoren in dem Intervall zwischen 0,835 und 0,862 liegen. Der Random Forest mit Tuning weist das höchste F-Maß von 0,862 auf, 0,120 höher als der Vergleichsmaßstab. Alle vier komplexeren Modelle übertreffen den Dummy signifikant. Somit sind Decision Tree und Random Forest Klassifikatoren für den 250-Tage-Horizont sinnvoll einsetzbar. Das bedeutet, dass die Modelle dazu in der Lage sind, anhand der Beispieldaten so zu lernen, dass sie bessere Aussagen über die Zukunft treffen als der Dummy Klassifikator.

Um die Veränderung des F-Maßes mit steigenden Horizonten für die Klassifikatoren miteinander zu vergleichen, sind diese als Linien in Abbildung 4.1 zu sehen. Das F-Maß ist wieder als Durchschnitt über die Datensets berechnet. Es ist bei der Darstellung zu beachten, dass die X-Achse verzerrt ist, denn zwischen den äquidistanten Markierungen liegen zuerst vier Tage, dann fünf Tage, zehn Tage, 40 Tage und schließlich 190 Tage. Diese Verzerrung führt, vor allem im Abschnitt zwischen 20d und 250d, zu einer höheren Steigung, als es bei einer linearen X-Achse der Fall wäre.

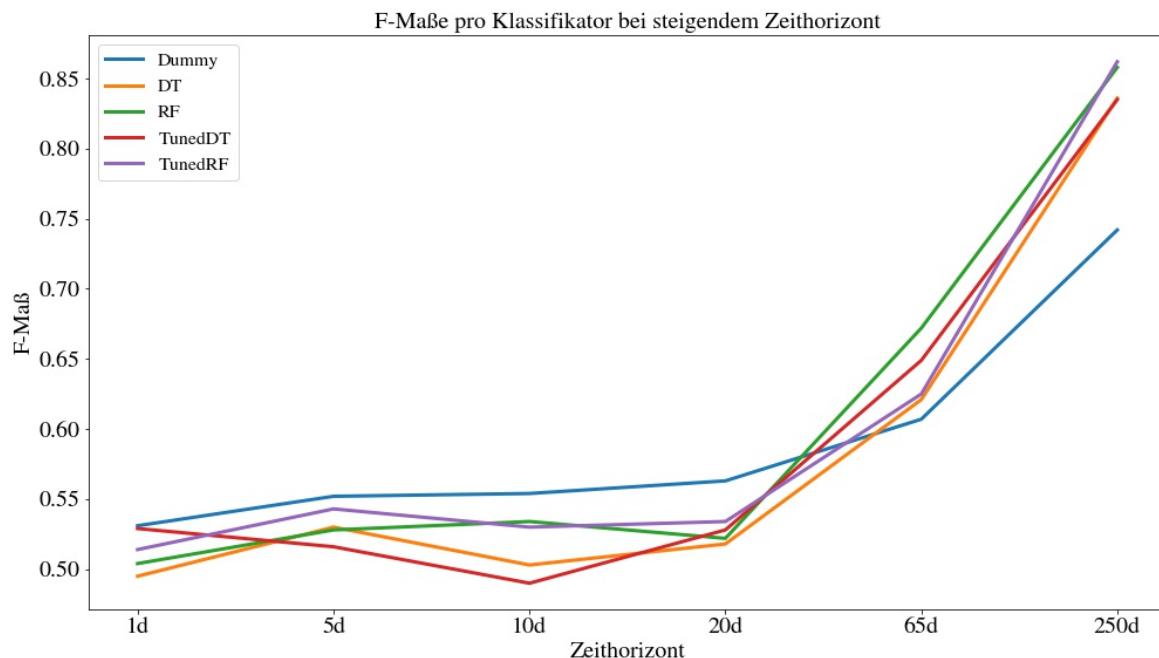


Abbildung 4.1 TBD

Es ist zu sehen, dass auf kurzfristige Horizonte der Dummy Klassifikator die besten Ergebnisse erzielt, ab 65d der Dummy jedoch, wie beschrieben, das schlechteste Ergebnis erreicht. Diese Arbeit hat sich auf maximal 250 Tage beschränkt. Mit größeren Datensets bietet es sich zukünftig an, Horizonte über mehrere Jahre auszuprobieren. Hält der positive Trend der F-Maße, wie in Abbildung 4.1 zu sehen, an, so könnten sich damit zuverlässigere Modelle finden.

Ein Investor, der mit den vier komplexeren Modellen – DT, RF, TunedDT und TunedRF – die Entscheidung trifft, welche Aktien er kauft und welche nicht, und dafür die zukünftige Richtung des Aktienkurses betrachtet, liegt durchschnittlich bei mehr als 8 von 10 Entscheidungen richtig. Investiert er denselben Betrag in jede der 10 Entscheidungen, so kann

er Gewinne erzielen. Es ist zu beachten, dass zwar die Richtung der Kursänderung in 8 von 10 Fällen korrekt bestimmt wird, jedoch die absolute oder prozentuale Abweichung, Δ_{Kurs} , unbekannt bleibt. Es besteht die Gefahr, dass die negativen Δ_{Kurs} -Werte der zwei fehlklassifizierten Aktienbewegungen größer sind als jene der korrekt-klassifizierten. Ist das der Fall, würde der Investor – trotz korrekter Vorhersage von 8 von 10 Aktientrends – Verluste erzielen. Wann dieser Fall eintritt und wann nicht, und welche Faktoren für den Erfolg des Investors maßgeblich sind, wird nachfolgend erläutert. Entscheidet sich der Investor aber dafür, nur in steigende Aktien anzulegen und die fallenden zu vermeiden, dann wird er – unter der Annahme, dass die Aktienmärkte langfristig steigen und positive Δ_{Kurs} -Werte im Durchschnitt höher ausfallen als negative – bei hinreichend hoher Anzahl von Investitionen einen Gewinn erzielen. In der Realität muss diese Annahme aber nicht zutreffen. Ob positive oder negative Δ_{Kurs} -Werte durchschnittlich höher sind, hängt von der Entwicklung des Gesamtmarktes sowie von der Anzahl steigender und fallender Aktien ab. Wenn der Gesamtmarkt konstant bleibt, also ein Index über alle Aktien weder zu- noch abnimmt, und die Anzahl der steigenden und fallenden Aktien gleich ist, dann würde man im Durchschnitt erwarten, dass der Δ_{Kurs} -Wert in beide Richtungen, positiv oder negativ, gleich ist. Dann erwirtschaftet der Investor mit den komplexen 250d-Klassifikatoren und 8 von 10 richtigen positiven Trendprognosen einen Gewinn, sofern er in eine hinreichend hohe Anzahl an Aktien investiert. Es muss also das große Gesetz der Zahlen greifen, wonach mit steigender Anzahl an Beobachtungen einer Zufallsvariable deren Durchschnitt sich an den Erwartungswert annähert. Anders verhält es sich in den Extremfällen. Steigen 100% der Aktien, so ist der durchschnittliche positive Δ_{Kurs} -Wert größer als Null, der durchschnittliche negative Δ_{Kurs} -Wert ist dann gleich Null. Dann erzielt der betrachtete Anleger, der nur auf steigenden Kurse setzt, einen Gewinn. Im umgekehrten Fall, wenn 100% der Aktien fallen, so ist der durchschnittliche positive Δ_{Kurs} -Wert gleich Null, der durchschnittliche negative Δ_{Kurs} -Wert ist kleiner Null. Dann erleidet eben dieser Anleger einen Verlust. In der Realität ist davon ausgehen, dass es global gesehen in jedem beliebigen 250-Tage-Horizont sowohl steigende als auch fallende Aktien gibt. Betrachtet man dann den Fall, dass 99% der Aktien steigen, so müssten die fallenden 1% der Aktien deutlich stärker nachgeben, um den Markt-Index konstant zu halten. Also wäre der erwartete positive Δ_{Kurs} -Wert der steigenden Aktien deutlich kleiner als der negative Δ_{Kurs} -Wert der fallenden Aktien. In diesem Fall drohen die 2 Fehlklassifizierungen pro 10 Aktien die Gewinne der 8 korrekten Klassifizierungen auszugleichen, sodass der Investor weder einen Gewinn noch einen Verlust erfährt, oder überzukompensieren, sodass er Verluste erleidet. Im anderen Fall steigen 1% der Aktien sehr stark an, die restlichen 99% fallen leicht. Dann würde die das Gegenteil des genannten Prozederes eintreten, was zum Vorteil des Investors ist. Es liegt nahe, dass die historische Verteilung von steigenden und fallenden Aktien in den letzten Jahrzehnten zwischen den genannten Extrempunkten lag, und sich diese fortlaufend ändert. Ab einem zu bestimmenden Schwellwert s_{Gewinn} erwartet man dann, dass die Strategie, die 250d-Klassifikatoren anzuwenden und nur in positive Kursänderungen zu investieren, zu Gewinnen führt. Je kleiner der Anteil der steigenden Aktien ist, desto höher ist bei konstantem Gesamtmarkt deren Δ_{Kurs} -Wert im Vergleich zu jenem der fallenden – was zum Vorteil des Investors ist – und vice versa. Wenn nun der Gesamtmarkt nicht konstant ist sondern sich auf- bzw. abwärts bewegt, so führt dies zu einer negativen bzw. positiven Verschiebung des Schwellwerts s_{Gewinn} . Untersucht man die Verteilung von steigenden und fallenden Aktientrends und das gesamte Marktwachstum genauer und prognostiziert diese, zum Beispiel wiederum mit Klassifikatoren, dann lassen sich basierend auf den Erkenntnissen fortgeschrittenere Anlagestrategien formulieren. In Abhängigkeit von dem resultierenden

s_{Gewinn} und den erwarteten Marktumständen lässt sich dann beispielsweise bestimmen, ob die Anwendung des 250d-Klassifikators ökonomisch sinnvoll ist oder nicht. Es ist weiterhin zu beachten, dass der Investor nur dann einen beträchtlichen Gewinn erzielen kann, wenn die Anzahl der Aktien, die als positiv klassifiziert und gekauft werden, groß genug ist. Falls der Markt-Index stark steigt und nur sehr wenige Aktien dieses Wachstum begründen, also stark positiv wachsen, und der Klassifikator die Treffergenauigkeit von über 8 von 10 einhält, droht eine Verknappung an steigenden Aktien. Im Extremfall findet der Investor keine oder nur wenige Aktien, die als steigend klassifiziert werden. Dann kann der Anleger das Investitionsrisiko nicht streuen und ist von der Kursentwicklung weniger Aktien abhängig, ausgleichende Kursschwankungen durch das Gesetz der großen Zahlen bleiben aus. Auch die benötigte Laufzeit zur Klassifikation der Aktien würde dementsprechend verlängert, was möglicherweise zu entgangenen Gewinnen, sicher aber zu erhöhten Ressourcenkosten führt. Der Investor muss global handeln, damit das Angebot an Aktien möglichst groß ist und die Strategie funktioniert, und er muss dabei die Effizienz der Klassifizierung sicherstellen. Alternativ kann die Strategie umgedreht werden, sodass der Investor nur auf fallende Aktienkurse setzt. Das bietet sich zum Beispiel in Krisenzeiten an, in denen der Großteil der Aktien und der Markt-Index fallen. Je nach Marktsituation eignet sich eine der beiden Strategien besser, was weiteres Optimierungspotenzial bietet.

Zuletzt veranschaulicht die Tabelle 4.1 auch die Unterschiede zwischen den F-Werten von Decision Tree und Random Forest Klassifikatoren ohne Tuning und jenen mit Tuning. Diesbezüglich ist zu sehen, dass das Tuning der Hyperparameter nur teilweise zu besseren Ergebnissen führt. In 7 von 12 Fällen bewirkt das Tuning eine Steigerung des F-Maßes, in den restlichen 5 Fällen eine Minderung. Eine detailliertere Untersuchung dieses Sachverhalts folgt im Abschnitt 4.4.

Neben den durchschnittlichen F-Maßen über alle Datensets ist es von Interesse, die einzelnen Werte pro Aktie zu analysieren. Im Anhang A.2 sind diese als Säulendiagramme zu sehen. Eine horizontale Linie bei 0,500 hilft bei der Auswertung. Es ist zu sehen, dass sich die Veränderung im F-Maß mit steigendem Zeithorizont zwischen den einzelnen Aktien systematisch – für alle fünf Klassifikatoren – unterscheidet. Dieser Unterschied, gemessen in Prozentpunkten, wird mit steigendem Horizont signifikant größer. Auffällig ist, dass verschiedene Aktien ihr höchstes F-Maß in unterschiedlichen Horizonten erreichen. So sind die Klassifikatoren für XRX tendenziell auf den Horizonten von 1d bis 20d besser, für AMZN jedoch auf 65d und 250d. Es ist also offensichtlich, dass sich die Lernfähigkeit der Klassifikatoren pro Aktie unterscheidet. Die Idee ist nun, die Aktien anhand ihrer Kursverläufe zu gruppieren und dann genauer zu untersuchen, wie erfolgreich die Klassifikation pro Gruppe ist. Mit dieser Frage beschäftigt sich Abschnitt 4.1.3. Eine noch detailliertere Darstellung der Ergebnisse, die zusätzlich Recall- und Precision-Werten beinhaltet, findet sich in Tabellen in Anhang A.3.

4.1.2 Precision und Recall

Neben dem F-Maß zeigt die Tabelle 4.1 auch die Recall- und Precision-Werte. Es ist erkennbar, dass der Dummy Klassifikator in allen bis auf den letzten Horizonten einen höheren Recall als Precision erzielt. Das heißt, der Klassifikator erkennt zwar viele der positiven Instanzen als positiv. Wenn er aber Instanzen als positiv bestimmt, dann hat er dort eine vergleichsweise niedrigere Erfolgsquote, richtig zu liegen. Er neigt also dazu, eher falsche

Positive als falsche Negative zu deklarieren, als die anderen Modelle. Denn die vier Decision Tree und Random Forest Modelle weisen in 19 von insgesamt 24 Fällen (4 Klassifikatoren mit jeweils 6 Horizonten) eine höhere Precision als Recall auf. Somit neigen sie im Vergleich zum Dummy Klassifikator eher dazu, . Je nach Auslänge des Use Cases macht das den einen oder den adnernen Klassifikator besser geeignet.

Ob der Recall- oder oder Precision-Wert wichtiger ist und maximiert werden sollte, hängt stets vom konkreten Anwendungsfall ab. In diesem Fall sollen die Anlageentscheidungen eines Investors möglichst hohe Renditen bei minimalem Risiko einbringen. Um die Präferenz eines Investors zu finden, werden zuerst zwei Extremfälle verglichen. Im ersten Szenario liegen 100 Kaufempfehlungen für Aktien vor, von denen 80 Stück im betrachteten Horizont tatsächlich steigen. Der Anleger hat also pro Aktie eine Erfolgswahrscheinlichkeit von 80%. Im zweiten Szenario liegen nur 20 Kaufempfehlungen vor, von denen aber 19 Stück im betrachteten Horizont tatsächlich steigen. Dann liegt die Erfolgswahrscheinlichkeit pro Aktie bei 95%. Ohne weitere Annahmen über die Risikobereitschaft zu treffen, lässt sich keine eindeutige Präferenz des Investors bestimmen. Möchte der Investor um jeden Preis Verluste vermeiden und ist dafür bereit, möglicherweise Gewinn zu verpassen, so würde er für Szenario 2 präferieren. Möchte er hingegen eine möglichst hohe Anzahl an steigenden Aktien besitzen und nimmt dafür mehr fallende in Kauf, so würde er Szenario 1 vorziehen. Zur Anwendung der Klassifikatoren in der Realität muss also die Risikobereitschaft des Anlegers bekannt sein, um die richtigen – bezogen auf die Anleger-spezifische Optimierung des Recall- oder des Precision-Wertes – Modelle einzusetzen.

