Methodik, Anwendung und Interpretation moderner Benchmark-Studien am Beispiel der Risikomodellierung bei akuter Cholangitis

Anna Theresa Stüber

Masterarbeit in Biostatistik

München 2019

Methodik, Anwendung und Interpretation moderner Benchmark-Studien am Beispiel der Risikomodellierung bei akuter Cholangitis

Anna Theresa Stüber

Masterarbeit in Biostatistik

zur Erlangung des akademischen Grades
eines Masters of Science (M. Sc.)

am Institut für Statistik
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität, München

vorgelegt von
Anna Theresa Stüber

München, den 14. Juli 2019

Erstgutachter:	Prof. Dr. Bernd Bischl Institut für Statistik, LMU
Zweitgutachter:	PD Dr. Alexander Hapfelmeier ImedIS, TUM
Disputation am:	29.07.2019

Danksagung

Hiermit möchte ich mich insbesondere bei Prof. Dr. Bernd Bischl bedanken, durch den mir die Möglichkeit zur Verfassung dieser Masterarbeit überhaupt erst bereitgestellt wurde. So konnte ich mein Interesse an Aspekten des maschinellen Lernens weitgehend vertiefen und erhielt durch sein umfangreiches Fachwissen, sowie entsprechende sachkundige Hilfestellungen die nötigen Richtungsweisungen in einem Wissenschaftsbereich von beträchtlichem Ausmaß.
Mein besonderer Dank geht an PD Dr. Alexander Hapfelmeier, der mir die Thematik bzw. den Inhalt dieser Arbeit im Zuge meines vorangehenden Praktikums am Institut für medizinische Statistik und Epidemiologie (ImedIS) unterbreitete. Hervorhebend möchte ich mich dabei für die permanente Verfügbarkeit bei Fragen unterschiedlichster Art, sowie die äußerst kompetente Betreuung erkenntlich zeigen.
Im Hinblick auf die Betreuung gilt es auch Stefan Coors meinen Dank auszusprechen, der mich seitens der LMU kurzerhand bei der Erstellung der als Teil dieser Arbeit inbegriffenen Shiny App fortwährend und mit großer Geduld unterstützt hat.
Auch möchte ich mich bei meiner Familie für den permanenten Rückhalt und kulinarische Zufriedenstellung bedanken, ebenso wie bei meinen Freunden, die es verstanden mich durch die ein oder andere Beachvolleyball-Session bei Laune zu halten.

Abstract

This thesis concentrates on the analysis of benchmark studies as a part of machine learning. Hereby the focus is set on measuring the performance of different machine learning methods such that information about the suitability of the competing methods can be provided. As these analyses require instruments, such as performance measures and hyperparameter-tuning, explanations concerning these issues are offered with specific focus on classification problems. Moreover, well-chosen machine learning methods are described, specifically those, which were mainly used in the developed benchmark study concerning acute cholangitis. The results of this analysis were used to develop a winner model that has been trained on the whole data to determine effects each independent variable has on in-hospital mortality. In the process of these analyses, the complexity of benchmark studies established along with the need for opportunities to visualize the contained results, which lead to the implementation of a shiny app. Beside the aspects mentioned before, this app also includes techniques for making the applied (black box) methods interpretable.

Die vorliegende Arbeit befasst sich spezifisch mit der Analyse von Benchmark-Studien im Rahmen des maschinellen Lernens. Hierbei wird das Ziel verfolgt, die Performance unterschiedlicher ML-Methoden in Form einer Wettbewerbs-Analyse zu messen und die Eignung der jeweiligen Methodik hinsichtlich gegebener Daten beurteilen zu können. In diesem Zuge können eine Reihe von Instrumentarien wie etwa Gütemaße zur Beurteilung der Leistungsfähigkeit oder Hyperparameter-Tuning von Nöten sein, sodass auch hierzu genauere Erklärungen dargelegt und dabei insbesondere Klassifikationsprobleme fokussiert werden. Des Weiteren werden ausgewählte Methoden des maschinellen Lernens eingehend beschrieben, wobei der Großteil dieser bei der konkreten Durchführung einer Benchmark-Studie zu Daten akuter Cholangitis herangezogen wurden. Auf Basis der so erlangten Ergebnisse wurde ein Gewinnermodell deklariert und schließlich an die Daten angepasst, sodass die Effekte einzelner Einflussvariablen auf die In-Hospitale Mortalität offengelegt werden konnten. Im Verlauf dieser Auswertungen verdeutlichte sich die Komplexität derartiger Benchmark-Analysen und der Mangel an geeigneten Visualisierungsmöglichkeiten, sodass ergänzend eine Shiny App ausgearbeitet wurde. Neben alledem finden sich bei dieser auch Implementierungen von Methoden, die die Interpretierbarkeit von angewandten (Blackbox-)Methoden zugänglich machen sollen.

Inhaltsverzeichnis

Inhaltsverzeichnis **I**

Inhalt CD-ROM und Links zur App shinyBMR **IV**

Abbildungsverzeichnis **V**

Tabellenverzeichnis **VII**

Abkürzungsverzeichnis **VIII**

Notation **XI**

1 Einleitung **1**

2 Maschinelles Lernen (ML) **2**
2.1 Begriffsbestimmung und Abgrenzung zu verwandten Termini 2
2.2 Grundlagen: Vom menschlichen zum maschinellen Lernen 4
2.2.1 Der Begriff des „Lernens“ und zugehörige Komponenten 4
2.2.2 Maschinelles Lernen als generische Aufgabe 7

3 Benchmark-Studie **9**
3.1 Begriffsbestimmung 9
3.2 Aufteilung in Trainings-, Validierungs- und Testdatensatz 10
3.2.1 Notwendigkeit der Unterteilung und Vorgehen 10
3.2.2 Resampling-Methoden 12
3.2.3 Genestetes Resampling 17
3.3 Hyperparameter und deren Tuning 18
3.3.1 Abgrenzung der Hyperparameter von Modellparametern 18
3.3.2 Tuning der Hyperparameter 19
3.4 Gütemaße zur Beurteilung der Performance von ML-Methoden 23
3.4.1 Accuracy (ACC) und Missklassifikationsfehler (MMCE) 25
3.4.2 ROC-Analysen und Fläche unterhalb der ROC-Kurve (AUC) 26
3.4.3 Mittlerer quadratischer Fehler (MSE) bzw. Brier-Score 31
3.4.4 (Korrigiertes) Bestimmtheitsmaß R^2 31

4 Ausgewählte ML-Methoden **33**
4.1 k-Nearest Neighbors (kNN) 33
4.2 Support Vector Machines (SVM) 36
4.2.1 Darstellung der Grundlagen anhand linearer Separation 37

4.2.2 Ausweitung auf nicht-lineare Separation . . . 37
4.2.3 Kernelfunktionen . . . 40
4.3 Entscheidungsbäume basierend auf konditionaler Inferenz (cTree) . . . 41
4.3.1 Entscheidungsbäume im Allgemeinen . . . 41
4.3.2 cTree: Bäume basierend auf konditionaler Inferenz . . . 42
4.4 Ensemblemethoden . . . 47
4.4.1 Ensemblemethoden im Allgemeinen . . . 47
4.4.2 Random Forests (RFs) als Erweiterung des Baggings . . . 48
4.4.3 (Componentwise Gradient) Boosting . . . 52
4.5 Logistische Regression (mit Penalisierung) . . . 57
4.5.1 Das Logit-Modell . . . 57
4.5.2 Regularisierung mittels Shrinkage-Methoden . . . 59

5 Software-Nutzung und Entwicklung einer Shiny App 63
5.1 Machine Learning in R (*mlr*) . . . 63
5.1.1 Wrapper . . . 64
5.1.2 Imbalance-Korrektur . . . 64
5.1.3 Zusatzpaket *mlrHyperopt* . . . 67
5.2 Entwicklung der Shiny App *shinyBMR* . . . 68
5.2.1 Motivation . . . 68
5.2.2 Erarbeitung der App *shinyBMR* und Nutzungsmöglichkeiten . . . 69
5.2.3 Herausforderungen und Ausblick . . . 74
5.3 Interpretable Machine Learning (Zusatzpaket *iml*) . . . 76
5.3.1 Partial Dependence (PD) Plot . . . 78
5.3.2 Individual Conditional Expectation (ICE) Plot . . . 79
5.3.3 Accumulated Local Effects (ALE) Plot . . . 81
5.3.4 Feature Importance . . . 83
5.3.5 Feature Interaction . . . 85
5.3.6 Surrogate Interpretationsmethoden - global und lokal (LIME) . . . 87

6 Durchführung einer Benchmark-Studie basierend auf einem Datensatz zu akuter Cholangitis 90
6.1 Daten zu Todesfällen bei akuter Cholangitis . . . 90
6.1.1 Krankheitsbild der akuten Cholangitis . . . 90
6.1.2 Beschreibung der Studie und der Daten (-aufbereitung) . . . 91
6.1.3 Deskriptive und univariate Analysen . . . 93
6.2 Vergleich von ML-Methoden mittels einer Benchmark-Studie . . . 94
6.2.1 Vorbereitungen und Einstellungen in Anbetracht der Benchmark-Analyse 94
6.2.2 Auswertung der Ergebnisse aus der Benchmark-Studie . . . 97
6.2.3 Anpassung und Anwendung von *glmnet* zur Beurteilung der Effekte . . 100

7 Zusammenfassung und Ausblick 104

8 Anhang i
8.1 Ergänzungen zu Kap. 2 - Maschinelles Lernen . . . i
8.1.1 Abgrenzung des ML's von den Termini AI, DL und Statistik . . . i
8.1.2 Supervised (SL) gegenüber unsupervised Learning (USL) . . . ii

8.2 Ergänzungen zu Kap. 3 - Benchmark-Studie v
8.2.1 Ergänzung Trainings-, Validierungs- und Testdatensatz - Prototyping . v
8.2.2 Sammlung von Performance-Maßen vi
8.2.3 SMBO vii
8.2.4 ROC-Analyse viii
8.3 Ergänzungen zu Kap. 4 - Ausgewählte ML-Methoden xii
8.3.1 Bagging: Out-of-Bag (OOB) Fehlerschätzung xii
8.3.2 RF: Variable Importance xiii
8.3.3 Adaptives Boosting (AdaBoost) xiv
8.3.4 Logit-Modell (mit Penalisierung) xvi
8.4 Ergänzungen zu Kap. 5: Software-Nutzung und Entwicklung einer Shiny App . xxi
8.4.1 Beispiel-basierte Erklärungsmethoden xxi
8.5 Ergänzungen zu Kap. 6: Durchführung einer Benchmark-Studie basierend auf einem Datensatz zu akuter Cholangitis xxiii
8.5.1 Cholangitis - Aufbau und Funktion der Gallenwege xxiii
8.5.2 Imputation fehlender Werte xxiv
8.5.3 Deskriptive Analysen xxv
8.5.4 Beschreibung der Studie und der Daten - Univariate Analyse von Prädiktoren der Sterblichkeit xxv
8.5.5 Einstellungen bei Default-Setting, sowie Hyperparameter-Tuning und SMOTE xxvii
8.5.6 Auswertung der Ergebnisse aus der Benchmark-Studie xxix

Handbuch zu shinyBMR **xxxiv**

Literaturverzeichnis **XIV**

Eidesstattliche Erklärung

Inhalt CD-ROM und Links zur App shinyBMR

Aufbau beigefügte CD-ROM

Auf der beigefügten CD-ROM finden sich alle durchgeführten Analysen, sowie der aktuelle Code zur Shiny App *shinyBMR*. Es gilt zu beachten, dass aus datenschutzrechtlichen Gründen der zugrundeliegende Datensatz nicht zur Verfügung gestellt werden kann.

- Analyse: *Ordner mit jeglichen Dateien und R-Codes, die für die (Benchmark-) Analyse der Daten zu akuter Cholangitis genutzt und angefertigt wurden.*
  - _default_tuning: *Ordner mit .txt- und .R-Dateien, die Werte bezüglich Hyperparameter-Tunings umfassen und im Wesentlichen auf den Setting von mlrHyperopt (Stand: 22.05.2019) aufbauen.*
  - _saved_files: *Ordner mit .RDS-Dateien, die das BMR-Objekt, selbständig generiertes Gütemaß pAUC, sowie die angepassten Modelle cForest und glmnet beinhalten.*
  - 0_1_Datenaufbereitung.R
  - 0_2_Guetemass_pAUC.R
  - 1_Deskriptive_Analyse.R
  - 2_Benchmark_Analyse.R
  - 3_Finales_Modell_glmnet.R
  - 4_IML.R
- Graphiken: *Alle Graphiken, die aus der Analyse entnommen wurden und in der Arbeit angehängt sind. Die meisten dieser stellen .pdf-Dateien dar, die innerhalb von shinyBMR erzeugt wurden.*
- shinyBMR: *(R-) Code der für die Funktionsweise der Shiny App (Stand: 13.07.2019) erforderlich ist.*
- Handbuch_shinyBMR.pdf
- MA_Stüber.pdf

Links zur App shinyBMR

Online verfügbar (Stand: 13.07.2019) unter:

- Über Shinyapps.io: `https://compstat-lmu.shinyapps.io/shinyBMR/`
- Als GitHup Repository: `https://github.com/tess-st/shinyBMR`

Abbildungsverzeichnis

2.1 Hierarchischer Struktur von AI, ML und DL; Ergänzung um Einordnung der DS in Abb. 8.1 2

3.1 Bias-Varianz-Tradeoff: Mit zunehmender Modellkomplexität steigt die Wahrscheinlichkeit der Überanpassung (Overfitting) an die Trainingsdaten 13
3.2 Beispielhafte Veranschaulichung zum genesteten Resampling: 3-fache CV in äußerer Schleife und 4-fache CV in innerer Schleife (Hyperparameter-Tuning) . 18
3.3 Genereller Ansatz der SMBO-Prozedur 22
3.4 Beispielhafte Veranschaulichung zur ROC-Kurve: Entstehung der Stufenfunktion durch Verschiebung des Thresholds θ 28
3.5 Beispielhafte Veranschaulichung zum pAUC: Fokussierung von links: niedriger FPR ($c_1 = 0, c_2 = 0.2$); rechts: hoher TPR ($c_1 = 0.8, c_2 = 1$) . . 30

4.1 Beispielhafte Veranschaulichung zu kNN: Prädiktion in Abhängigkeit von der Anzahl in Betracht gezogener Nachbarn k 35
4.2 Beispielhafte Veranschaulichung zu SVMs links: Darstellung der Hyperebene für den Fall linearer Separierbarkeit; rechts: Darstellung der Hyperebene für den Fall nicht-linearer Separierbarkeit . 39
4.3 Übliche Architektur von Ensemblemethoden 48

5.1 Aufbau der App *shinyBMR*; gepunktete Pfeile als indirekte Schritte 68
5.2 Beispielhafte Veranschaulichung zu PD (gelb) und ICE (schwarz) Plots, cForest: Funktionaler Zusammenhang zwischen prognostizierter IHM (nein/ja) in partieller Abhängigkeit von der Anzahl an Leukozyten. 81
5.3 Beispielhafte Veranschaulichung zu ALE Plots, cForest: Visualisierung der ALEs von der Anzahl an Leukozyten auf die prognostizierter IHM (nein/ja). 83
5.4 Beispielhafte Veranschaulichung zu Feature Importance, cForest: Assoziation der Features gemessen anhand der Höhe des Modellfehlers bei der Prädiktion von IHM (nein/ja) durch Permutation des jeweiligen Features. 84
5.5 Beispielhafte Veranschaulichung zu Feature Interaktion, cForest: Interaktionsstärke (H-Statistik) für jedes Feature mit allen übrigen zur Prognose der IHM (nein/ja). 86
5.6 Beispielhafte Veranschaulichung zu Surrogatem Modell (global), cForest: Terminale Knoten eines surrogaten Entscheidungsbaumes, der die Prädiktionen des cForest's modelliert, mit Darstellung der Häufigkeiten von Klassifikationen der Blackbox-Methode innerhalb der einzelnen Knoten. 88

6.1 Performance (pAUC) der 22 Lerner in Anwendung auf Daten zu akuter Cholangitis: Gruppierung nach ML-Methodik und Bearbeitung mit Tuning/SMOTE. Oben: aggregierte BMRs; unten: unaggregierte BMRs. 98
6.2 Skyline-Level-Plot mit Berücksichtigung der Gütemaße AUC und pAUC: Pareto-Dominanz des glmnet's mit Tuning gegenüber den übrigen Lernern in Anwendung auf Daten zu akuter Cholangitis. 99

8.1 Hierarchischer Struktur von AI, ML, DL mit zusätzlicher Einbeziehung von DS ii
8.2 Zusammenfassung (herkömmlicher) ML Methoden iii
8.3 Veranschaulichung der Prototyping-Phase: Verwendung der Validierungs- und Trainingsdaten v
8.4 Beispielhafte Veranschaulichung zur ROC-Kurve: Klassenzuweisung in Abhängigkeit von festgesetztem Schwellenwert θ viii
8.5 Beispielhafte Veranschaulichung zur ROC-Kurve: Iso-Accuracy Linien (IAL's) und Bestimmung des zugehörigen ACC's x
8.6 Beispielhafte Veranschaulichung zur ROC-Kurve: Konvexe Hülle (ROCCH) . . xi
8.7 Beispielhafte Veranschaulichung zu AdaBoost
links: 3 Iterationen des Algorithmus mit jeweiliger Klassifikation über einen schwachen Basis-Lerner und vorzunehmender Gewichtung der Beobachtungen; rechts: Kombination der drei Basis-Lerner zu einem starken Klassifikator xv
8.8 Aufbau der Gallenwege xxiii
8.10 Tabelle und Zusammenfassung der (aggregierten) Ergebnisse aus Benchmark-Studie zu akuter Cholangitis. xxix
8.9 Histogramme zu numerischen Prädiktoren; Blaue Linie zur Kenntlichmachung desjenigen Punktes, der die zusätzlichen Dichotomisierung der Variablen ausmacht xxx
8.11 Performance (AUC) der 22 ausgewählten Lerner in Anwendung auf Daten zu akuter Cholangitis: gruppiert nach ML-Methodik und vorgenommenem Tuning/SMOTE.
Oben: aggregierte BMRs; unten: unaggregierte BMRs. xxxi
8.12 Heatmap zur Performance der 22 ausgewählten Lerner in Anwendung auf Daten zu akuter Cholangitis: gruppiert nach ML-Methodik und vorgenommenem Tuning/SMOTE.
Oben: Beurteilung über pAUC; unten: Beurteilung über AUC. xxxii
8.13 Tabelle zu Skyline-Levels mit Berücksichtigung der Gütemaße AUC und pAUC: Pareto-Dominanz des glmnet's mit Tuning gegenüber den übrigen Lernern in Anwendung auf Daten zu akuter Cholangitis. xxxiii

Tabellenverzeichnis

2.1 Statistik vs. ML: Unterschiedliche Begrifflichkeiten [vgl. Wasserman (2012) und Tibshirani (2012)] 4

3.1 Konfusionsmatrix 27

4.1 Bekannte Kernelfunktionen zur potentiellen Anwendung innerhalb von SVMs, wobei d den Grad des Polynoms und σ die Standardabweichung bezeichnen . . 40
4.2 Annahmen im GLM mit Logit-Link 58