4.1.3 Analyse pro Aktiengruppe

Es folgt eine Analyse der beobachteten F-Maß-Differenzen der Klassifikatoren zwischen den einzelnen Aktien. Beim Vergleich der F-Maße pro Aktie im Anhang A.2 fällt auf, dass XRX bei 250 Tagen für jeden Klassifikator unter 0,100 liegt, und somit dem allgemeinen Trend – ein steigendes F-Maß bei steigendem Horizont – gemessen am Durchschnitt, widerspricht. Man betrachte nun den Aktienverlauf von XRX im Anhang A.1. Bis 2015 steigt die Aktie in einem deutlich positiven Trend, anschließend jedoch fällt sie kontinuierlich bis 2017. Von 2017 bis zum letzten Datensatz aus 2018 verhält sich der Kurs alternierend. Gleichzeitig liegen die F-Maße aller Klassifikatoren für AMZN mit Zeithorizont 250d bei über 96%. Der Aktienkurs von AMZN verfolgt über den gesamten Beobachtungszeitraum von fünf Jahren einen Aufwärtstrend mit nur wenigen kurzweiligen Ausnahmen. Es kommt die Vermutung auf, dass der Kursverlauf der Aktie Einfluss auf die F-Maß Entwicklung der Klassifikatoren hat. Deshalb werden die elf Datensets für eine genauere Analyse in drei Gruppen unterteilt. Das Kriterium für die Zuordnung einer Aktie in eine Gruppe ist der Trend ihres Kursverlaufs. Die Einteilung ist in Tabelle 4.2 dargestellt.

Tabelle 4.2 Einteilung der Aktien in Gruppen zur weiteren Analyse

Gruppe	Kursverlauf	Datensets
1	Steigend	AAPL, AMZN, CSCO, GOOGL, INTC, MSFT
2	Alternierend	GE, HP, WU, XRX
3	Fallend	IBM

Die Liniendiagramme 4.2, 4.3 und 4.4 zeigen die relativen Kursentwicklungen der Aktien pro Gruppe ausgehend vom 08.02.2013.

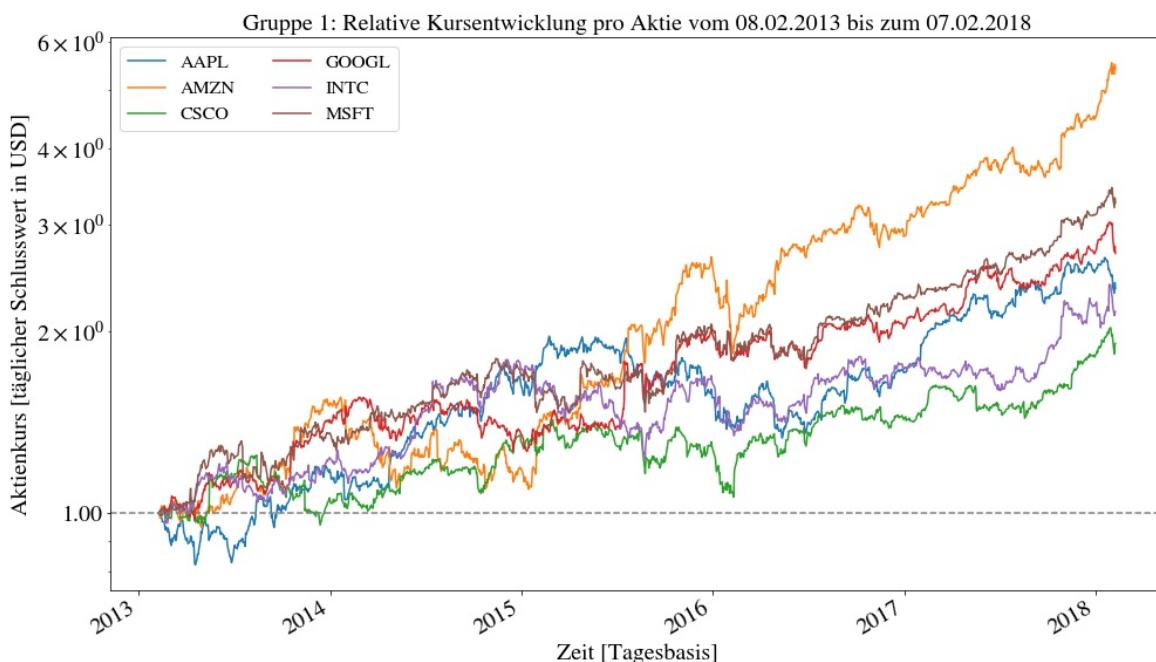


Abbildung 4.2 TBD

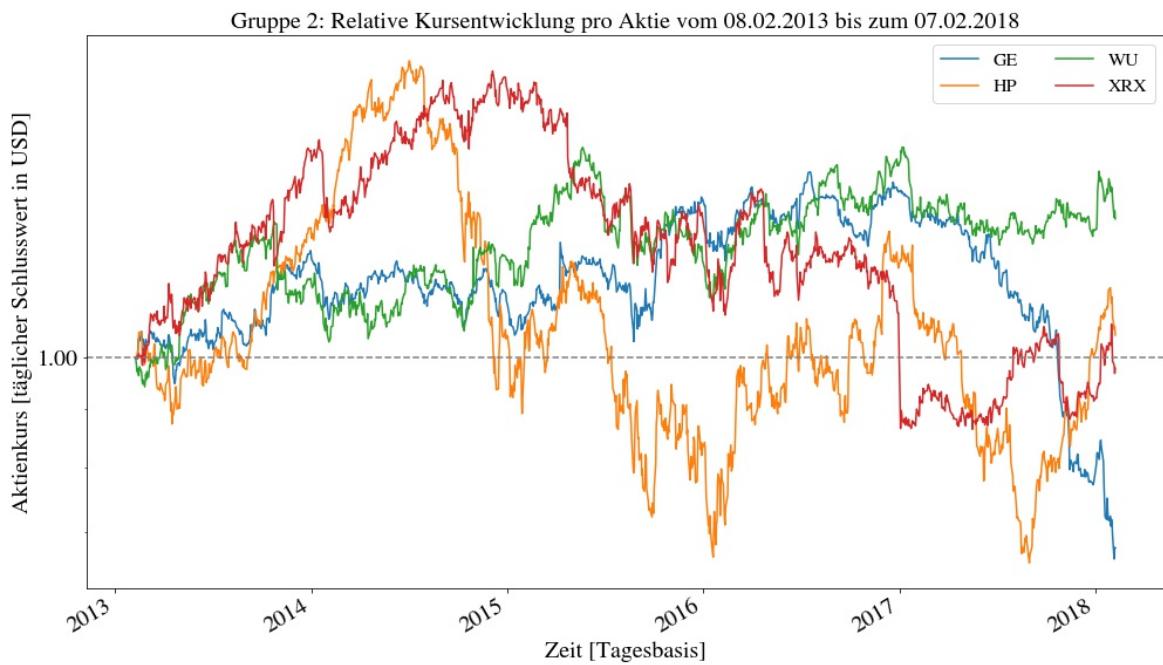


Abbildung 4.3 TBD

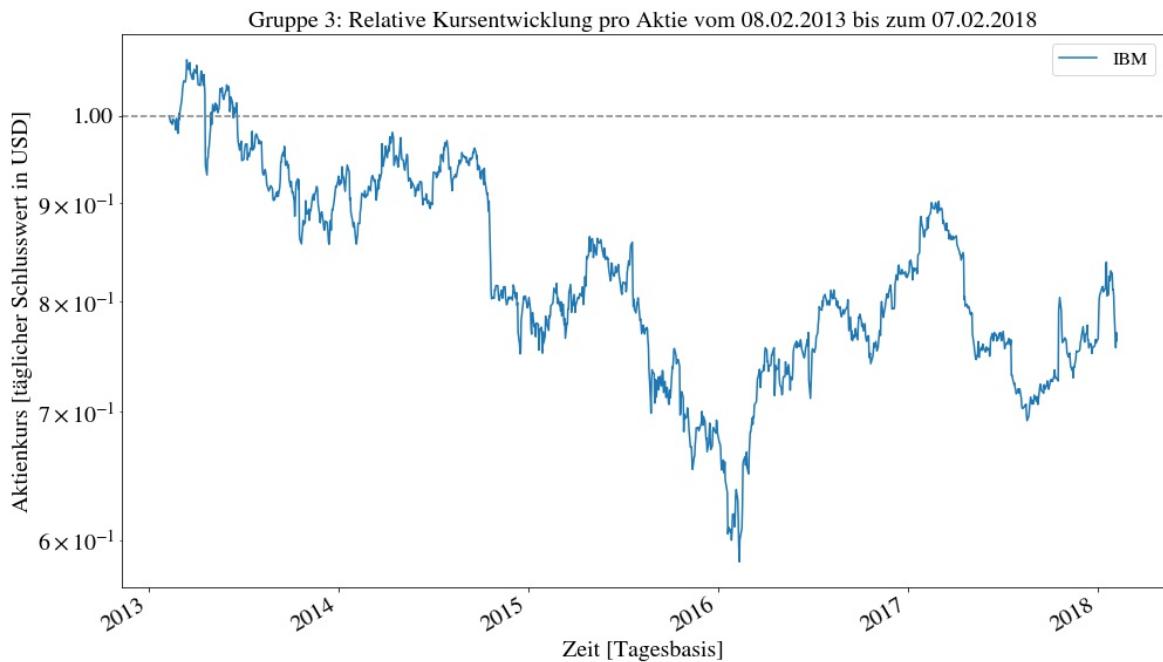


Abbildung 4.4 TBD

Stellt man sich den Zeithorizont der Klassifikation als horizontale Linie vor, so kann man Vermutungen anstellen, wie sich eine Verlängerung dieser Horizonts bzw. der horizontalen Linie auf den Klassifikator auswirkt. In Abbildung 4.2 kann man so feststellen, dass ein längerer Horizont in den meisten Fällen zu einer deutlicheren Entscheidung führt. Sucht man

sich einen der sechs Aktienkurse aus und stellt sich einen 30d-Horizont vor, den man von 2013 startend an den Tagespunkt anhängt, so finden sich zwar sehr viele positive Instanzen, bei denen der Wert am Ende des Horizonts größer als der aktuelle Tageswert ist. Aber es gibt auch negative Instanzen, bei denen der Kurs in den kommenden 30 Tagen sinkt. Verlängert man nun den Horizont auf 5 Jahre, also auf die komplette vorhandene Zeitdauer, so kommt man bei jeder Klassifikation zu einem positiven Ergebnis. Der Klassifikator profitiert also von einer möglichst langen Histoel Ähnlich verhält es sich in Abbildung 4.4, nur in die andere Richtung: je länger der Horizont, desto mehr negative Instanzen sind zu erkennen. Für die Aktien aus Gruppe 2 gelten diese Regeln nicht. Abhängig von den Wellenlängen führen verschiedene Zeithorizonte zum höchsten F-Maß.

Um die Vermutungen, wie sich eine Horizontverlängerung auf die erzielten F-Maße pro Gruppe auswirken, zu untersuchen, wurden in Abbildung 4.5 die entsprechenden F-Maße berechnet. Jedes F-Maß ist als Durchschnitt über die vier Klassifikatoren, die in als sinnvoll befunden wurden, errechnet. Wieder zeigt die X-Achse die Zeithorizonte in verzerrter Skalierung, sodass die äquidistanten Markierungen zunehmende Zeitintervalle bedeuten.

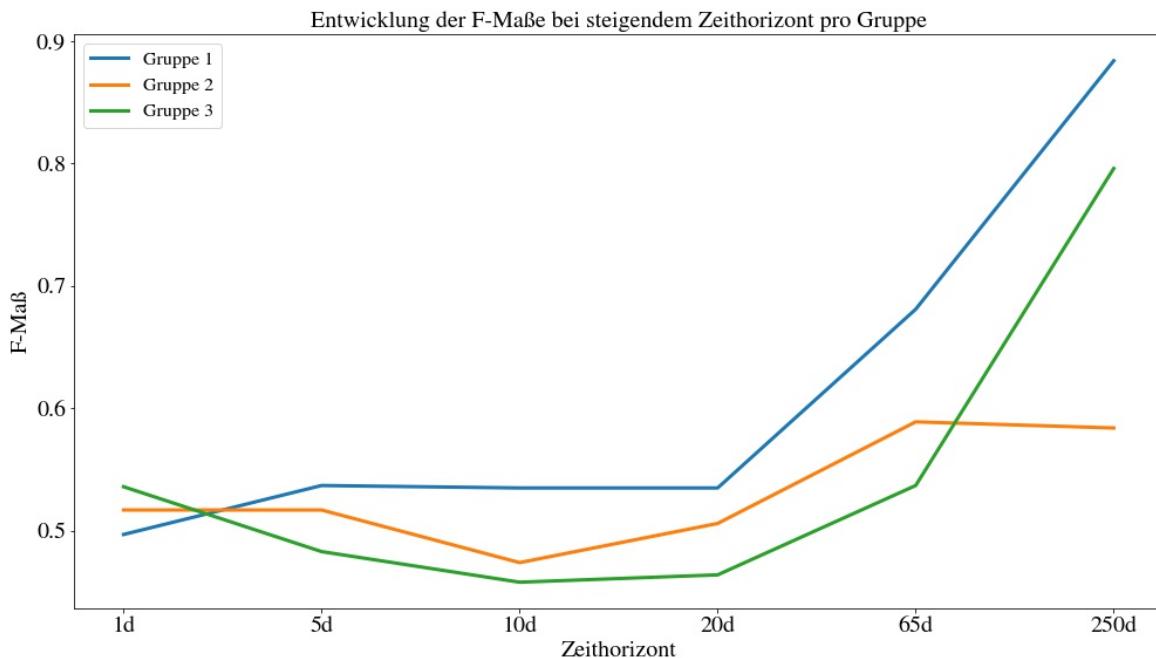


Abbildung 4.5 TBD

Es lässt sich festhalten, dass die Klassifikatoren auf den Aktien mit Aufwärts- und Abwärtsstrends, Gruppe 1 und Gruppe 3, mit längerem Horizont bessere Ergebnisse erzielen. Bei 250d erreichen sie ein deutliches Maximum. Auf den alternierenden Daten hingegen, aus der Gruppe 2, stagniert das F-Maß nach 65d. Die Klassifikatoren profitieren nicht von den zusätzlichen 185 Tagen. Es ist zu beachten, dass in diesen Untersuchungen die Gruppe 2 aus vier ausgewählten alternierenden Aktien besteht, wodurch eine Verzerrung gegenüber der entsteht. Für andere alternierende Aktien kann sich der Höhepunkt unterschieden. Auch ist es denkbar, dass bei wellenförmigen Kursverlauf mehrere Zeithorizonte optimal sind. Beträgt die Wellenlänge l und liegt ein lokales Maximum bei p , dann könnten alle Punkte aus $\{q \mid q \in p \times n, n \in \mathbb{N}\}$ ebenfalls zu lokalen Maxima führen. Wie viele dieser Punkte

tatsächlich zu lokalen Maxima führen hängt davon ab, wie viele Perioden sich die Welle im Aktienkurs wiederholt. Solche Vorhersagen könnten funktionieren, wenn beispielsweise jährliche Events wie die Veröffentlichung des Jahresberichts oder Ankündigungen von Zentralbanken einen absehbaren Effekt auf den Aktienkurs haben. Um Klassifikatoren für solche Prognosen zu trainieren, sollten Unternehmen verschiedener Branchen verglichen werden. Es ist zum Beispiel vorstellbar, dass Aktienkurse von Unternehmen mit besonders hohem Finanzierungsbedarf stärker auf Leitzinsänderungen reagieren. Die Vorhersage, wie ein solches wiederkehrendes Event den Aktienkurs beeinflusst, ist von externen Faktoren abhängig. So ist es möglich, dass eine Entscheidung der europäischen Zentralbank von vorherigen Entscheidungen anderer Notenbanken abhängt. Kann man einflussreiche Events und deren Einfluss auf ausgewählte Aktienkurse vorherzusagen, könnte man für alternierende Aktien eine Handelsstrategie entwerfen, um mit jeder Welle zu profitieren.

Die ausführlichen Ergebnisse jedes Klassifikators pro Aktiengruppe finden sich im Anhang A.4. Es fällt auf, dass alle Decision Tree und Random Forest Varianten auch in Gruppe 3 – also auf Aktien, die einen Abwärtstrend verfolgen, wobei die Modelle von längeren Horizonten profitieren sollten – in den Horizonten 65d und 250d eine deutliche Steigerung des F-Maßes erreichen. Der Dummy Klassifikator jedoch weist in den zwei längsten Horizonten fallende F-Maße auf. Während er auf 1d, 5d und 10d ungefähr eine Trefferrate von circa 50% erreicht, liegt diese bei 250d bei knapp 30%, dem Minimum.

4.2 Feature Importances in Abhängigkeit von dem Zeithorizont

Aussage: DT und RF gewichten zur Vorhersage von langen Zeithorizonten die langfristigen Features stärker als schwachen, und vice versa.

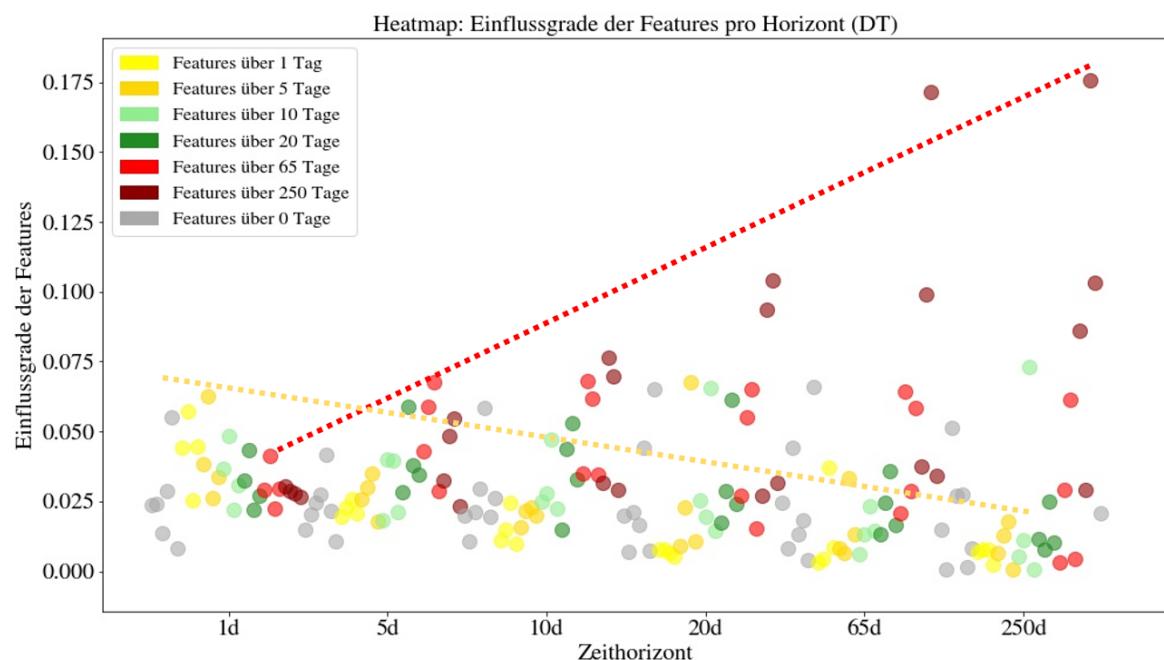


Abbildung 4.6 TBD

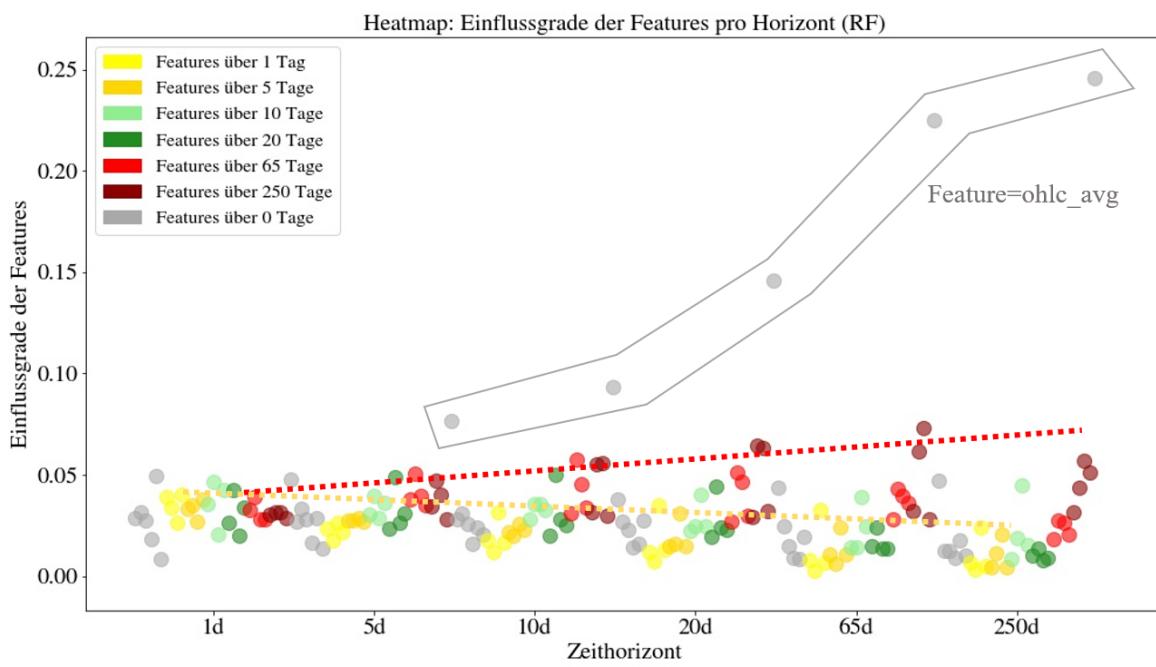


Abbildung 4.7 TBD

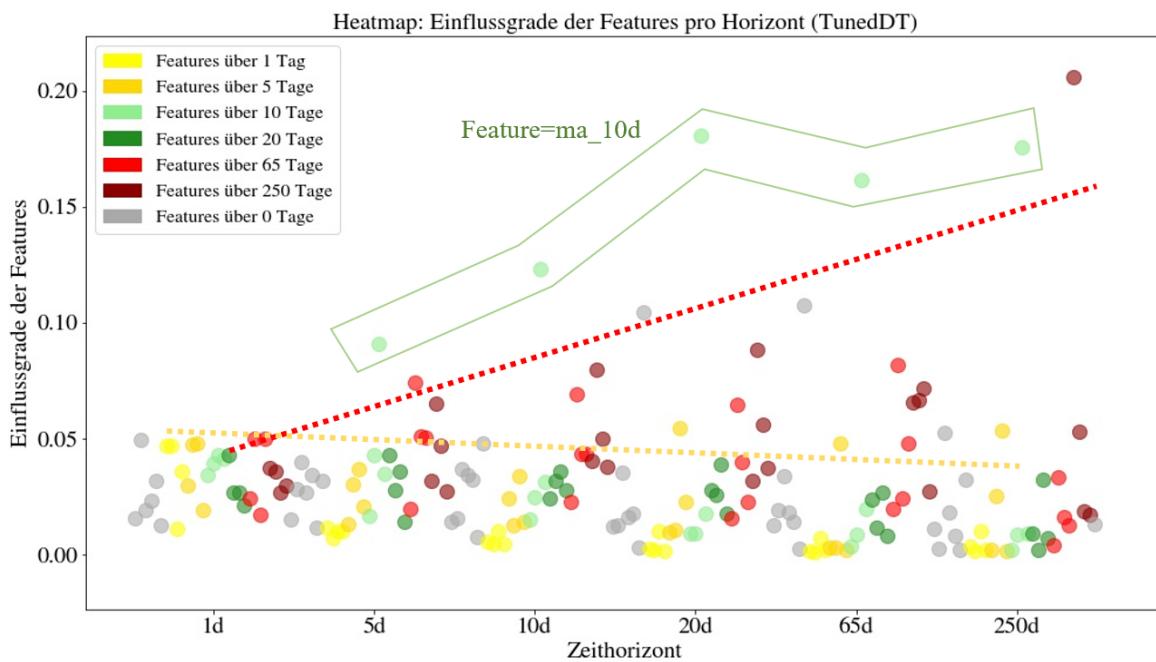


Abbildung 4.8 TBD

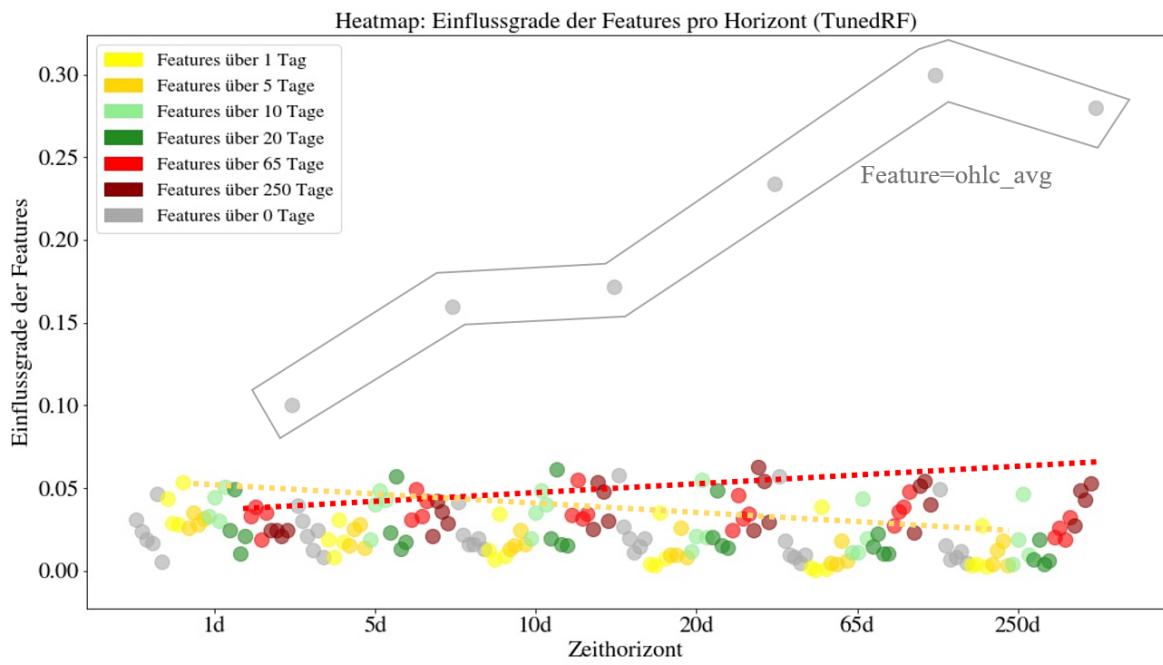


Abbildung 4.9 TBD

4.3 Einfluss von Feature Extraction

Aussage: Feature Extraction führt zu höherer Genauigkeit (F-Maß) von DT und RF Klassifikatoren, nicht jedoch von Dummy Classifier.

Plot:

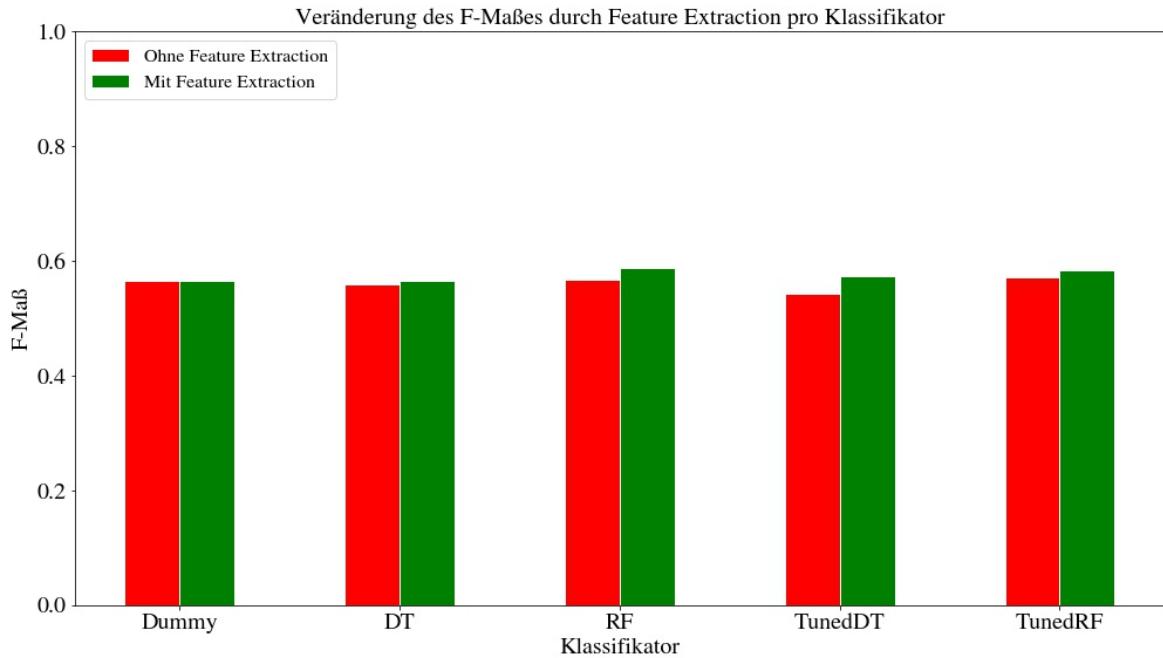


Abbildung 4.10 TBD

Table:

Tabelle 4.3 TBD

Veränderung des F-Maßes durch Feature Extraction	Ohne Feature Extraction	Mit Feature Extraction	Differenz
Dummy	0,565	0,565	0,000
DT	0,559	0,567	0,007
RF	0,569	0,587	0,019
TunedDT	0,543	0,574	0,030
TunedRF	0,572	0,584	0,012

4.4 Optimierung der Hyperparameter

Aussage: Tuning generell für diesen Anwendugnsfall sehr schwer, teilweise negative Auswirkungen. Neben 10TSCV auch 5TSCV, 2TSCV und 3fCV probiert, selbes Ergebnis.