6.1 Als Prädiktoren für die Anpassung des Modells zur Prognose der Mortalität bei akuter Cholangitis herangezogenen (dichotome und numerische) Variablen . 92
6.2 Durchführung der Benchmark-Studie mit 22 verschiedenen Lernern, basierend auf sechs ML-Methoden mit jeweils unterschiedlichen Einstellungen (s. Tab. 8.4) 96
6.3 Kovariableneffekte resultierend aus der Anpassung eines glmnet's mit $\alpha = 0.25$ und $\lambda = 0.0093$ an die Daten zu akuter Cholangitis zur Modellierung der IHM 101

8.1 Performance-Maße für Klassifikationsaufgaben, ableitbar aus Konfusionsmatrix in Tab. 3.1 vi
8.2 Performance-Maße für Regressionsaufgaben vii
8.3 Studie zu akuter Cholangitis: Univariate Analyse der Prädiktoren für Sterblichkeit, übernommen von Schneider et al. (2016) xxvii
8.4 Default-Settings und Einstellungen des Hyperparameter-Tunings: SMOTE-, sowie Klassifikationsmethoden mit Parameter-Konfiguration. Ohne Suchraum $\hat{=}$ kein Tuning. xxviii

Abkürzungsverzeichnis

ACC	Accuracy	Genauigkeit
AI	Artificial Intelligence	Künstliche Intelligenz
AIC	Akaike Information Criterion	Akaikes Informationskriterium
ALE(P)	Accumulated Local Effect (Plot)	-
AUC	Area Under the (ROC-) Curve	Fläche unterhalb der ROC-Kurve
BIC	Bayesian Information Criterion	Bayessches Informationskriterium
BMR	Benchmark Result	Ergebnis der Benchmark-Studie
BS	Bootstrap	-
CART	Classification and Regression Trees	Klassifikations- und Regressionsbäume
CGB	Component-wise GB	Komponenten-weises GB
c-ICE	centered ICE	-
cTree	Conditional Inference Trees	Entscheidungsbäume basierend auf konditionaler Inferenz
cForest (oder CIF)	Conditional Inference Forest	RF basierend auf cTrees Basis-Lerner
CV	Cross-Validation	Kreuzvalidierung
DGP	Data Generating Process	Daten-generierender Prozess
DL	Deep Learning	Tiefgehendes Lernen (Optimierungsverfahren, Teilgebiet von ML)
DS	Data Science	Daten Wissenschaften
EDA	Estimation of Distribution Algorithm	Algorithmus zur Schätzung der Verteilung
EI	Expected Improvement	-
ETC	Endoscopic Retrogarde Cholangiography	Endoskopisch Retrograde Cholangiopankreatikographie
FP(R)	False-Positive(-Rate)	Falsch-Positiv(-Rate)
FS	Feature Space	Feature-Raum
GAM	Generalized Additive Model	Generalisiertes Additives Modell
GB(M)	Gradient Boosting (Machines)	-
GE	Generalization Error	Generalisierungsfehler (bedingtes Risiko)
GLM	Generalized Linear Model	Generalisiertes Lineares Modell
IAL	Iso-Accuracy Line	Iso-Accuracy Linie (gleicher Genauigkeit bzw. ACC'S)

ICE(P)	Individual Conditional Expectation (Plot)	-
IHM	In-Hospital Mortality	In-Hospital Mortalität
IML	Interpretable ML	Methoden zur Interpretierbarkeit von ML
KI	Confidence Interval	Konfidenzintervall
kNN	k-Nearest-Neighbors (-Algorithm)	k-Nächste-Nachbarn (-Algorithmus)
LASSO	Least Absolute Shrinkage and Selection Operator	-
LDA	Linear Discriminant Analysis	Lineare Diskriminanzanalyse
LIME	Local Interpretable Model-agnostic Explanations	Lokal Interpretierbare Modell-agnostische Erklärungen
LM	Linear Model	Lineares Modell
Log Odds	Logarithm of the Odds	Logarithmiertes Chancenverhältnis/Risiko (Log Odds)
LOOCV	Leave-One-Out CV	-
MBO	Model-based Optimization	Modell-basierte Optimierung
MCCV	Monte-Carlo CV	-
ML	Machine Learning	Maschinelles Lernen
MLE	Maximum-Likelihood Estimator	Maximum-Likelihood Schätzer
MMCE	Mean Missclassification Error	Mittlerer Missklassifikationsfehler
MMH	Maximum Margin Hyperplane	Maximum Margin Hyperebene
MPI	Maximum Probability of Improvement	-
OLS	Ordinary Least Squares	Gewöhnliche Methode der kleinsten Quadrate
OOB	Out-of-Bag	-
OR	Odds Ratio	Chancenveräl̈tnis
pAUC	partial AUC	partielles AUC
PCA	Principal Component Analysis	Hauptkomponentenanalyse
PCP	Parallel Coordinates Plot	Parallel Koordinaten (Grafik)
PD(P)	Partial Dependence (Plot)	-
PTC	Percutaneous Transhepatic Cholangiography	Perkutane Transhepatische Cholangiographie
QDA	Quadratic Discriminat Analysis	Quadratische Diskriminanzanalyse
RF	Random Forests	Zufallswälder
RHKS	Reproducing Kernel Hilbert Space	-
ROC	Receiver Operating Characteristic	Grenzwertoptimierung

ROCCH	ROC Convex Hull	ROC konvexe Hülle
SL	Supervised Learning	Überwachtes Lernen
SMBO	Sequential MBO	Squentielle MBO
SMOTE	Synthetic Minority Oversampling Technique	-
SQR	Sum of Squared Residuals	Summe der quadrierten Residuen
SQT	Sum of Squared Total deviations	Summe der quadrierten totalen Abweichungen
SVM	Support Vector Machine	Stützvektormaschine/-methode
TP(R)	True-Positive(-Rate)	Richtig-Positiv(-Rate)
USL	Unsupervised Learning	Unüberwachtes Lernen
VI	Variable Importance	Variablen-Wichtigkeit
XGBoost	E**X**treme GB	-

Notation

Allgemein sei vermerkt, dass lateinische bzw. griechische Kleinbuchstaben Skalare kennzeichnen oder Vektoren, insofern diese fett gedruckt sind. Des Weiteren werden fett gedruckte Großbuchstaben für die Kenntlichmachung von Matrizen verwendet. Ebenso sind Zufallsgrößen durch Großbuchstaben dargestellt, sobald jedoch eine Realisierung dieser angesprochen wird, ist auf Kleinbuchstaben zurückgegriffen.
Jegliche Schätzer können anhand des Zirkumflex ($\wedge$) erkannt werden und arithmetische Mittel anhand von einem waagrechten Strich ($-$) oberhalb des jeweiligen Vektors. Zusätzlich sind Approximationen durch ein übergestelltes Tilde-Zeichen ($\sim$) markiert.

$$\boldsymbol{X} = \begin{array}{c} \\ X_{1\cdot} \\ X_{2\cdot} \\ \dots \\ X_{(N-1)\cdot} \\ X_{N\cdot} \end{array} \begin{array}{c} \begin{array}{cccc} X_{\cdot 1} & X_{\cdot 2} & \dots & X_{\cdot P} \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1P} \\ x_{21} & x_{22} & \dots & x_{2P} \\ \dots & \dots & \ddots & \dots \\ x_{(N-1)1} & x_{(N-1)2} & \dots & x_{(N-1)P} \\ x_{N1} & x_{N2} & \dots & x_{NP} \end{pmatrix} \end{array}$$

Indizes	**Bedeutung**
$b = 1, \dots, B$	Bootstrap-Stichprobe
$i = 1, \dots, N$	Subjekt/Person
$i = 1, \dots, n$	Subgruppe/-set der Beobachtungen
$k = 1, \dots, K$; $k = g$ bei kNN	Kategorien/Klassen
$m = 1, \dots, M$	Boosting-Iteration
$p = 1, \dots, P$	Variable/Merkmal

Matrizen, Vektoren, Räume und Verteilungen	**Bedeutung**
$\mathcal{X}$	P-dimensionaler Input-Raum; zumeist $\mathbb{R}^P$, aber auch kategoriale Features möglich
$\mathcal{Y}$	Output-/Ziel-Raum; z.B. $\mathcal{Y} = \mathbb{R}$, $\mathcal{Y} = \{0; 1\}$, $\mathcal{Y} = \{1, \dots, K\}$ oder auch $\mathcal{Y} = \{Label_1, \dots, Label_K\}$
$\boldsymbol{X} \in \mathcal{X}$	$N \times P$-Datenmatrix *(Aufbau s. nachstehende Matrix)*

$\boldsymbol{x}_p^T = (x_{1p}, \ldots, x_{Np})^T$	Vektor mit Beobachtungen zu p-ter unabhängigen/erklärenden/Input-Variable, Einflussgröße, Attribut, Feature, Prädiktor
$\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iP})$	Vektor mit Beobachtungen zu dem Subjekt i
$Y = \boldsymbol{y} = (y_1, \ldots, y_i, \ldots, y_N)$	Vektor mit Beobachtungen der N Zielgrößen, Abhängige/Output-Variablen; Responsevektor
$\{\boldsymbol{x}_\bullet, y_\bullet\}$	Neue Beobachtung; $\boldsymbol{x}_\bullet$ als neue Instanz/Input, $y_\bullet$ zugehöriger (unbekannter) Output
$\mathcal{H}$	Hypothesenraum
$\mathcal{L}$, $\mathcal{T}$	Lern-, Trainingsdatensatz
$\mathcal{P}$	Verteilung
$\mathcal{P}_N$	Empirische Verteilung
$\mathbb{P}(\boldsymbol{x}, y) = \mathbb{P}_{\boldsymbol{x},y}$	Gemeinsame Wahrscheinlichkeitsverteilung von x und y auf $\mathcal{X} \times \mathcal{Y}$