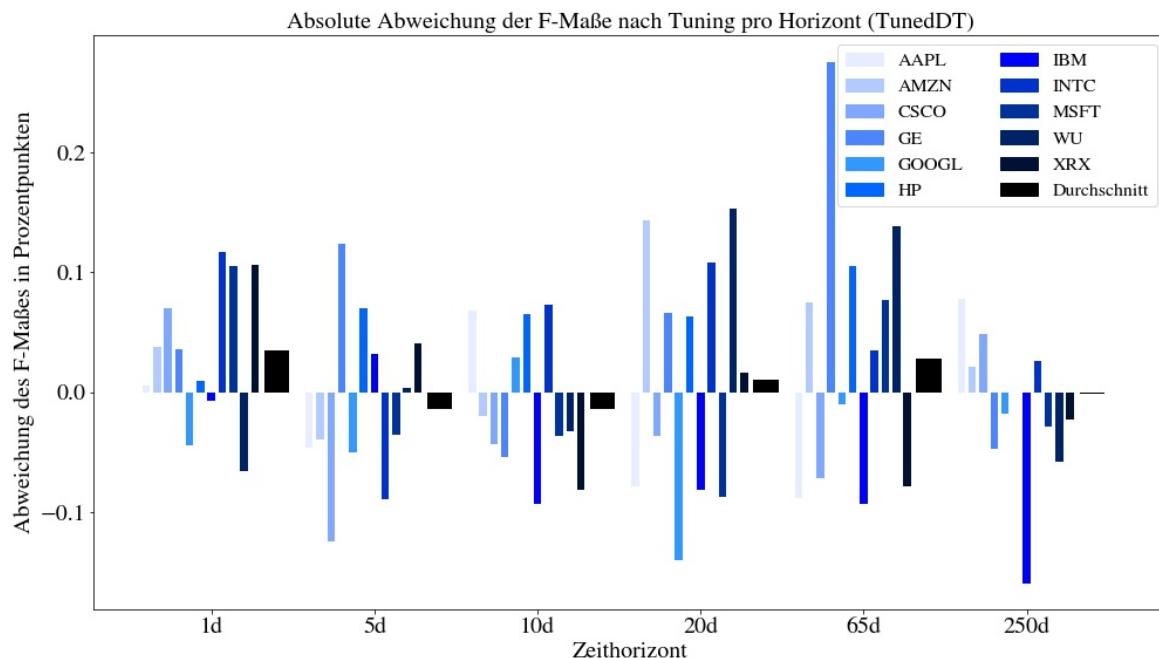


Abbildung 4.11 TBD

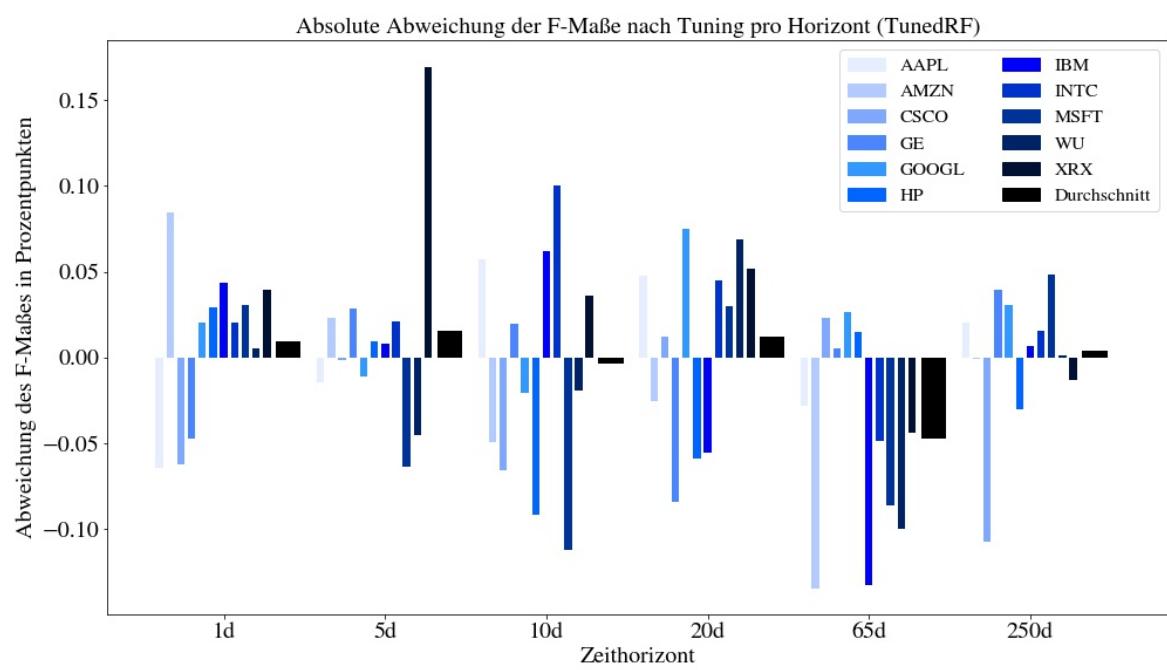


Abbildung 4.12 TBD

5 Zusammenfassung

5.1 Zusammenfassung

Diese Arbeit hat gezeigt, dass kurzfristig nicht möglich ist den Markt zu übertreffen. Langfristig aber schon.

5.2 Ausblick

Alle TBDs durchschauen und hier vorschlagen. z.B. Gewichtung der Base Classifier im Ensemble testen, etc.

Nicht nur zwei Klassen (upward, downward), sondern mehrere Klassen (bisschen hoch, sehr hoch, etc.) -> Classification of intraday S&P500 returns with a Random Forest Christoph Lohrmann, Pasi Luukka

A Anhang

A.1 Aktienverläufe absolut

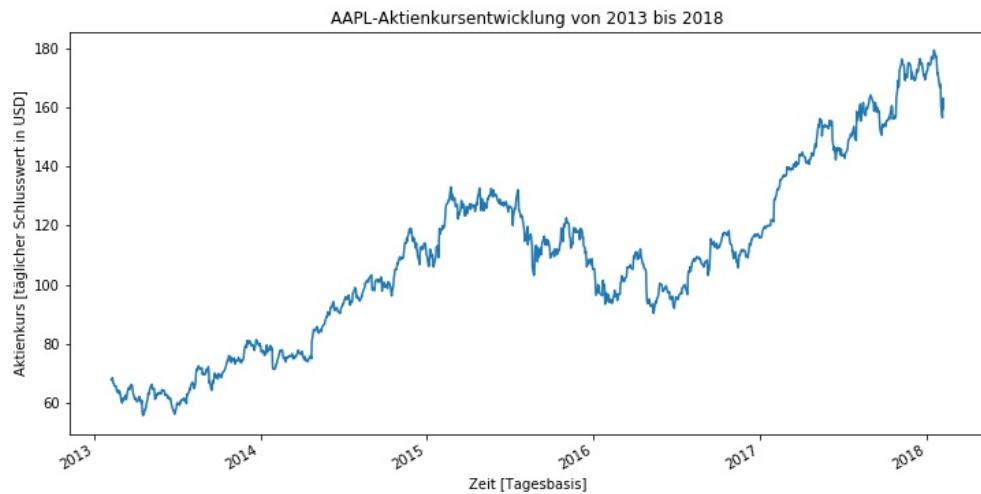


Abbildung A.1 TBD



Abbildung A.2 TBD

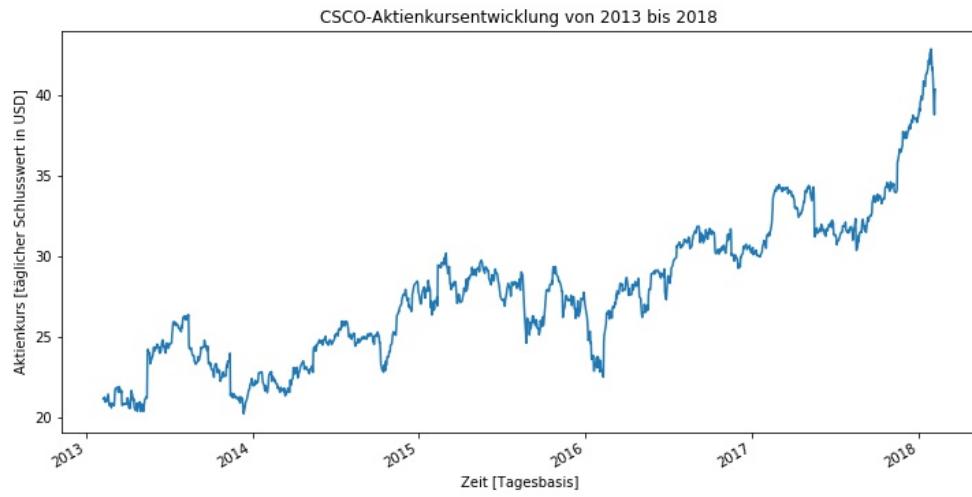


Abbildung A.3 TBD

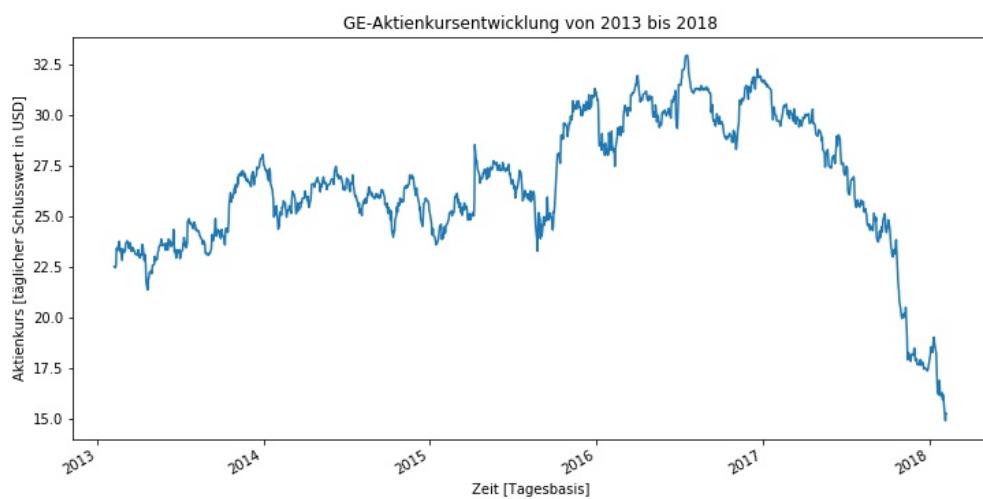


Abbildung A.4 TBD

weitere Datensets..

A.2 Auswertungen der Klassifikatoren: F-Maße pro Aktie pro Horizont als Diagramme

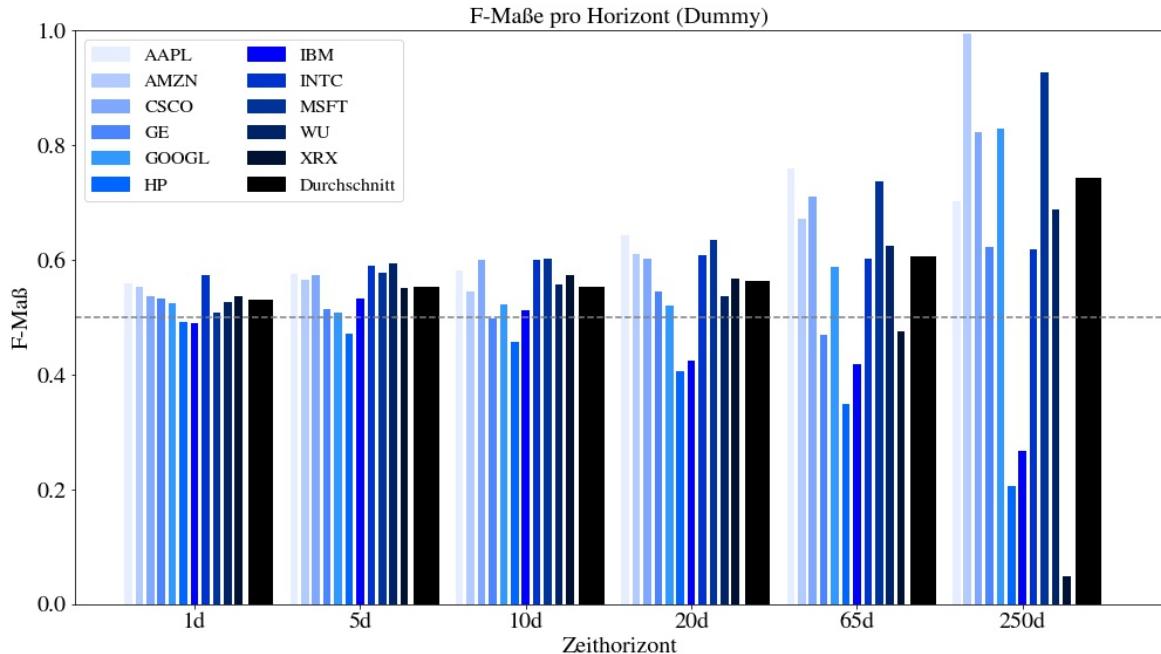


Abbildung A.5 TBD

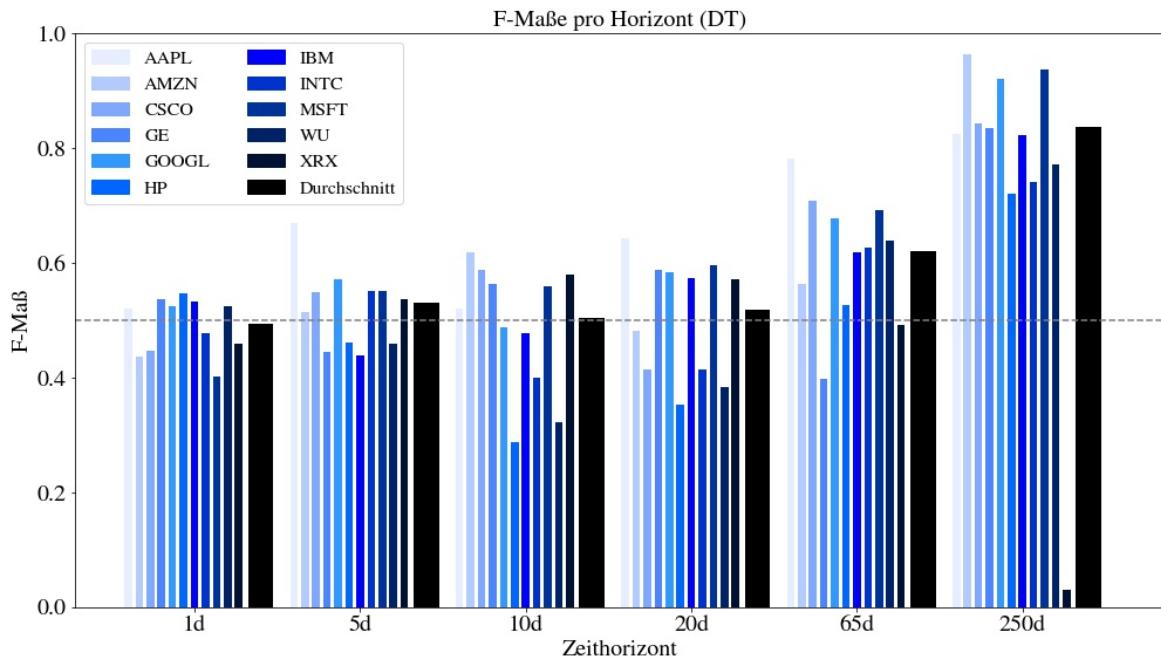


Abbildung A.6 TBD

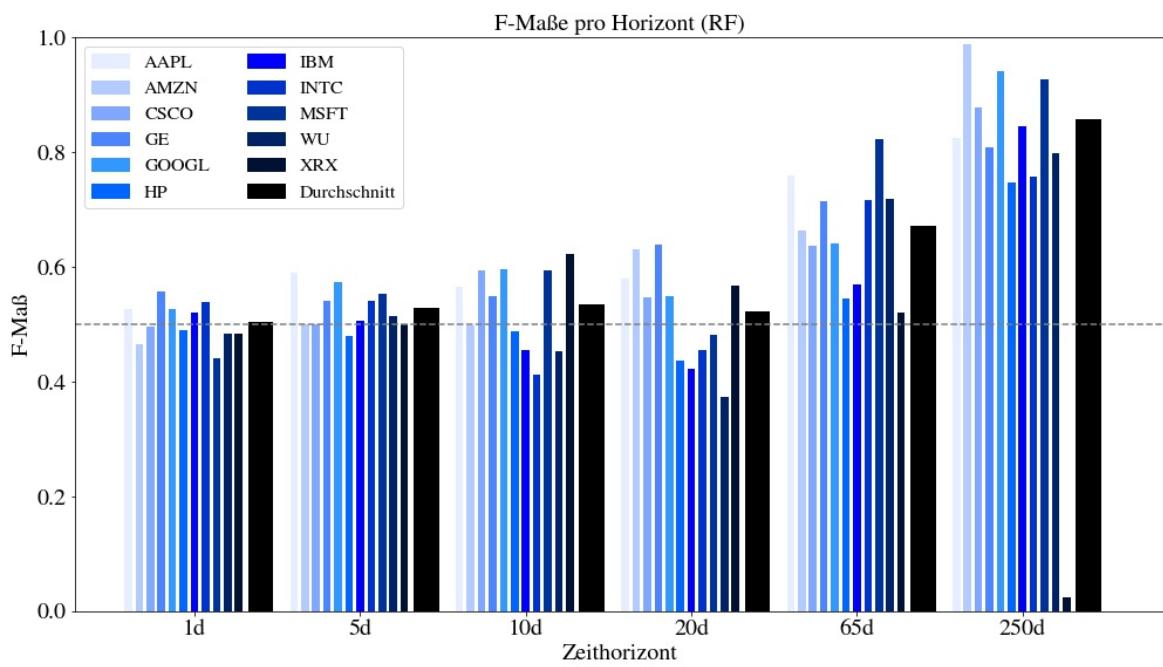


Abbildung A.7 TBD

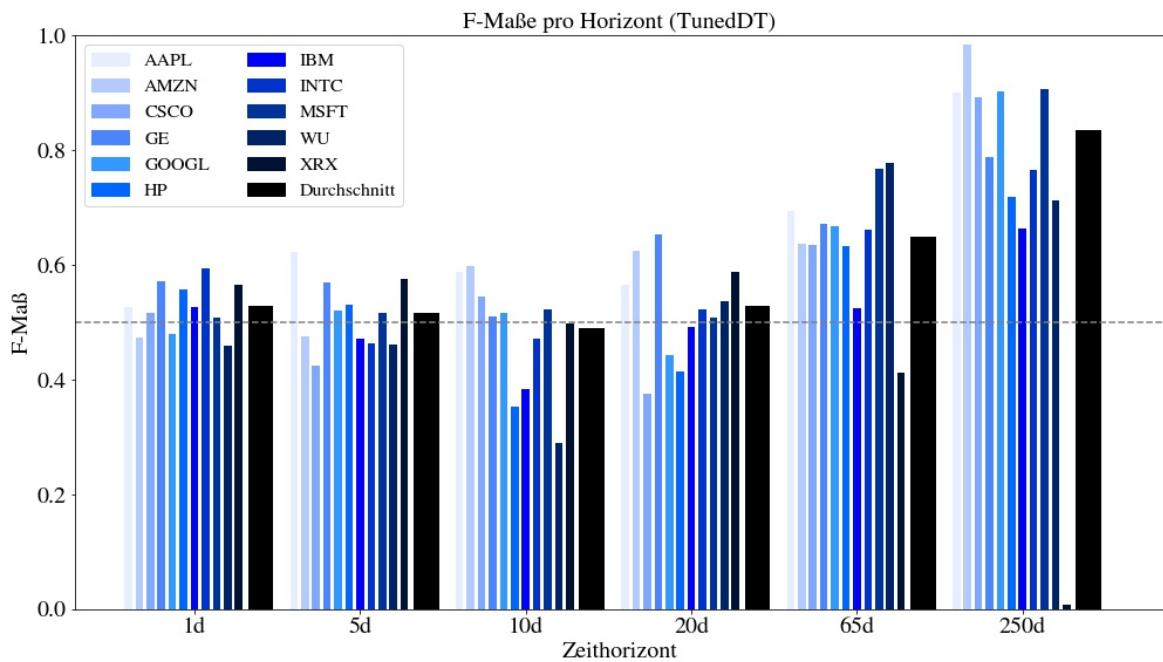


Abbildung A.8 TBD

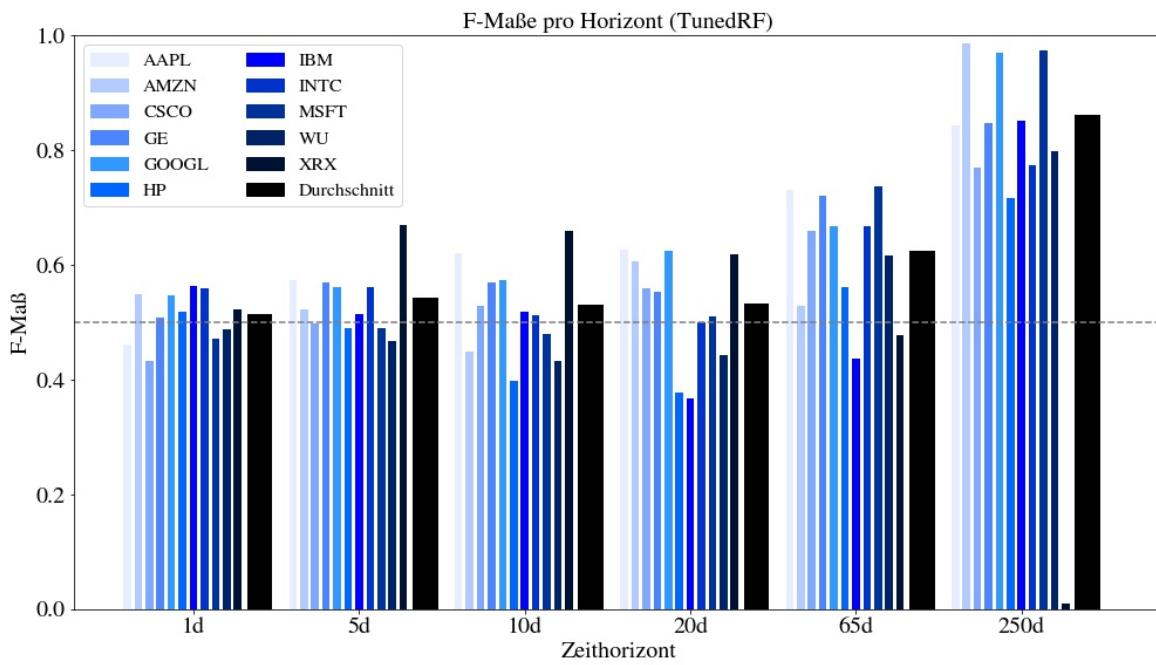


Abbildung A.9 TBD

A.3 Auswertungen der Klassifikatoren: Precision, Recall und F-Maß in Tabellen

Tabelle A.1 TBD

Horizont		Durchschnitt über alle Datensets					
		1d	5d	10d	20d	60d	250d
Precision	Dummy	0,519	0,531	0,538	0,540	0,582	0,743
	DT	0,514	0,553	0,551	0,540	0,632	0,845
	RF	0,515	0,553	0,571	0,542	0,658	0,885
	TunedDT	0,515	0,547	0,556	0,550	0,637	0,836
	TunedRF	0,521	0,539	0,556	0,550	0,618	0,879
Recall	Dummy	0,543	0,576	0,570	0,589	0,633	0,741
	DT	0,476	0,509	0,463	0,497	0,611	0,828
	RF	0,493	0,505	0,501	0,503	0,686	0,832
	TunedDT	0,544	0,488	0,438	0,508	0,662	0,835
	TunedRF	0,507	0,548	0,507	0,519	0,632	0,845
F-Maß	Dummy	0,531	0,552	0,554	0,563	0,607	0,742
	DT	0,495	0,530	0,503	0,518	0,621	0,836
	RF	0,504	0,528	0,534	0,522	0,672	0,858
	TunedDT	0,529	0,516	0,490	0,528	0,649	0,835
	TunedRF	0,514	0,543	0,530	0,534	0,625	0,862

Tabelle A.2 TBD

AAPL		Horizont	1d	5d	10d	20d	60d	250d
Precision	Dummy	0,533	0,536	0,546	0,599	0,686	0,656	
	DT	0,515	0,576	0,575	0,602	0,802	0,905	
	RF	0,508	0,565	0,578	0,647	0,691	0,867	
	TunedDT	0,524	0,571	0,589	0,547	0,685	0,879	
	TunedRF	0,529	0,533	0,588	0,622	0,671	0,884	
Recall	Dummy	0,588	0,621	0,621	0,695	0,850	0,756	
	DT	0,525	0,797	0,474	0,692	0,764	0,756	
	RF	0,545	0,616	0,551	0,525	0,843	0,785	
	TunedDT	0,528	0,685	0,587	0,584	0,704	0,925	
	TunedRF	0,410	0,624	0,659	0,633	0,805	0,807	
F-Maß	Dummy	0,559	0,575	0,581	0,643	0,759	0,703	
	DT	0,520	0,669	0,520	0,644	0,782	0,824	
	RF	0,526	0,589	0,564	0,580	0,759	0,824	
	TunedDT	0,526	0,623	0,588	0,565	0,694	0,901	
	TunedRF	0,462	0,575	0,622	0,628	0,732	0,844	

Tabelle A.3 TBD

AMZN		Horizont	1d	5d	10d	20d	60d	250d
Precision	Dummy	0,533	0,582	0,580	0,686	0,762	1,000	
	DT	0,555	0,606	0,603	0,646	0,649	1,000	
	RF	0,562	0,599	0,583	0,667	0,706	1,000	
	TunedDT	0,571	0,565	0,628	0,657	0,670	1,000	
	TunedRF	0,559	0,612	0,528	0,672	0,634	1,000	
Recall	Dummy	0,573	0,550	0,513	0,550	0,599	0,987	
	DT	0,359	0,448	0,635	0,385	0,496	0,928	
	RF	0,397	0,428	0,436	0,599	0,625	0,976	
	TunedDT	0,405	0,410	0,572	0,597	0,607	0,969	
	TunedRF	0,541	0,455	0,392	0,552	0,454	0,975	
F-Maß	Dummy	0,552	0,565	0,545	0,611	0,670	0,993	
	DT	0,436	0,515	0,618	0,483	0,562	0,963	
	RF	0,465	0,499	0,499	0,631	0,663	0,988	
	TunedDT	0,474	0,476	0,599	0,625	0,637	0,984	
	TunedRF	0,550	0,522	0,450	0,606	0,529	0,987	

Tabelle A.4 TBD

CSCO		1d	5d	10d	20d	60d	250d
Horizont		1d	5d	10d	20d	60d	250d
Precision	Dummy	0,527	0,563	0,571	0,555	0,657	0,812
	DT	0,496	0,599	0,631	0,506	0,786	0,862
	RF	0,483	0,545	0,649	0,615	0,728	0,966
	TunedDT	0,480	0,547	0,613	0,534	0,710	0,886
	TunedRF	0,480	0,553	0,578	0,665	0,726	0,853
Recall	Dummy	0,547	0,586	0,631	0,659	0,774	0,832
	DT	0,406	0,506	0,550	0,349	0,644	0,825
	RF	0,509	0,463	0,547	0,493	0,565	0,805
	TunedDT	0,561	0,347	0,490	0,291	0,576	0,898
	TunedRF	0,395	0,455	0,487	0,483	0,604	0,704
F-Maß	Dummy	0,537	0,574	0,600	0,602	0,710	0,822
	DT	0,447	0,549	0,587	0,413	0,708	0,843
	RF	0,496	0,501	0,594	0,547	0,636	0,878
	TunedDT	0,518	0,425	0,544	0,377	0,636	0,892
	TunedRF	0,433	0,499	0,528	0,560	0,659	0,771

Tabelle A.5 TBD

GE		1d	5d	10d	20d	60d	250d
Horizont		1d	5d	10d	20d	60d	250d
Precision	Dummy	0,514	0,495	0,471	0,485	0,459	0,557
	DT	0,520	0,524	0,554	0,558	0,412	0,849
	RF	0,526	0,567	0,557	0,627	0,692	0,738
	TunedDT	0,511	0,541	0,544	0,619	0,639	0,721
	TunedRF	0,509	0,576	0,568	0,580	0,702	0,809
Recall	Dummy	0,552	0,536	0,529	0,620	0,482	0,706
	DT	0,555	0,388	0,574	0,620	0,383	0,821
	RF	0,590	0,516	0,543	0,650	0,740	0,894
	TunedDT	0,651	0,601	0,480	0,692	0,710	0,868
	TunedRF	0,509	0,563	0,571	0,530	0,740	0,891
F-Maß	Dummy	0,532	0,515	0,498	0,544	0,470	0,623
	DT	0,537	0,446	0,564	0,587	0,397	0,835
	RF	0,556	0,540	0,550	0,638	0,715	0,808
	TunedDT	0,573	0,569	0,510	0,653	0,672	0,788
	TunedRF	0,509	0,569	0,570	0,554	0,720	0,848

Tabelle A.6 TBD

GOOGL		1d	5d	10d	20d	60d	250d
Horizont		1d	5d	10d	20d	60d	250d
Precision	Dummy	0,518	0,524	0,571	0,576	0,621	0,975
	DT	0,533	0,553	0,572	0,580	0,689	0,947
	RF	0,545	0,577	0,630	0,628	0,730	0,990
	TunedDT	0,593	0,539	0,601	0,561	0,659	0,935
	TunedRF	0,560	0,587	0,627	0,642	0,721	0,992
Recall	Dummy	0,534	0,495	0,482	0,473	0,559	0,721
	DT	0,514	0,592	0,426	0,586	0,667	0,897
	RF	0,511	0,570	0,564	0,488	0,572	0,897
	TunedDT	0,402	0,505	0,454	0,366	0,676	0,875
	TunedRF	0,536	0,541	0,531	0,609	0,623	0,951
F-Maß	Dummy	0,525	0,509	0,523	0,520	0,588	0,829
	DT	0,523	0,572	0,488	0,583	0,678	0,921
	RF	0,527	0,573	0,595	0,550	0,642	0,941
	TunedDT	0,479	0,522	0,517	0,443	0,667	0,904
	TunedRF	0,548	0,563	0,575	0,625	0,668	0,971

weitere Datensets..