Operatoren, Funktionen und Symbole	**Bedeutung**
$\hat{x}$	Schätzwert von x
$\tilde{x}$	Approximation zu x
$\overline{\boldsymbol{x}}$	Mittelwert des Vektors $\boldsymbol{x}$
$\boldsymbol{x}^T$	Transposition des Vektors $\boldsymbol{x}$
$\langle \cdot, \cdot \rangle$	Skalarprodukt
$\|\| \cdot \|\|$	(Euklidische) Norm
$\phi(\cdot)$	Abbildungsfunktion zu höher-dimensionalem Raum
$\psi(\cdot)$	Penality, Bestrafungsfunktion
$\mathbb{E}(\cdot) = \mu$	Erwartungswert
$\mathbb{I}(\cdot)$	Indikatorfunktion
$I(\cdot)$	Unreinheitsfunktion (Impurity)
$J(\cdot)$	Bestrafungsfunktion
$K(\cdot, \cdot)$	Kernelfunktion, Kerne
$L(\cdot, \cdot)$	Verlustfunktion
$\mathcal{L}(\cdot)$	Likelihood-Funktion
$N_k(\cdot)$	k-Nachbarschaft
$R(\cdot, \cdot)$	Risikofunktion

$\mathbb{T}(\cdot,\cdot)$	(Multivariate) lineare Teststatistik
$\mathbb{V}(\cdot) = \Sigma$	Varianz
$d(\cdot,\cdot)$	Distanz
$g(\cdot)$	cTree: Transformation Logit: Link-Funktion
$h(\cdot)$	cTree: Einflussfunktion Logit: Response-Funktion
$\ell(\cdot)$	logLikelihood-Funktion
$\tilde{\theta}_{Risk}$	Risiko, erwarteter Verlust
$\tilde{\theta}_{GE}$	Bedingtes Risiko, Generalisierungsfehler
η	Linearer Prädiktor
ρ	Korrelation
σ	Standardabweichung
ξ	Slack Variable
C	Fehlergewicht
H	Hypothese
T	Spezifischer (Entscheidungs-) Baum
b	Bias, Offset
m	Anzahl in Betracht gezogener Split-Kandidaten bei Knoten innerhalb des RF's
$u, \boldsymbol{u}$	Negativer Gradient(en-Vektor)
$\mathbf{t}$	(multivariate) lineare Statistik
w,$\boldsymbol{w}$	(Vektor der) Gewicht(e); $\boldsymbol{w}$ bei SVMs Richtungs-/Normalenvektor

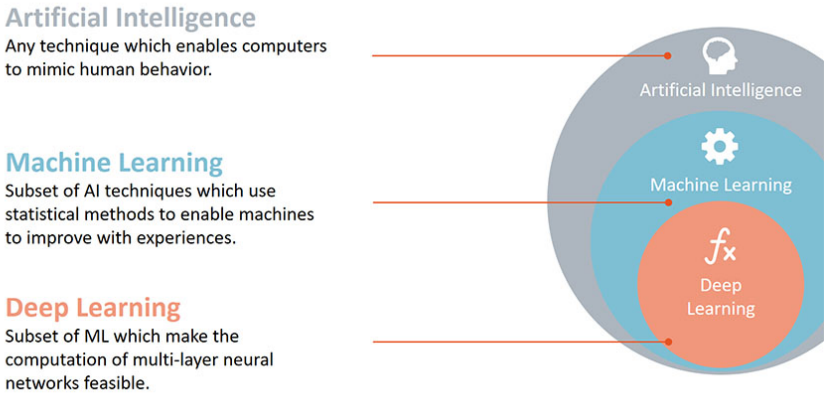
1 Einleitung

„Jupiter, Juno und Minerva waren im antiken Rom das tonangebende Dreiergespann und waren nebeneinander sowie miteinander von höchster Wichtigkeit. Ein ähnliches Bild lässt sich zurzeit im modernen Datenzeitalter zeichnen, jedoch nicht mehr von Götterbildern, sondern von den neu entstandenen tragenden Säulen der Zukunftstechnologie: vom Menschen getragene Datenanalyse, Künstliche Intelligenz und Machine Learning.“ (Becker, 2019)

Derartige gottähnliche Vergleiche mögen wohl etwas überspitzt erscheinen, wenn man nach Beschreibungen für Begrifflichkeiten wie künstliche Intelligenz und Machine Learning (ML) sucht, doch soll hierdurch vielmehr verdeutlicht werden, wie sehr man bei jeglicher Form der Datenanalyse im Zeitalter der Digitalisierung und Flut der Daten auf diese Hilfsmittel angewiesen ist. Insbesondere gilt es hier nunmehr die Vorzüge des ML's im Anwendungsfeld der Statistik für sich zu nutzen. So stößt man jedoch unweigerlich auf eine Vielzahl möglicher Methodiken bzw. Lerner, die sich bei der Analyse - sei es explorativen oder prädiktiven Charakters - von Daten anwenden lassen. Verfolgt man nun zunächst das Ziel die Güte potentiell nutzbarer Methodiken zu messen, einander gegenübzustellen und dadurch gewisse Vorrangstellungen aus unterschiedlichen Blickwinkeln auszumachen, findet man sich im Bereich s.g. Benchmark-Studien wider. Somit stellen diese ein starkes Werkzeug mitunter im Bereich des ML's dar und lassen sich - rückblickend auf den bildlichen Vergleich mit Göttern - wohl als mythologischen Gegenstand und damit sinnbildlich als einen Teil deren Stärken ausmachen.
In diesem Sinne fokussierte diese Arbeit die Methodik, Anwendung und Interpretation moderner Benchmark-Studien am Beispiel der Risikomodellierung bei akuter Cholangitis. Um dem gerecht zu werden, wird in Kap. 2 zunächst auf das maschinelle Lernen selbst eingegangen. Anschließend wird in Kap. 3 ein in diesem Kontext wichtiges Instrumentarium, die s.g. Benchmark-Studie fokussiert, indem hiermit einhergehende Bearbeitungen der Daten, Funktionalitäten und insbesondere Gütemaße Beschreibung finden. Innerhalb derartiger Analysen werden also verschiedene Methoden des maschinellen Lernens einander gegenübergestellt und bewertet. Daher werden in dem darauffolgenden Kap. 4 ausgewählte Methodiken einschließlich deren Funktionsweise offengelegt, die Größtenteils ihre Anwendung in der Benchmark-Studie zu Daten akuter Cholangitis finden. Daran anschließend werden in Kap. 5 die zur Analyse im Software-Kontext herangezogenen Werkzeuge kurz vorgestellt, von welchen insbesondere auch bei der im Rahmen dieser Arbeit entwickelten Shiny App Gebrauch gemacht wurden, und insbesondere auch das s.g. Interpretable Machine Learning fokussiert. Schließlich wird in Kap. 6 die konkrete Durchführung einer Benchmark-Studie auf Basis der Daten zu akuter Cholangitis einschließlich der aus der Analyse resultierenden Ergebnisse präsentiert. Abrundend soll in Kap. 7 eine Zusammenfassung der behandelten Thematiken mitsamt der durch die konkrete Analyse erlangten Erkenntnisse, sowie ein Ausblick bereitgestellt werden.

2 Maschinelles Lernen (ML)

Maschinelles Lernen (ML) findet seine Wurzeln gemäß Machart (2012, Kap. 1) in einem breiten Spektrum wissenschaftlicher (Teil-) Gebiete, wie etwa der künstlichen Intelligenz, Optimierung, Mustererkennung oder Statistik. Zunächst lag das wesentliche Augenmerk auf computationalen Herausforderungen, wie etwa der Berechenbarkeit einer vorliegenden Problemstellung und der effizienten Implementierung zugehöriger Algorithmen. Später rückte die statistische Sichtweise in den Vordergrund und damit insbesondere die Frage nach Möglichkeiten der verallgemeinerten Anwendung, sowie dem Verständnis/Interpretierbarkeit und der Beurteilung der Performance bereitgestellter Lösungen bei Anwendung auf neue Daten.
Um die Theorie des maschinellen Lernens bestmöglich vermitteln zu können, soll zunächst in Sektion 2.1 eine genauere Begriffsbestimmung erfolgen bzw. Unterschiede und Gemeinsamkeiten mit anderen verwandten Wissenschaftsfeldern aufgezeigt werden und dadurch das Gebiet des ML's selbst umrissen werden. Anschließend werden mittels der Untersektion 2.2 - anhand des Übergangs vom menschlichen zum maschinellen Lernen - die Grundlagen des ML's verdeutlicht und erste mathematische Formulierungen aufgestellt.