A.4 Auswertungen der Klassifikatoren pro Gruppe

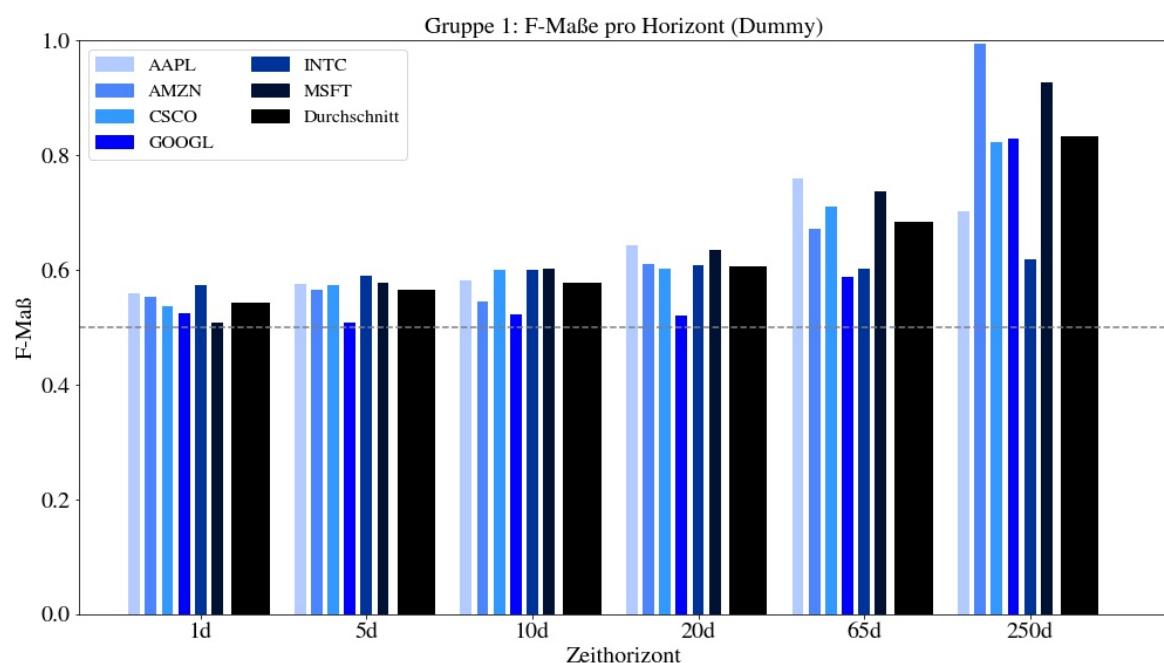


Abbildung A.10 TBD

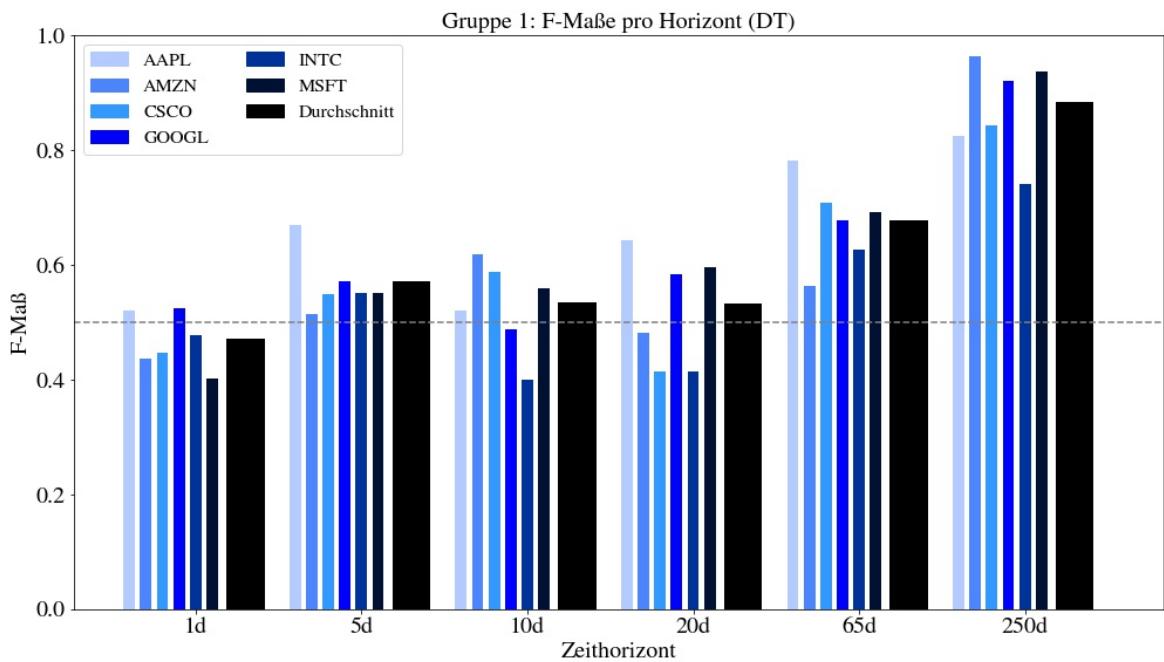


Abbildung A.11 TBD

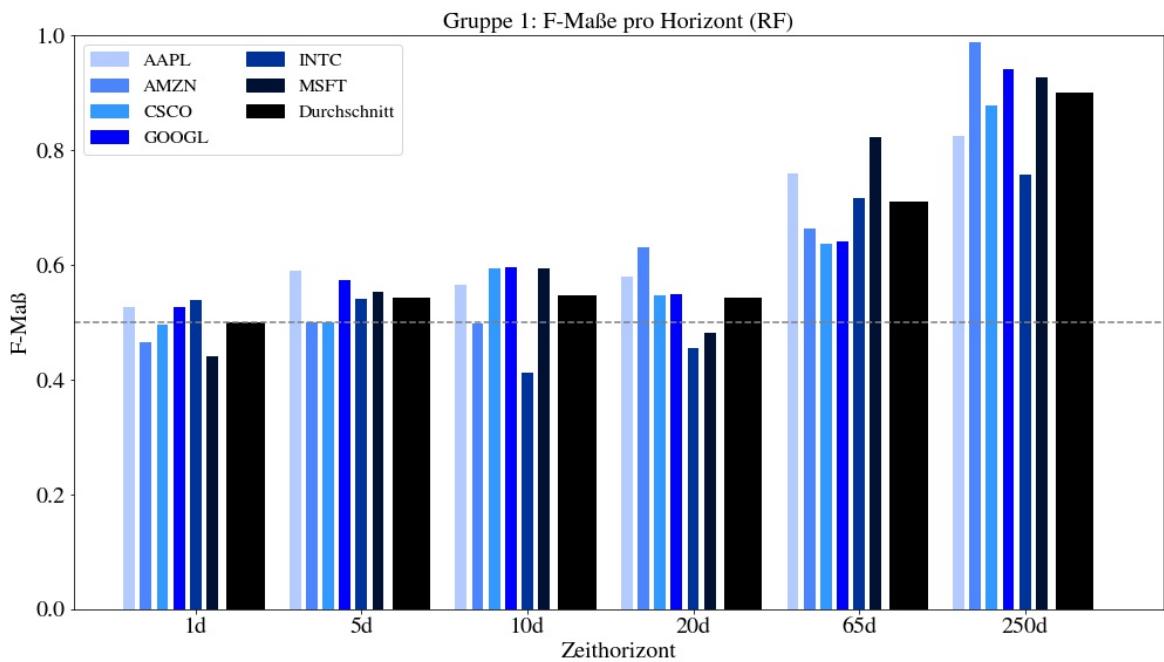


Abbildung A.12 TBD

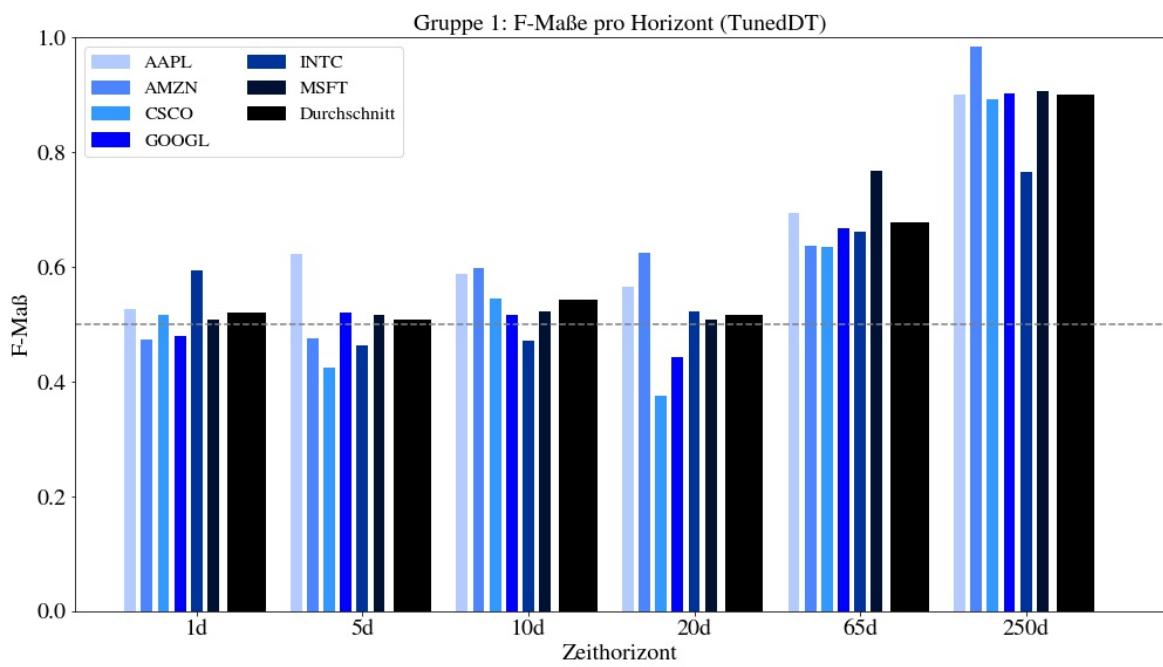


Abbildung A.13 TBD

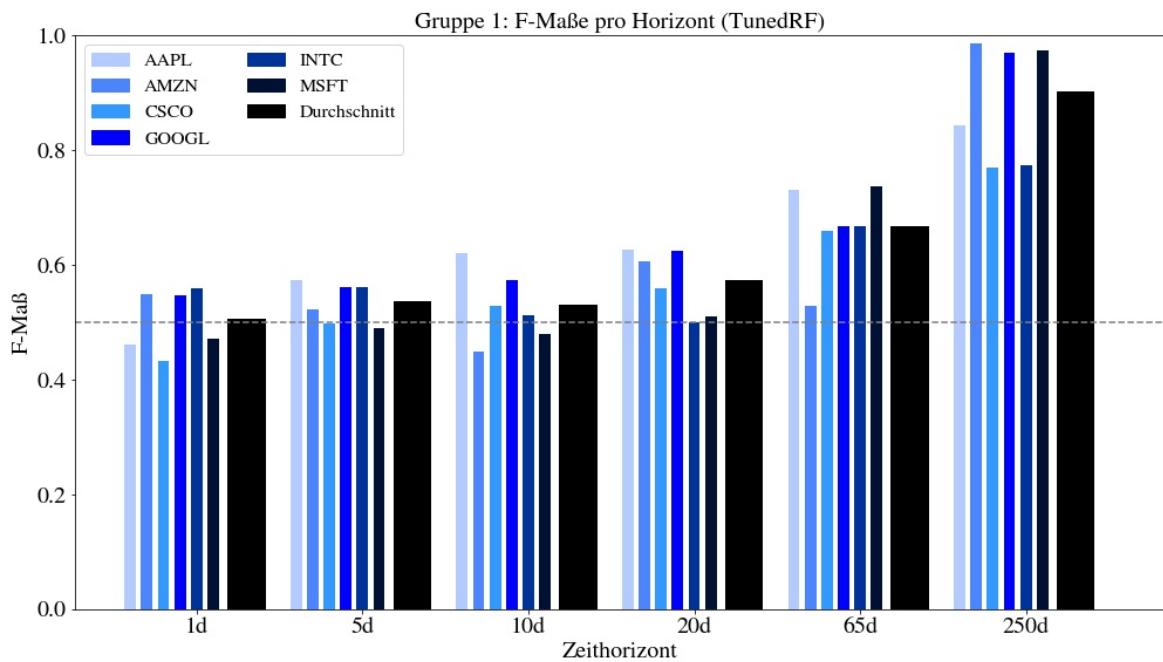


Abbildung A.14 TBD

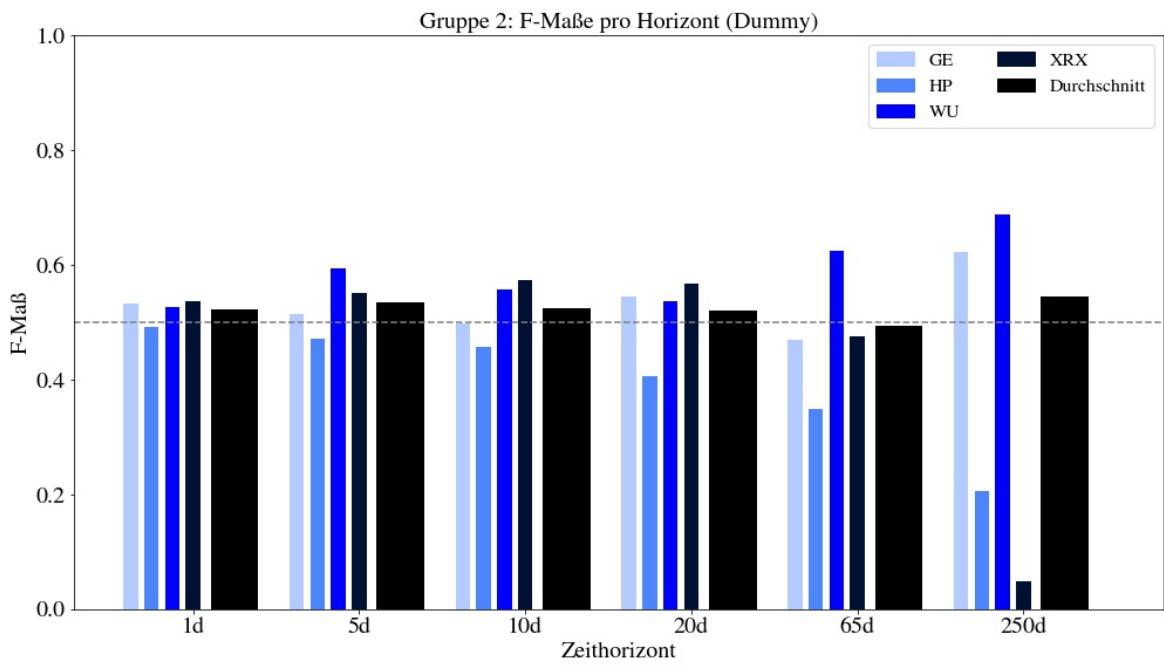


Abbildung A.15 TBD

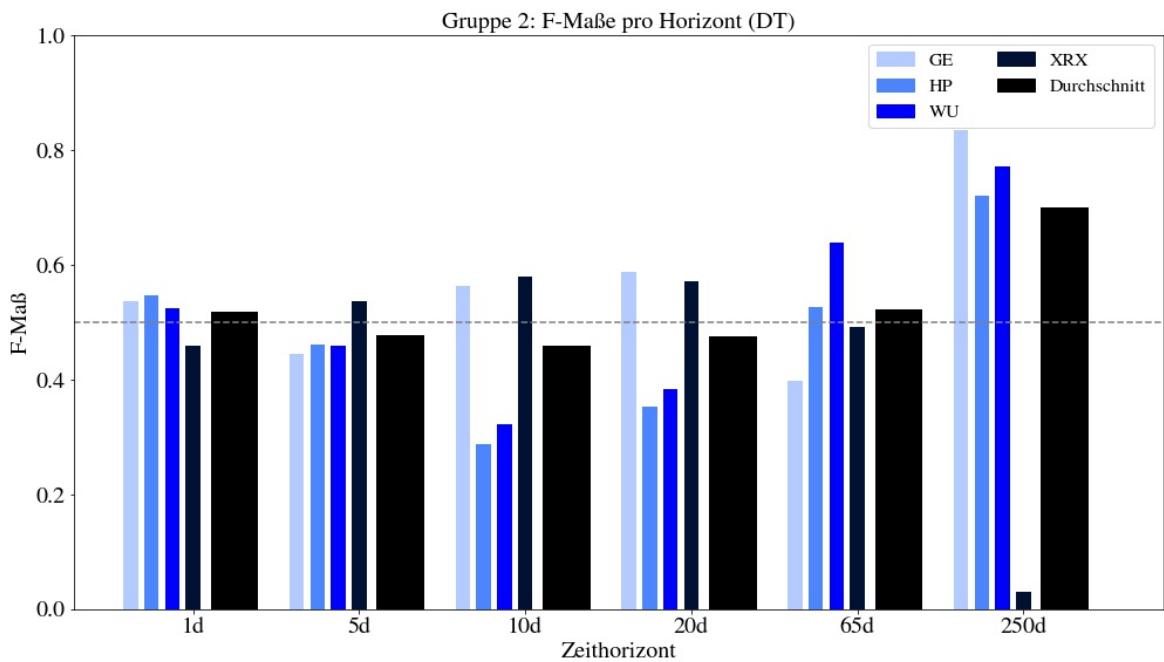


Abbildung A.16 TBD

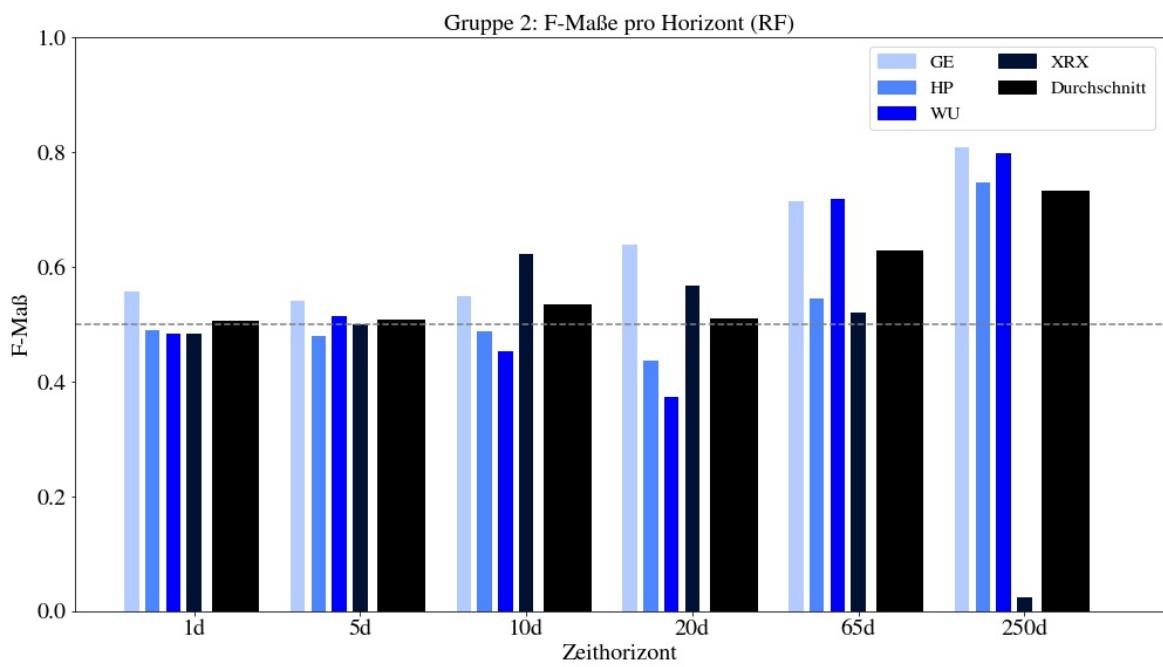


Abbildung A.17 TBD

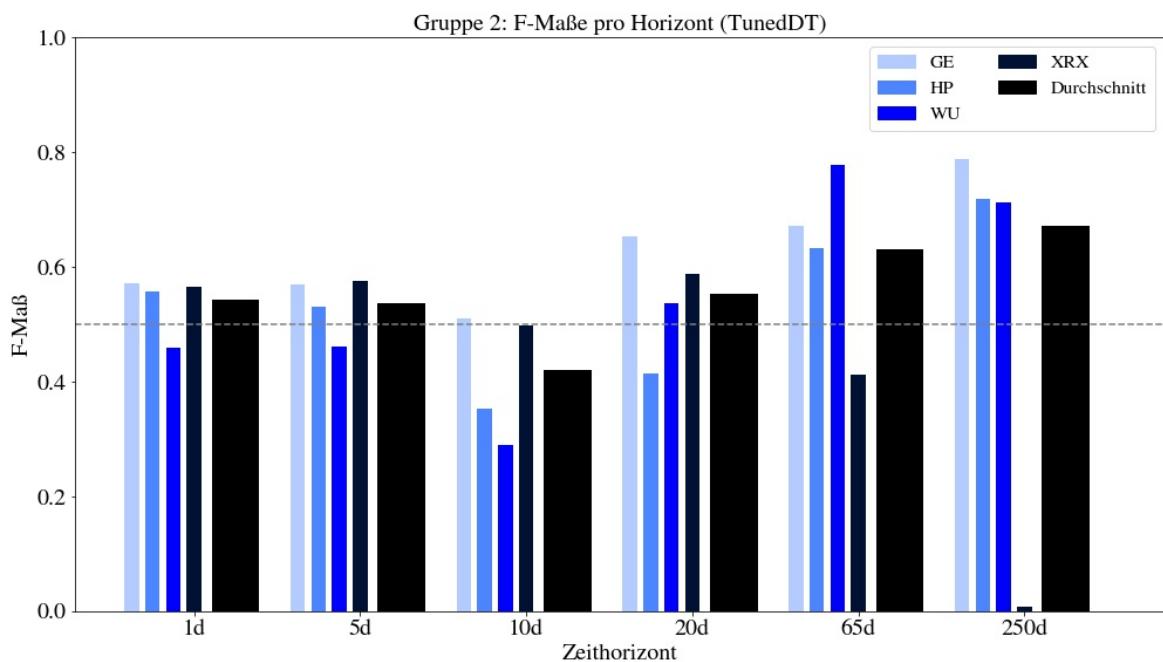


Abbildung A.18 TBD

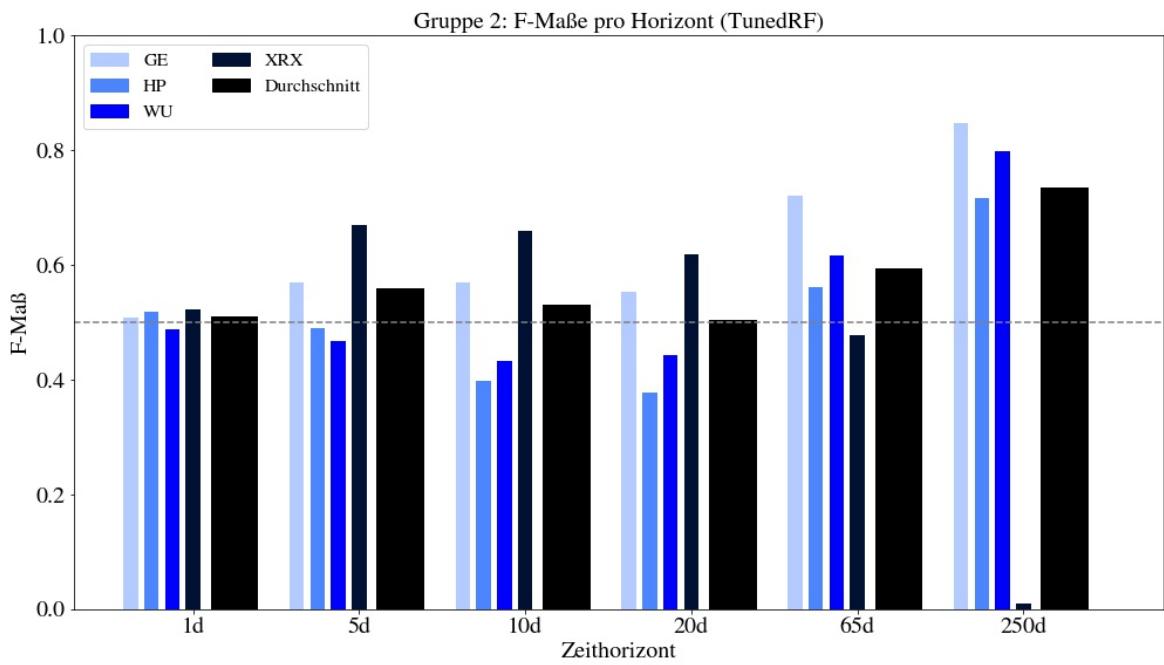


Abbildung A.19 TBD

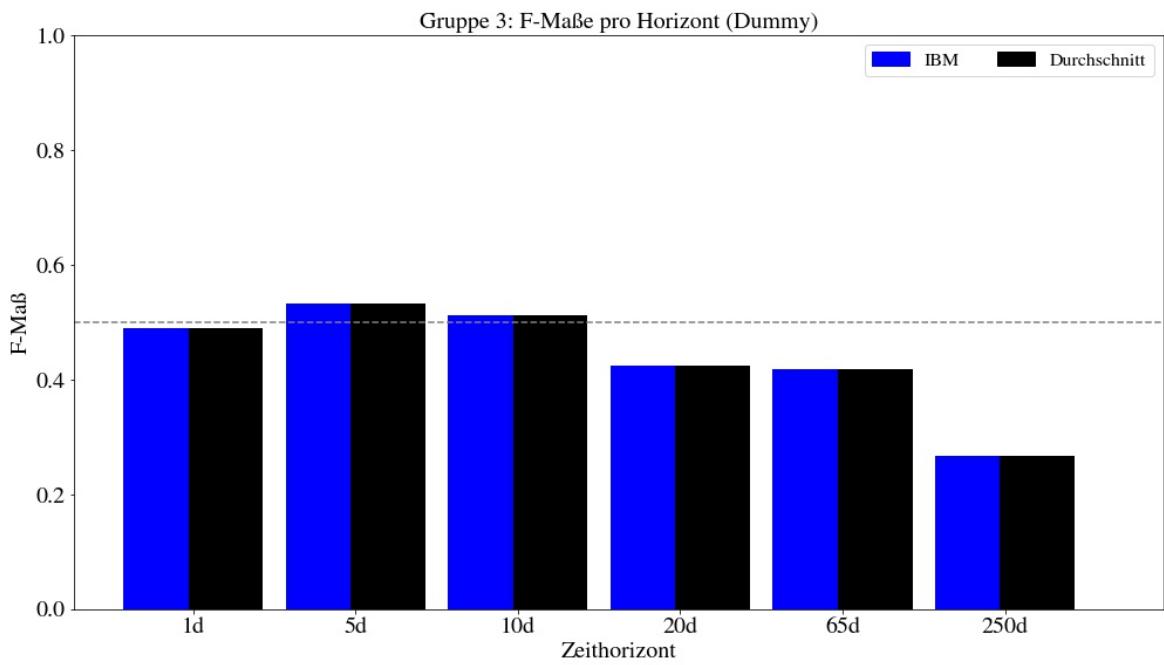


Abbildung A.20 TBD

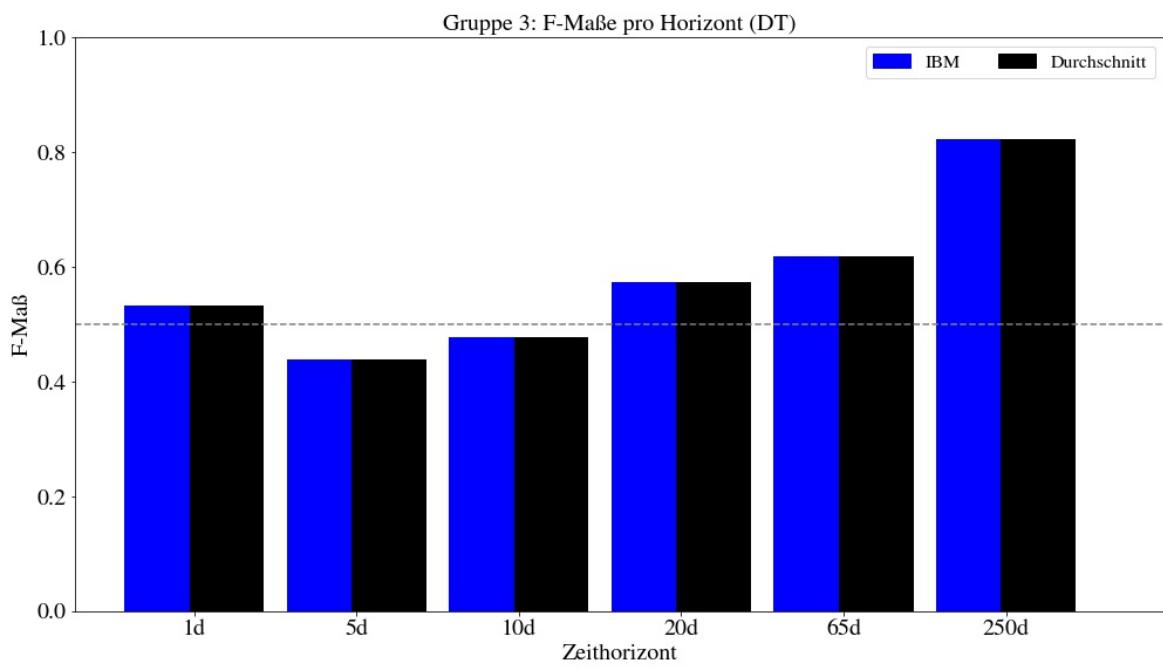


Abbildung A.21 TBD

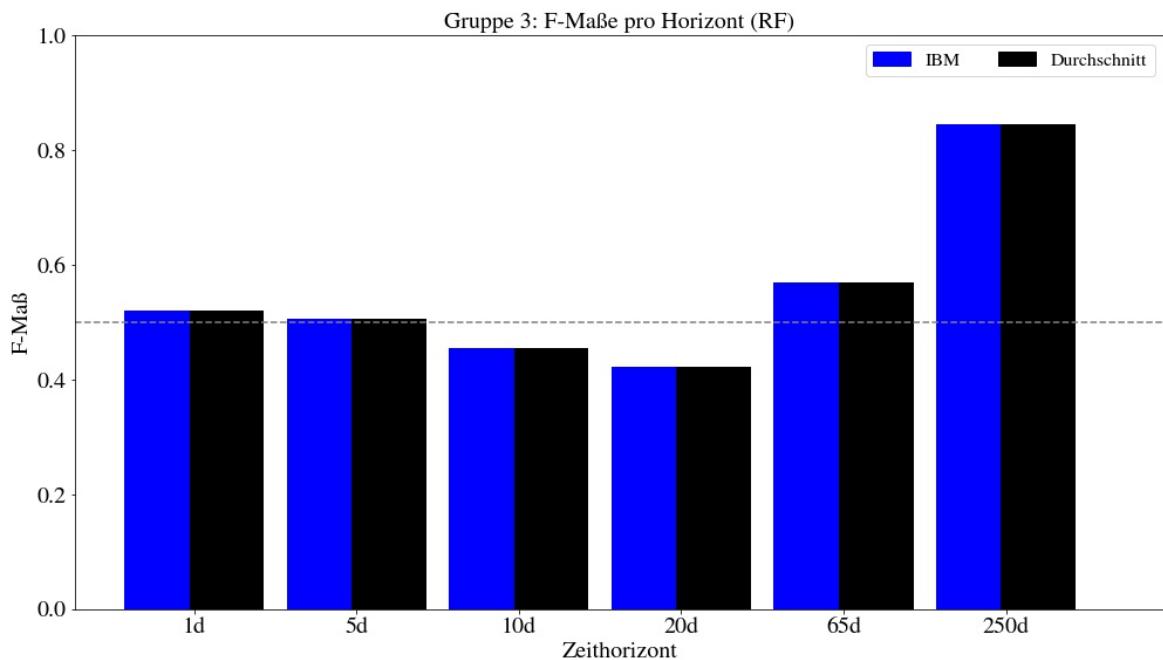


Abbildung A.22 TBD

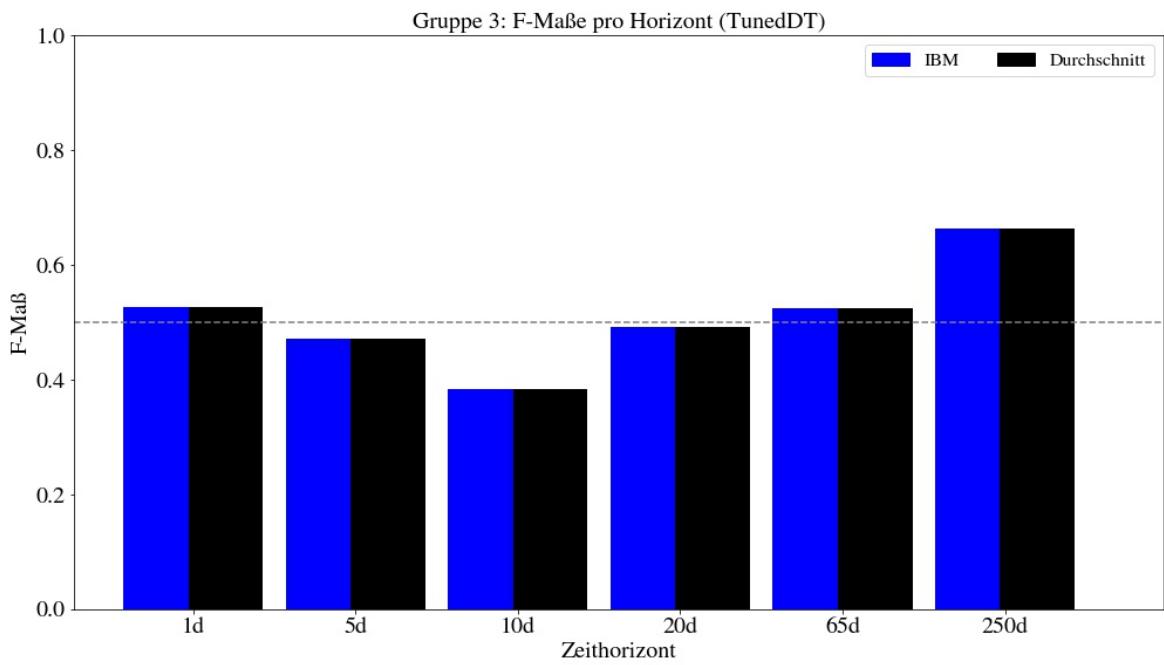


Abbildung A.23 TBD

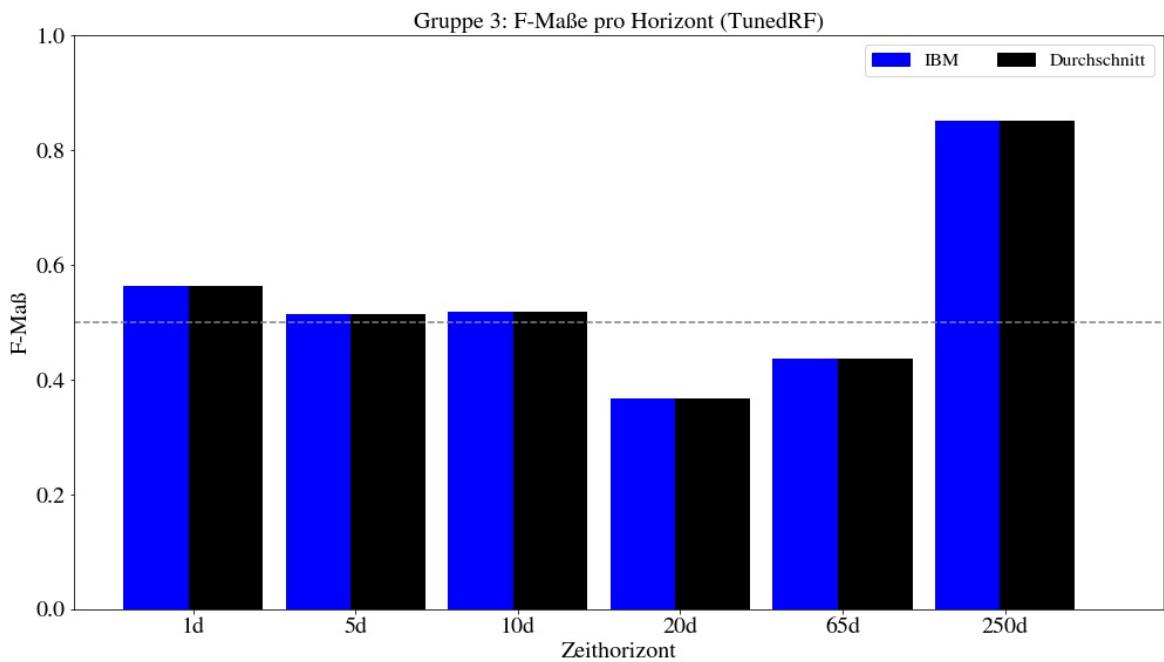


Abbildung A.24 TBD

A.5 Tuning Einfluss pro Gruppe ergänzen?

A.6 Jupyter Code? Welche wichtigste Funktionen?

TBD: LaTex-Export aus Jupyter laden

Literaturverzeichnis

- [Abdou u. Pointon 2011] ABDOU, Hussein ; POINTON, John: Credit Scoring, Statistical Techniques and Evaluation Criteria: A Review of the Literature. In: *Int. Syst. in Accounting, Finance and Management* 18 (2011), 04, S. 59–88. <http://dx.doi.org/10.1002/isaf.325>. – DOI 10.1002/isaf.325
- [Alavi u. a. 2015] ALAVI, Seyed E. ; SINAEI, Hasanali ; AFSHARIRAD, Elham: Predict the trend of stock prices using machine learning techniques. In: *International Academic Journal of Economics* 2 (2015), S. 1–11
- [Alpaydin 2008] ALPAYDIN, E.: *Maschinelles Lernen*. Oldenbourg, 2008. – ISBN 9783486581140
- [Bergmeir u. Benitez 2011] BERGMEIR, C. ; BENITEZ, J. M.: Forecaster performance evaluation with cross-validation and variants. In: *2011 11th International Conference on Intelligent Systems Design and Applications*, 2011, S. 849–854
- [BlackRock 2019a] BLACKROCK, Inc.: *Aladdin - Benefits to risk managers*. <https://www.blackrock.com/aladdin/benefits/risk-managers>, 2019. – Besucht: 06. Oktober 2019
- [BlackRock 2019b] BLACKROCK, Inc.: *Risikomanagement mit Aladdin*. <https://www.blackrock.com/at/finanzberater-und-banken/uber-blackrock/risk-management-with-aladdin?switchLocale=y&siteEntryPassthrough=true>, 2019. – Besucht: 06. Oktober 2019
- [Breiman 2001] BREIMAN, Leo: Random Forests. In: *Machine Learning* 45 (2001), Oktober, Nr. 1, S. 5–32. <http://dx.doi.org/10.1023/A:1010933404324>. – DOI 10.1023/A:1010933404324. – ISSN 0885–6125
- [Cerqueira u. a. 2019] CERQUEIRA, Vitor ; TORGÓ, Luis ; MOZETIC, Igor: *Evaluating time series forecasting models: An empirical study on performance estimation methods*. 2019
- [Cutler u. a. 2007] CUTLER, D. R. ; EDWARDS, Thomas C. ; BEARD, Karen H. ; CUTLER, Adele ; HESS, Kyle ; GIBSON, Jacob ; LAWLER, Joshua J.: Random forests for classification in ecology. In: *Ecology* 88 11 (2007), S. 2783–92
- [Di 2014] DI, Xinjie: Stock Trend Prediction with Technical Indicators using SVM / Stanford University. 2014. – Forschungsbericht
- [Diaz-Uriarte u. Alvarez 2006] DIAZ-URIARTE, Ramon ; ALVAREZ, Sara: Gene Selection and Classification of Microarray Data Using Random Forest. In: *BMC bioinformatics* 7 (2006), 02, S. 3. <http://dx.doi.org/10.1186/1471-2105-7-3>. – DOI 10.1186/1471-2105-7-3
- [Drakopoulou 2016] DRAKOPOULOU, Veliota: A Review of Fundamental and Technical Stock Analysis Techniques. In: *Journal of Stock & Forex Trading* 05 (2016), 01. <http://dx.doi.org/10.4172/2168-9458.1000163>. – DOI 10.4172/2168-9458.1000163