Quelle: Mierswa (2017a)

Abb. 2.1: Hierarchischer Struktur von AI, ML und DL; Ergänzung um Einordnung der DS in Abb. 8.1

2.1 Begriffsbestimmung und Abgrenzung zu verwandten Termini

Bereits zu Beginn dieser Arbeit wurden Begriffe genannt, die genauerer Definition bedürfen. Insbesondere sollte hier die Bedeutung von Data Science (DS), künstlicher Intelligenz (Artificial Intelligence = AI), maschinellem Lernen (Machine Learning = ML), Deep Learning (DL) und Statistik erläutert werden, was vorwiegend in Anlehnung an Mierswa (2017a) geschieht. So ist es jedoch teilweise schwierig eine klare Abgrenzung festzulegen, da die Übergänge flie-

ßend sind. Ebenso sind eindeutige Definitionen der Begrifflichkeiten damit fast unmöglich, genauso wie eine klare hierarchische Struktur Auslegungssache ist. In Rahmen dieser Sektion soll nun vorrangig die Begrifflichkeit des ML's verdeutlicht werden und dadurch auf indirektem Wege die Grenze bzw. der Übergang zu anderen Termini aufgezeigt werden. Sollte es dennoch genauerer Definitionen von AI, DL oder DS bedürfen, sei Untersektion 8.1.1 herangezogen.
Fokussiert man nun Abb. 2.1, so lässt sich maschinelles Lernen als Teilbereich der AI ausmachen. Im Wesentlichen ist dies darauf zurückzuführen, dass ML gekennzeichnet ist durch die Fähigkeit Prognosen - auf Datenbasis - zu generieren, ohne explizit hierfür programmiert zu werden. Wie hier also anklingt sind dafür Daten essentiell. Im Wesentlichen stützt sich maschinelles Lernen also auf das Erkennen von Mustern innerhalb der gegebenen Daten, anhand derer das System lernen kann und so ein „optimales“ Vorgehen für ähnliche Problemstellungen bzw. neue Daten hervorbringt. Derartige Lösungen werden mithilfe von Algorithmen angefertigt, die teilweise schon Jahrzehnte lang existieren, aber dank starker Entwicklungen im Bereich der Computerwissenschaften nun mit großen Datenmengen umgehen können und durch Parallelisierung deutlich beschleunigt werden können.
Gleichzeitig kommt wohl bei der eingehenderen Beschreibung von ML auch die Schnittstelle mit dem Bereich der Statistik zum Vorschein. Dieser zielt wohl v.a. auf die Frage ab, was man von gegebenen Daten lernen kann. Wasserman (2012) ging sogar so weit, auf die Frage, was der Unterschied zwischen den beiden Feldern - Statistik und ML - sei, zu antworten:

„The short answer is: None. They are both concerned with the same question: how do we learn from data?“ (Wasserman, 2012)

Diese Ansicht mag etwas grob erscheinen, soll aber verdeutlichen, wie eng verbunden die beiden Fachbereiche doch sind. Gleichzeitig hebt Wasserman (2012) nämlich auch hervor, dass der wohl größte Unterschied darin besteht, dass Statistik auf formale Inferenz (Konfidenzintervalle, Hypothesentests, Bestimmung optimaler Schätzer, etc.) bei niedrig dimensionierten Problemen fokussiert ist und ML dagegen auf hoch-dimensionale Prädiktionsprobleme abzielt. Es kommt hinzu, dass die Statistik ihren Ursprung in der Mathematik findet, wohingegen sich ML mehr dem Fachbereich der Informatik zuteilen lässt. Historischen gesehen kann man die Statistik also bereits zu einer Zeit ansiedeln, bevor es überhaupt Computer gab. Dagegen ist ML ohne die Nutzung derartiger Maschinen unmöglich. So allerdings entwickelten sich ähnliche bzw. gleichartige Methoden innerhalb der beiden Bereich, wobei sich v.a. hinsichtlich der Terminologie Unterschiede auffinden lassen die anhand von Tab. 2.1 aufgezeigt werden.
Allerdings sei hier gleichzeitig auch angemerkt, dass nicht alle ML-Methodiken auf probabilistischen Modellen basieren, sodass ML und Statistik eine gemeinsame Schnittmenge besitzen, keinesfalls jedoch ML nur eine Teilmenge der Statistik darstellt. Vielmehr sind gemäß Shah (2016) ML und Statistik zwei unterschiedliche Zweige, die sich im Laufe der Zeit immer mehr angenähert haben, voneinander gelernt haben und wohl auch in Zukunft eine immer größer

werdende Schnittmenge inne haben werden. Doch die Fähigkeit die Gemeinsamkeiten, sowie Unterschiede innerhalb der beiden Wissenschaftsfelder zu kennen, erlaubt es das jeweilige Wissen auszubreiten und Methoden außerhalb der eigenen Expertise anzuwenden. Damit ist die Beschreibung der Data Science selbst getan, welche die beiden Felder der ML und Statistik verbindet, sowie Lücken schließt.

Statistik		**ML**	
Englisch	*Deutsch*	*Englisch*	*Deutsch*
Estimator	Schätzer	Learner	Lerner
Fitting	Anpassung	Learning	Lernen
Model	Modell	Network, Graphs	Netzwerk, Graph
Paramters	Parameter	Weights	Gewichte
Observation/Data Point	Datenpunkt	Example/Instance	Fall/Beispiel
Classifier	Klassifizierer	Hypothesis	Hypothese
Hypothesis	Hypothese	-	-
Regression, Classification	Regression, Klassifizierung	Supervised Learning	Überwachtes Lernen
Density Estimation, Clustering	Dichteschätzung, Clusterbildung	Unsupervised Learning	Unüberwachtes Lernen
Covariate	Ko-/Einflussvariable	Feature, Input	Merkmal, Eingabewert, Eigenschaft
Response	Zielvariable	Label, Output	Attribut, Rückgabewert

Tab. 2.1: Statistik vs. ML: Unterschiedliche Begrifflichkeiten [vgl. Wasserman (2012) und Tibshirani (2012)]

2.2 Grundlagen: Vom menschlichen zum maschinellen Lernen

Durch die vorhergehenden Beschreibungen sollten nun wesentliche Aspekte des maschinellen Lernens (ML) verdeutlicht sein. Nachdem dieses Wissenschaftsgebiet jedoch sehr komplex ist, bedarf es eingehender Erklärungen. Die hierzu folgenden Beschreibungen stützen sich zunächst auf Hastie et al. (2008) und sollen den Einstieg in die Thematik erleichtern.

2.2.1 Der Begriff des „Lernens“ und zugehörige Komponenten

Um entsprechende Beschreibungen bestmöglich bewerkstelligen zu können, sollte man vorab der Begriff des Lernens verdeutlichen. Auf der Suche nach Definitionen dieses Begriffes, erhält man eine Vielzahl an Vorschlägen, wobei nachfolgend drei derer beispielhaft gelistet sind.

„Lernen ist das Aufnehmen, Verarbeiten und Umsetzen von Informationen“
Schilling (1997, S. 159)

„Lernen ist ein Prozess, der zu relativ stabilen Veränderungen im Verhalten oder Verhaltenspotential führt und auf Erfahrung aufbaut. Lernen ist nicht direkt zu beobachten, sondern muß aus den Veränderungen des beobachtbaren Verhaltens erschlossen werden.“
Zimbardo und Gerrig (1999, S. 206)

„Der Vorgang des Lernens ist vielmehr grundlegend für jede Art von aktueller Anpassung, die sich als Resultat von Erfahrungen bildet.“
Krech und Crutchfield (1992, S. 7)

Auch wenn die vorangehenden Definitionen sich im Wesentlichen auf das Lernen im Sinne der menschlichen Natur stützen, so kann man den Begriff doch auch computerbasierte Ansätze ausweiten und damit das maschinelle Lernen zum Ausdruck bringen. Insbesondere sollte deutlich werden, dass Lernen ein Prozess ist, bei dem durch Aufnahme von Informationen - in welcher Form auch immer - und deren Verarbeitung eine Anpassung oder Veränderung des Verhaltens stattfindet. Dies kann im weitesten Sinne auf das computergestützte Lernen übertragen werden: eine „Maschine“ - im engeren Sinne - wird auf Basis vorhandener Daten oder vergangener Erfahrung in die Lage versetzt, Muster innerhalb dieser zu erkennen, zu verarbeiten und Lösungen zu entwickeln.
Innerhalb dieser Sektion werden Beschreibungen nun basierend auf dem s.g. überwachten (supervised) Lernen erfolgen. Zur Abgrenzung des überwachtem vom unüberwachten (unsupervised) Lernen kann Kapitel 8.1.2 herangezogen werden. Im weiteren Verlauf dieser Arbeit wird auf die englischen Begrifflichkeiten „Supervised“ und „Unsupervised Learning“ zurückgegriffen und damit auf deren Abkürzungen SL und USL.
Es stellt sich nun die Frage, wie ein Computer dieses „Lernen“ bewältigen soll bzw. wie sichergestellt wird, dass ein angewandter Algorithmus tatsächlich lernt? Um diesen Aspekt auszuweiten, denke man unter Einbeziehung von de Mello und Ponti (2018, Kap. 1), sowie Morik (2013, Kap. 1) zunächst erneut an die Lern-Gewohnheiten eines Menschen zurück. Wie erwirbt beispielsweise ein Kind das Wissen, dass es sich bei dem betrachteten Gegenstand um ein Buch handelt? Erstmals werden wohl die Eltern dieses Gebilde als Buch deklarieren. Doch dann ist es auch notwendig, dass das Kind relevante Eigenschaften - s.g. Features - erkennt, um derartige Definitionen selbstständig ausmachen zu können. Was sind also die notwendigen Voraussetzungen, die ein Gegenstand erfüllen muss, um als Buch erkannt zu werden? Die Farbe scheint wohl zumeist kein Ausschlag gebendes Kriterium zu sein. Hingegen spielt die Form bzw. die Abmessung hier schon eher eine Rolle, um das Objekt der Klasse oder dem Label „Buch“ zuordnen zu können. Entsprechend orientiert sich das menschliche Lernen an bestimmten Eigenschaften, ebenso das maschinelle Lernen, wobei man hier für Letzteres eher