- [Fama 1965] FAMA, Eugene F.: Random Walks in Stock-Market Prices. In: *Financial Analysts Journal* 21 (1965), S. 55–59
- [Feurer u. Hutter 2019] FEURER, Matthias ; HUTTER, Frank: *Automated Machine Learning - Methods, Systems, Challenges*. Springer International Publishing, 2019. – ISBN 978-3-030-05318-5
- [Freund u. Schapire 1997] FREUND, Yoav ; SCHAPIRE, Robert E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In: *Journal of Computer and System Sciences* 55 (1997), Nr. 1, S. 119 – 139. <http://dx.doi.org/https://doi.org/10.1006/jcss.1997.1504>. – DOI <https://doi.org/10.1006/jcss.1997.1504>. – ISSN 0022-0000
- [Gron 2017] GRON, Aurlien: *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st. O'Reilly Media, Inc., 2017. – ISBN 1491962291, 9781491962299
- [Gupta u. a. 2017] GUPTA, Bhumika ; RAWAT, Aditya ; JAIN, Akshay ; ARORA, Arpit ; DHAMI, Naresh: Analysis of Various Decision Tree Algorithms for Classification in Data Mining. In: *International Journal of Computer Applications* 163 (2017), 04, S. 15–19. <http://dx.doi.org/10.5120/ijca2017913660>. – DOI 10.5120/ijca2017913660
- [Howard u. Bowles 2012] HOWARD, Jeremy ; BOWLES, Mike: *The Two Most Important Algorithms in Predictive Modeling Today*. <https://conferences.oreilly.com/strata/strata2012/public/schedule/detail/22658>, 2012. – Besucht: 19. Oktober 2019
- [Khaidem u. a. 2016] KHAIDEM, Luckyson ; SAHA, Snehanshu ; BASAK, Suryoday ; KAR, Saibal ; DEY, Sudeepa: Predicting the direction of stock market prices using random forest. In: *Applied Mathematical Finance* (2016), 04
- [Lendasse u. a. 2002] LENDASSE, Amaury ; BODT, E. ; WERTZ, Vincent ; VERLEYSEN, Michel: Non-linear financial time series forecasting - Application to the Bel 20 stock market index. In: <http://dx.doi.org/10.1051/ejess:2000110> 14 (2002), 11. <http://dx.doi.org/10.1051/ejess:2000110>. – DOI 10.1051/ejess:2000110
- [Li 2016] LI, C.: The application of high-dimensional data classification by random forest based on hadoop cloud computing platform. 51 (2016), 01, S. 385–390. <http://dx.doi.org/10.3303/CET1651065>. – DOI 10.3303/CET1651065
- [Lohrmann u. Luukka 2019] LOHRMANN, Christoph ; LUUKKA, Pasi: Classification of intraday S&P500 returns with a Random Forest. In: *International Journal of Forecasting* 35 (2019), Nr. 1, S. 390–407. <http://dx.doi.org/10.1016/j.ijforecast.2018.10.016>. – DOI 10.1016/j.ijforecast.2018
- [McCarthy u. a. 1955] McCARTHY, J. ; MINSKY, M. L. ; ROCHESTER, N. ; SHANNON, C. E.: *A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE*. <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>. <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>. Version: 1955
- [Mitchell 1997] MITCHELL, Thomas M.: *Machine Learning*. 1. New York, NY, USA : McGraw-Hill, Inc., 1997. – ISBN 0070428077, 9780070428072

- [Pasupulety u. a. 2019] PASUPULETY, U. ; ABDULLAH ANEES, A. ; ANMOL, S. ; MOHAN, B. R.: Predicting Stock Prices using Ensemble Learning and Sentiment Analysis. In: *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2019, S. 215–222
- [Pedregosa u. a. 2011] PEDREGOSA, Fabian ; VAROQUAUX, Gaël ; GRAMFORT, Alexandre ; MICHEL, Vincent ; THIRION, Bertrand ; GRISEL, Olivier ; BLONDEL, Mathieu ; PRETTENHOFER, Peter ; WEISS, Ron ; DUBOURG, Vincent ; VANDERPLAS, Jake ; PASSOS, Alexandre ; COURNAPEAU, David ; BRUCHER, Matthieu ; PERROT, Matthieu ; DUCHESNAY, Édouard: Scikit-learn: Machine Learning in Python. In: *J. Mach. Learn. Res.* 12 (2011), November, S. 2825–2830. – ISSN 1532–4435