Begriffe wie Attribute bzw. Features oder Variablen verwendet und in diesem Sinne sei noch einmal auf Tab. 2.1 verwiesen.
Psychologisch gesehen ist der Mensch nun dazu geneigt Beobachtungen, die sich ähnlich sehen zu Gruppen zusammenzufassen bzw. zu kategorisieren, was als Clustering bezeichnet wird. Auf diese Weise kann anhand von Ähnlichkeitsmerkmalen eine große Anzahl an Objekten strukturiert werden bzw. einer bestimmten Gruppe zugeordnet werden, wobei sich die Gruppen teilweise auch überlappen können. Naives Clustering bestünde also in der Unterteilung von Objekten in die Kategorien „Buch“ (positiv) und „Nicht-Buch“ (negativ).
Vom Clustering muss die s.g. Klassifikation abgegrenzt werden, welche letztliche eine Funktion darstellt, mithilfe derer eine Wahrnehmung einer Klasse zugeordnet wird. So stellt das Wort x - hier „Buch“ - eine Erkennungsfunktion $f(\cdot)$ dar, wobei jeder Wahrnehmung dann die Klasse y mit den Ausprägungen „Buch“ oder „Nicht-Buch“ zugeordnet werden kann.

$$f(x) = y,$$

$$\begin{aligned} \textit{mit } f(\cdot) &= \textit{(Erkennungs-) Funktion} \\ x &= \textit{Wahrnehmung/Objekt/Eigenschaft} \\ y &= \textit{Klasse/Label} \end{aligned}$$

Bei dem vorliegenden Beispiel bietet es sich also an $f(\cdot)$ als eine logische Funktion in Form einer Definition zu spezifizieren. Eine Definition stellt hierbei eine Erkennungs- und Ergänzungsfunktion als hinreichende und notwendige Bedingungen dar. In diesem Sinne definiere man ein Buch also wie folgt: rechteckiger Gegenstand aus gebundenen, gehefteten, oder anderweitig verbundenen Seiten bestehend, mit zumeist kartoniertem Einband für verschiedene Verwendung und mit unterschiedlich ausfallender Größe. Hierbei ergibt sich die Erkennungsfunktion also durch die Eigenschaften wie *rechteckig*, *gebundeneSeiten*, *bedruckteSeiten* und *kartonierterEinband*.

$$\textit{rechteckig}(x),\ \textit{gebundeneSeiten}(x),\ \textit{bedruckteSeiten}(x),\ \textit{kartonierterEinband}(x) \rightarrow \textit{Buch}(x)$$

Die Ergänzungsfunktion hingegen lässt sich beispielsweise durch die Frage klären, ob man in dem Buch blättern und/oder lesen kann. Ein Buch besitzt eine Vielzahl an bedruckten Seiten, womit entsprechend in dem Objekt geblättert und/oder gelesen werden kann.

$$\textit{Buch}(x) \rightarrow \textit{kann\_gelesen\_werden}(x)$$

Auffallend ist hier nun, dass bestimmte Begriffe genutzt werden, um einen anderen zu definieren. So kann es von Nöten sein z.B. die hier verwendeten Wörter „Seite“ „Text“ und „Einband“ vorab zu definieren, um eine derartige Beschreibung des Buches überhaupt vornehmen zu können.

2.2.2 Maschinelles Lernen als generische Aufgabe

Mit dieser Hinführung über das menschliche Lernen, kann nun leichter das maschinelle Lernen als generische Aufgabe eingeordnet werden, wobei dabei erneut auf Morik (2013, Kap. 1) Bezug genommen wird. Dazu gilt es nun erste mathematische Festsetzungen zu treffen. Im Folgenden gehe man von $i = 1, \ldots, N$ Subjekten/Personen aus, die jeweils anhand der $p = 1, \ldots, P$ Merkmalsausprägungen beschrieben werden. Entsprechend bildet die Gesamtheit aller Beobachtungen zu allen berücksichtigten Personen eine $N \times P$ -Matrix $\boldsymbol{X}$, deren Aufbau sich anhand der verallgemeinerten Matrix bzw. Datensatzes im Kapitel der Notation findet. Fortfolgend gehe man davon aus, dass die Zeilen die einzelnen Beobachtungen aufgeteilt nach den einzelnen Subjekten umfasst und entsprechend der Zeilenvektor $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iP})$ die Beobachtungen der Person i beinhaltet. Analog wird durch den Spaltenvektor $\boldsymbol{x}_p^T = (x_{1p}, \ldots, x_{Np})^T$ die gegebenen Merkmalsausprägungen der p-ten Einflussvariable zum Ausdruck gebracht. Des Weiteren werden die einzelnen Zielgrößen in Form des Vektors $\boldsymbol{y}^T = (y_1, \ldots, y_N)^T$ dargeboten. Gleichzeitig sei daran erinnert, dass Zufallsvariablen anhand von Großbuchstaben und Realisationen durch Kleinbuchstaben kenntlich gemacht sind.

Lernaufgabe Clustering

Mit den vorhergehenden Definitionen lassen sich nun die einzelnen Lernaufgaben aus maschineller Sicht beschreiben. So hat man beim Clustering eine Menge $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \boldsymbol{X}$ von Beobachtungen, die Anzahl k der zu bestimmenden Gruppen $C_1, \ldots, C_k$, sowie eine Abstandsfunktion $d(x, x')$ und eine Qualitätsfunktion gegeben. Die Aufgabe besteht nun darin die Gruppen $C_1, \ldots, C_k$ zu finden, sodass zum einen alle $\boldsymbol{x} \in \boldsymbol{X}$ einer Gruppe zugeordnet werden und zum anderen die festgesetzte Qualitätsfunktion optimiert wird. Letzteres impliziert, dass der Abstand der Beobachtungen innerhalb einer Gruppe minimiert werden und der Abstand zwischen den Gruppen möglichst groß sein soll.

Lernaufgabe Klassifikation

Ebenso kann nun auch die Aufgabe der Klassifikation im Hinblick auf ML beschrieben werden. Gegeben seien hierzu die Klassen in Y, wobei zurückgreifend auf das Buch-Beispiel mit der Unterteilung in „Nicht-Buch“ und „Buch“ für eine einzelne Beobachtung gelten würde $y \in \{0, 1\}$ und damit der Spezialfall binärer Klassifikation gegeben ist. Alternativ wird häufig auch der Ausdruck $y \in \{-1, 1\}$ gewählt, um die Einteilung in negative und positive Klassen zu verdeutlichen. Doch Klassifikation kann durchaus auch in mehr als zwei Klassen - wobei entsprechend gilt $\boldsymbol{y} \in \{1, \ldots, K\}$ - erfolgen, solange im Rahmen des SL die Menge von Beispielen $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subset \boldsymbol{X} \times Y$ bekannt ist. Nun gilt es diejenige Funktion $\hat{f} : \boldsymbol{x} \to y$ zu finden, die die festgelegt Qualitätsfunktion optimiert.

Lernaufgabe Regression

Eine andere Lernaufgabe im Feld des maschinellen Lernens stellt die Regression dar. Hierbei müssen nun die Zielwerte Y gegeben sein, wobei gilt $y \in \mathbb{R}$. Zudem verfügt man wiederum über die Menge $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subset \boldsymbol{X} \times Y$ von Beispielen. Erneut ist es nun elementar die Funktion $\hat{f} : \boldsymbol{x} \to y$ hinsichtlich einer Qualitätsfunktion zu optimieren.

In diesem Sinne gilt es noch kurz zu klären, was genau die zur Optimierung verwendete Qualitätsfunktion bezeichnet. Synonym kann für diese auch der Begriff der Fehler- oder Verlustfunktion gebraucht werden. Diese ist eine im Prozess der Modellanpassung unverzichtbares Maß zum Errechnen der Diskrepanz zwischen den wahren und den durch das Modell prognostizierten Werten. Wie sich anhand der Terminologie bereits vermuten lässt, dient die Funktion dabei der Bestimmung eines „optimalen“ Modells im Sinne der Minimierung des Verlusts/Fehlers. Als derartige Funktion ist insbesondere der mittlere quadratische Fehler (MSE) bekannt. Doch sieht man sich unweigerlich mit einer breiten Auswahl weiterer potentiell verwendbarer Verlustfunktionen konfrontiert, welche genauer in Kap. 3.4 beleuchtet werden.
Eine weitere Problematik, die angesichts der Verwendung der Qualitätsfunktion zu Tage kommt, ist die folgende: es steht nur eine begrenzte Menge an Daten/Beispielen zur Verfügung, mithilfe derer man ein Modell anpassen kann. Wenn nun aber mit allen diesen Daten ein Modell trainiert wird, so kann die Güte dessen nicht anhand eben dieser verwendeten Daten beurteilt werden. Dies würde zu verfälschten Ergebnissen in dem Sinne führen, dass das Modell spezifisch auf diese Daten angepasst wurde und entsprechend der jeweilige Fehler nach unten verzerrt ist. Die angeklungenen Umstände sollen erste Vorstellungen von der Komplexität des Wissenschaftsfeldes ML vermitteln und Anreize für die nachfolgenden Sektionen schaffen. Letztlich führt doch das Lösen einer Problemstellung zum Auffinden weiterer Hindernisse und es werden Fragen wie die nachfolgenden angestoßen: Welche Modellklassen können/sollten bei der vorliegenden Datensituation verwendet werden? Wie kann die Beurteilung der Güte/Anpassung der Modelle erfolgen und wie können diese untereinander verglichen werden? Welches Modell ist in diesem Sinne also am besten für die jeweiligen Daten geeignet bzw. zeigt die höchste Performance? Kann der Begriff „beste“ überhaupt definiert werden? Wie viel Zeit benötigen die jeweiligen Algorithmen im Umgang mit den vorgegebenen Daten und Dimensionen? Wie sind die Parameter der Prädiktionswerkzeuge zu wählen bzw. ist es ausreichend die Default-Einstellungen der Hyperparameter zu übernehmen? ...
Diese Kette an Fragen kann wohl beliebig fortgesetzt werden und es gilt nun vielmehr Antworten zu erarbeiten. So sei in diesem Zuge hervorgehoben, dass man im Zusammenhang des Vergleichens bzw. Gegenüberstellens verschiedener ML-Methoden unweigerlich auf die Wettbewerbsanalyse und damit auf das s.g. Benchmarking stößt. Aus diesem Grund sollen unter diesem Punkt im nachfolgenden Kap. 3 erste Antworten zu aufgekommenen Fragestellungen offengelegt werden.