- [Prasad u. a. 2006] PRASAD, Anantha ; IVERSON, Louis ; LIAW, Andy: Newer Classification and Regression Tree Techniques: Bagging and Random Forests for Ecological Prediction. In: *Ecosystems* 9 (2006), 03, S. 181–199. <http://dx.doi.org/10.1007/s10021-005-0054-1>. – DOI 10.1007/s10021-005-0054-1
- [Probst u. a. 2018] PROBST, Philipp ; BOULESTEIX, Anne-Laure ; WRIGHT, Marvin: *Hyperparameters and Tuning Strategies for Random Forest*. 04 2018
- [Quinlan 1986] QUINLAN, J. R.: Induction of decision trees. In: *Machine Learning* 1 (1986), Mar, Nr. 1, S. 81–106. <http://dx.doi.org/10.1007/BF00116251>. – DOI 10.1007/BF00116251
- [Russell u. Norvig 2009] RUSSELL, Stuart ; NORVIG, Peter: *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ, USA : Prentice Hall Press, 2009. – ISBN 0136042597, 9780136042594
- [Sadia u. a. 2019] SADIA, K. H. ; SHARMA, Aditya ; PAUL, Adarrsh ; PADHI, Sarmistha ; SANYAL, Saurav: Stock Market Prediction Using Machine Learning Algorithms. In: *International Journal of Engineering and Advanced Technology* 8 (2019), 04
- [Shannon 1948] SHANNON, Claude E.: A Mathematical Theory of Communication. In: *The Bell System Technical Journal* 27 (1948), 7, Nr. 3, S. 379–423. <http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x>. – DOI 10.1002/j.1538-7305.1948.tb01338.x
- [Shoham u. a. 2018] SHOHAM, Yoav ; PERRAULT, Raymond ; BRYNJOLFSSON, Erik u. a.: The AI Index 2018 Annual Report / AI Index Steering Committee, Human-Centered AI Initiative, Stanford University, Stanford, CA. 2018. – Forschungsbericht
- [Shotton u. a. 2011] SHOTTON, J. ; FITZGIBBON, A. ; COOK, M. ; SHARP, T. ; FINOCCHIO, M. ; MOORE, R. ; KIPMAN, A. ; BLAKE, A.: Real-time Human Pose Recognition in Parts from Single Depth Images. In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2011 (CVPR '11). – ISBN 978-1-4577-0394-2, S. 1297–1304
- [Silver u. a. 2017] SILVER, David ; SCHRITTWIESER, Julian ; SIMONYAN, Karen ; ANTONOGLOU, Ioannis ; HUANG, Aja ; GUEZ, Arthur ; HUBERT, Thomas ; BAKER, L R. ; LAI, Matthew ; BOLTON, Adrian ; CHEN, Yutian ; LILLICRAP, Timothy P. ; HUI, Fan Fong C. ; SIFRE, Laurent ; DRIESSCHE, George van d. ; GRAEPEL, Thore ; HASSABIS, Demis: Mastering the game of Go without human knowledge. In: *Nature* 550 (2017), S. 354–359

[von Statista herausfinden 2019] STATISTA HERAUSFINDEN, Statista TBD: Q.: *Largest asset management companies worldwide as of March 2019, by managed assets (in trillion U.S. dollars)*. <https://www.statista.com/statistics/431790/leading-asset-management-companies-worldwide-by-assets/>, 2019. – Besucht: 06. Oktober 2019

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

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

[Tin Kam Ho 1998] TIN KAM HO: The random subspace method for constructing decision forests. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), Aug, Nr. 8, S. 832–844. <http://dx.doi.org/10.1109/34.709601>. – DOI 10.1109/34.709601

[Turney 2002] TURNLEY, Peter: Types of Cost in Inductive Concept Learning. In: *CoRR* cs.LG/0212034 (2002), 12

[Verleysen u. François 2005] VERLEYSSEN, Michel ; FRANÇOIS, Damien: The Curse of Dimensionality in Data Mining and Time Series Prediction, 2005, S. 758–770

[Whitley u. Watson 2005] WHITLEY, L. D. ; WATSON, Jean-Paul: Complexity Theory and the No Free Lunch Theorem, 2005

[Wolpert 1992] WOLPERT, David H.: Stacked generalization. In: *Neural Networks* 5 (1992), Nr. 2, S. 241 – 259. [http://dx.doi.org/https://doi.org/10.1016/S0893-6080\(05\)80023-1](http://dx.doi.org/https://doi.org/10.1016/S0893-6080(05)80023-1). – DOI [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1). – ISSN 0893-6080