3 Benchmark-Studie

Wie der Titel dieser Arbeit bereits impliziert, wird hier der Vergleich unterschiedlicher ML-Methoden bzw. deren Performance im Rahmen der s.g. Benchmark-Studie angestrebt. Damit dies besser verständlich wird, soll zunächst in Sektion 3.1 die Begrifflichkeit des Benchmarkings an sich zugänglich gemacht werden. Des Weiteren ist für derartige Analysen nunmehr eine Aufteilung des Gesamtdatensatzes in drei Unterdatensätze erforderlich: Trainings-, Validierungs- und Testdatensatz. Der Notwendigkeit, sowie dem potentiellen Vorgehen wird in Sektion 3.2 nachgegangen. Die Aufteilung des Datensatzes in mehr als nur Trainings- und Testdatensatz ist insbesondere dem Hyperparameter-Tuning geschuldet, wobei dieses selbst, sowie der Unterschied zwischen Hyper- und Modellparametern in Sektion 3.3 Beschreibung findet.
Im Vordergrund von Benchmark-Studien steht v.a. auch der Vergleich der Performance unterschiedlicher Lernmethoden, nachdem im Bereich des ML's eine Vielzahl derartiger Methodiken zur Verfügung gestellt sind. Die Auswahl geeigneter Methodiken richtet sich zunächst nach der zugrundeliegenden Datensituation und so ist der Typ der Zielvariable mitunter entscheidend für die Art des Lernverfahrens. In diesem Sinne werden in Sektion 3.4 vorrangig Gütemaße für die Beurteilung der Performance von klassifizierenden ML-Methoden dargestellt, jedoch auch kurz einige Gütemaße angesprochen, die die Performance im Falle von Regressionsaufgaben beschreiben.

3.1 Begriffsbestimmung

Aufgrund der breiten Auswahl anwendbarer ML-Methodiken auf eine vorliegende Problemstellung, sieht man sich unweigerlich mit der Frage konfrontiert, welcher der potentiellen Lernalgorithmen mitunter am besten geeignet ist. Wie bestimmt man nun aber den in diesem Sinne „besten" Lerner? In dem Zusammenhang gilt es insbesondere die Schlagworte „Performance" und „Generalisierungsfehler" (GE) aufzuführen. So werden gemäß Hastie et al. (2008, S. 222) vorrangig zwei unterschiedliche Ziele angestrebt:

- Modell-Wahl: Bestimmung der Performance-Güte unterschiedlicher Modelle für die Auswahl des in diesem Sinne „optimalen" Lernmethodik
- Modell-Beurteilung: Bestimmung des GE's dieses ausgewählten Modells (Prädiktionsfehler bei neuen Daten, abgeleitet vom Englischen „Generalization Error")

Diese Zielvorstellungen lassen sich als Kernpunkte von Benchmark-Studien ausmachen. Dabei können durch Anwendung der Lernmethoden auf ein oder mehrere Datensätze die einzelnen Algorithmen bzw. deren Performance miteinander verglichen, Hyperparameter bestimmt und den Methoden an sich gewissen Sensitivitätsanalysen unterzogen werden. Diese Analysen stützen sich zumeist auf ausgewählte Gütemaße, doch auch die computationale Lauf- bzw.

Rechenzeit und Speicherauslastung können von zentralem Interesse sein - werden in dieser Arbeit allerdings nicht ausweitend betrachtet. Aufgrund der derartigen Gegenüberstellung von ML-Methoden lässt sich das Benchmarking also auch gut als „Instrument der Wettbewerbsanalyse“ [s. Wübbenhorst (2018)] deklarieren, womit eine allgemeine Bezeichnung über das Feld des ML's hinaus getan ist und insbesondere das bewertende Vergleichen der Methodiken zum Ausdruck gebracht werden soll. Die nachfolgenden Erklärungen dieser Sektion sind nun in Anlehnung an Hastie et al. (2008, Kap. 7) und Eugster (2011, Kap. 1) zu verstehen.
In Situationen, bei denen eine große Datengrundlage vorhanden ist, tritt man den zuvor genannten Zielvorstellung der Modell-Wahl und -Beurteilung am besten entgegen, indem man eine Aufteilung des Datensatzes in drei (disjunkte) Teile vornimmt: Trainings-, Validierungs- und Testdatensatz. Dieser Aspekt soll nun in Sektion 3.2 genauer beleuchtet werden.

3.2 Aufteilung in Trainings-, Validierungs- und Testdatensatz

Bereits mehrfach fand die Notwendigkeit der Aufteilung eines gegebenen Datensatzes in Subdatensätze für die konkrete Evaluierung von ML-Methodiken Anklang. Während die Aufteilung in Trainings- und Testdaten allgemein geläufig ist, sei darauf aufmerksam gemacht, dass im Kontext des Benchmarkings zumeist eine weitere Unterteilung des Gesamtdatensatzes $\mathcal{D}$ von Nöten ist: v.a. im Hinblick auf das Hyperparameter-Tuning, welches in Sektion 3.3 detailliert Beschreibung findet, wird zudem ein s.g. Validierungsset verlangt. Das neu eingeführte Validierungsset ist dabei nichts anderes als ein indirekter Schritt der Anpassungsprozedur und damit quasi nur ein weiteres Benchmark-Experiment für das Hyperparameter-Tuning.

3.2.1 Notwendigkeit der Unterteilung und Vorgehen

Warum muss eine derartige Unterteilung der Daten erfolgen? Dies lässt sich gemäß Zheng (2015) derart begründen: sobald man sich im Bereich statistischer Modellierung befindet, stößt man unweigerlich auf die Annahme von Zufälligkeit und damit auf den Gedanken, dass gegebene Daten zufällig verteilt sind. Man geht dabei von einem s.g. Daten-generierenden Prozess (DGP) aus, also einem (wahren) zugrundeliegendem Modell mit unbekannten Parametern. Man strebt nun an dieses Modell zu ermitteln, wobei es basierend auf den gegebenen, aber zufälligen Daten trainiert wird und entsprechend selbst zufällig ist. Das so erlernte Modell wird anhand eines weiteren beobachteten, aber wiederum zufälligen Subdatensatzes evaluiert und demzufolge sind auch die Testergebnisse zufällig. Wie dadurch nun deutlich wird, ist es für unverzerrte Ergebnisse dringend erforderlich, dass das Testen des Modells auf Basis anderer Daten - als für die Anpassung des Modells verwendet wurden - stattfindet, um so statistische Unabhängigkeit garantieren zu können. Aus demselben Grund müssen auch die Validierungsdaten andere sein als diejenigen, die innerhalb des Test- oder Trainingssets Verwendung gefunden haben, sodass ohne Verzerrung ein GE geschätzt und damit ein Pro-

gnosefehler für neue Daten ermittelt werden kann.
Eine noch ungeklärte Problematik stellt dabei die Frage nach der Größe der jeweiligen Subdatensätze dar. Hierzu kann allerdings keine generelle Lösung offen gelegt werden, doch empfiehlt Hastie et al. (2008, S. 222) beispielsweise, dass Validierungs- und Testdatensatz zusammen in etwa 50 % der Gesamtdaten - entsprechend jeweils 25% den beiden Subdatensätzen zugeteilt wird - ausmachen sollten und die übrige Hälfte als Trainingsdatensatz deklariert wird.
Angenommen der dem Anwender vorliegende Datensatz ist bereits in die drei geforderten Subdatensätze unterteilt und man betrachte vorrangig Trainings- und Validierungsdatensatz, während der Testdatensatz vorerst unberücksichtigt bleibt. Zufällig könnte es passieren, dass einer der ersteren Datensätze eine größere Menge an Ausnahmefällen/Ausreißern beinhaltet, was sich je nach betroffenem Datensatz dann in Form von Verzerrung der Modellschätzung oder der Testergebnisse bzw. Güte des Modells auswirkt. Entsprechend bedarf es einer Methodik, um dieser Problematik entgegenzutreten. Hierzu bedient man sich s.g. Resampling-Methoden. Diese Art des Vorgehens lässt sich allerdings auch in der Generierung multipler, unabhängiger Datensätze begründen, wenn beispielsweise nur eine kleine Anzahl an Gesamtdaten vorhanden ist und dennoch valide Ergebnisse angestrebt werden. Dieser Thematik soll in nachfolgender Untersektion 3.2.2 nachgegangen werden.
Bevor einige derartige Resampling-Methoden beschrieben werden, soll noch einmal kurz verdeutlicht werden, wie bzw. wozu die einzelnen Datensätze verwendet werden. Hierbei ist insbesondere Bezug genommen auf Zheng (2015, S. 24 u. 29). Die erste Phase ist dabei an der Modellselektion - auch als Prototyping Phase bezeichnet - orientiert, sodass die Performance von ein oder mehreren ML-Modellen basierend auf einem oder mehreren Validierungsdatensätzen einander gegenübergestellt wird. Bildlich kann diese erste Phase anhand der sich im Anhang befindenden Abb. 8.3 nachvollzogen werden. Dabei werden zuerst in einer Art Meta-Prozess bzw. inneren Optimierungsprozess unterschiedliche Settings der Hyperparameter einander gegenüber gestellt und schließlich dasjenige gewählt, mit welcher sich die beste Performance zeigt. Anschließend werden diese Hyperparameter genutzt, um im letzten Schritt der ersten Phase bzw. des Prototypings ein neues Modell zur Bestimmung der Modellparameter zu trainieren. Hierzu greift nun auf den gesamten Satz für das Training zur Verfügung stehender Daten - also insbesondere auch den Daten, die zuvor für die Validierung zurückgehalten wurden - zurück. So wird das finale Modell bestimmt und kann in der zweiten Phase anhand der Testdaten beurteilt werden. Das beschriebene Vorgehen bildet entsprechend die Basis von Benchmark-Studien.
Mit dieser Verdeutlichung kann sich nun zunächst allgemein unterschiedlichen Arten des Resamplings zugewandt werden. Daran anschließend soll das s.g. genestete Resampling geschildert werden, mithilfe dessen alle Prozessierungsschritte der Modellbildung - Preprocessing und Modellselektion - in das Resampling integriert werden können.

3.2.2 Resampling-Methoden

Anhand dieser Untersektion sollen zunächst einige grundlegende Gedankengänge hinsichtlich des Resamplings bzw. der Partitionierung eines Datensatzes mithilfe von Molinaro et al. (2005, Kap. 2), sowie Bischl et al. (2012, Kap. 2) zugänglich gemacht werden. Wie bereits hervorgehoben wurde, gilt es je nach Art des Outcomes zwischen Regressions- und Klassifikationsaufgaben zu unterscheiden. Konsequenter Weise hat man bei Ersterer also eine Zuordnungsregel bzw. Prädiktor $\hat{f}(\cdot)$ der vom Feature-Raum $\mathcal{X}$ beispielsweise in die reellen Zahlen $\hat{f} : \mathcal{X} \to \mathbb{R}$ überführt, sodass der prognostizierte Outcome basierend auf den beobachteten Einflussgrößen $\boldsymbol{x}$ dann die Form $\hat{y} = \hat{f}(\boldsymbol{x})$ annimmt. Im Falle einer kategorialen Zielgröße von $k = 1, \ldots, K$ Kategorien soll dagegen der Prädiktor $\hat{f}$ eine Partitionierung des Raumes $\mathcal{X}$ in K disjunkte, erschöpfende Gruppen G_k gewährleisten, sodass $\hat{y} = k$, falls $\boldsymbol{x} \in G_k$.
In Anlehnung an die zuvor getätigten Beschreibungen lässt sich dann die Regel $\hat{f}(\cdot)$ schreiben als $\hat{f}(\cdot|\mathbb{P}_N)$, wobei $\mathbb{P}_N$ der empirischen Verteilung von $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subset \boldsymbol{X} \times Y$ entspricht und die Abhängigkeit der hervorgebrachten Regel von den beobachteten Daten widerspiegelt. Demzufolge kann dann auf eine Verlustfunktion zurückgegriffen werden, um die Performance einer Regel zu beurteilen. Eine gängige Verlustfunktion im Rahmen von Regressionsproblemen wäre beispielsweise der quadrierte Verlust $L(y, \hat{f}(\boldsymbol{x})) = [y - \hat{f}(\boldsymbol{x})]^2$ und für Klassifikationen die Indikator-Verlustfunktion $L(y, \hat{f}(\boldsymbol{x})) = I[y \neq \hat{f}(\boldsymbol{x})]$, wobei innerhalb dieser Funktionen zusätzlich auch differentiale Missklassifikationskosten berücksichtigt werden können. Für jeglichen Typus des Outcomes kann schließlich der erwartete Verlust bzw. das s.g. Risiko in Abhängigkeit von der gemeinsamen Wahrscheinlichkeitsverteilung $\mathbb{P}(\boldsymbol{x}, y) = \mathbb{P}_{\boldsymbol{x},y}$ auf $\mathcal{X} \times \mathcal{Y}$ definiert werden als:

$$\tilde{\theta}_{Risk} = R(\hat{f}(\cdot), \mathbb{P}) = \mathbb{E}_P[L(y, \hat{f}(\cdot))] = \int L(y, \hat{f}(\boldsymbol{x}))d\mathbb{P}(\boldsymbol{x}, y) \tag{3.1}$$

Die in Gl. 3.1 dargestellte Regel ist konstruiert und evaluiert auf Basis der gemeinsamen Verteilung $\mathbb{P}_{\boldsymbol{x},y}$, sodass $\tilde{\theta}_{Risk}$ hier als asymptotisches Risiko zu bezeichnen ist. Problematisch ist allerdings, dass $\mathbb{P}$ in der Realität nicht bekannt ist, weshalb eine Regel, die auf den einzelnen Beobachtungen $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ basiert, einen erwarteten Verlust bzw. hier bedingtes Risiko - auch besser bekannt als Generalisierungsfehler (GE) - aufweist, der wie nachfolgend dargestellt definiert werden kann:

$$\tilde{\theta}_{GE} = R(\hat{f}(\cdot|\mathbb{P}_N), \mathbb{P}) = \mathbb{E}_{\mathbb{P}}[L(y, \hat{f}(\cdot|\mathbb{P}_N))] = \int L(y, \hat{f}(\boldsymbol{x}|\mathbb{P}_N))d\mathbb{P}(\boldsymbol{x}, y) \tag{3.2}$$

Letztendlich gibt es zwei Anreize auf die Evaluation mittels des bedingten Risikos zurückzugreifen: einerseits für Modellselektion und andererseits für die Zugänglichkeit zur Performance. Ersteres strebt das Ermitteln desjenigen Modells an, welches das bedingte Risiko minimiert. Zweiterer genannter Punkt verfolgt das Ziel der Berechnung des GE's für ein gegebenes Modell. Ein idealer Ausgangspunkt wäre also über unabhängige Datensätze für die Modellselek-

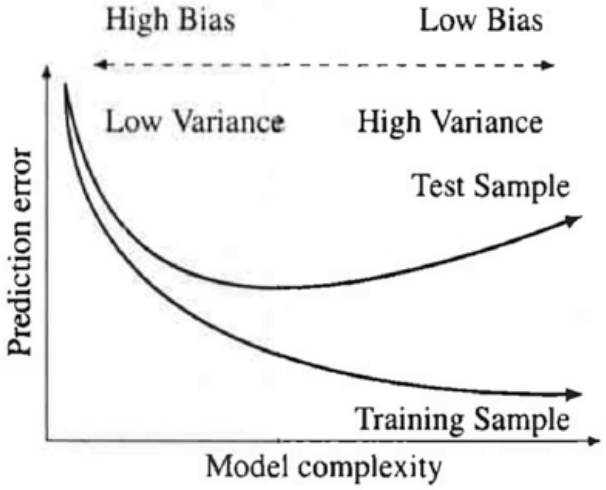
tion, sowie Kalkulation des GE's zu verfügen. Stattdessen muss man zumeist auf Grundlage eines beobachteten Datensatzes die Modellkonstruktion, - selektion, ebenso wie Beurteilung der Performance vollführen.

Resubstitution

Das einfachste Vorgehen in diesem Sinne wäre dann die Schätzung des konditionalen Risikos anhand des Resubstitutionsfehlers, wie sich dieser mathematisch formuliert in Gl. 3.3 findet. Dabei würden alle N Beobachtungen genutzt, um jeweils die Modellanpassung, sowie die Schätzung des GE's zu bewerkstelligen. Damit ergibt sich allerdings eine deutliche Unterschätzung des GE's, nachdem dieser auf Basis derjenigen Beobachtungen abgeschätzt wird, die konkret auch für die Modellkonstruktion genutzt wurden. So wird insbesondere auch die Genauigkeit eines Modells überschätzt und man erhält verzerrte Ergebnisse (Bias).

$$\hat{\theta}_{GE}^{RS} = R(\hat{f}(\cdot|\mathbb{P}_N), \mathbb{P}_N) = \int L(y, \hat{f}(\boldsymbol{x}|\mathbb{P}_N)) d\mathbb{P}_N(\boldsymbol{x}, y) \tag{3.3}$$

Die verzerrte Schätzung des GE's bedingt - falls diese Schätzungen zur Modellselektion herangezogen werden - die bevorzugte (ungewollte) Wahl von stark Daten-adaptiven, komplexeren Modellen. Damit kommt das allgemeine Problem des Bias-Varianz-Tradeoff's zum Vorschein: je komplexer und stärker das Modell an die Trainingsdaten angepasst ist, desto niedriger ist dessen Bias. Jedoch tendieren diese Modelle zur Überanpassung (Overfitting) an die Trainingsdaten, sodass zwar der Resubstitutionsfehler niedrig angesiedelt ist, allerdings derartige Modelle nicht unbedingt in der Lage sind, Prädiktionen zu zukünftigen Daten mit ausreichender Genauigkeit zu bewerkstelligen, was sich als zunehmender Abstand/Diskrepanz zwischen Trainings- und Testfehler in Abb. 3.1 verstehen lässt.



Quelle: (Bischl et al., 2012, Kap. 2)

Abb. 3.1: Bias-Varianz-Tradeoff: Mit zunehmender Modellkomplexität steigt die Wahrscheinlichkeit der Überanpassung (Overfitting) an die Trainingsdaten

Diese verzerrte Schätzung begründet die Nutzung von Resampling- bzw. Partitionierungsmethoden, wie derartige nachfolgend präsentiert werden. Dabei wird im weiteren Verlauf dieser Untersektion zunächst nur die möglichst ausschöpfende Nutzung der Informationen aus Trainings- und Validierungsdaten angestrebt. Die zusätzliche Einbindung der Modellselektion in Hinsicht auf das Resampling findet in Untersektion 3.2.3 schließlich Anklang.

Holdout-Methode

Die Holdout- oder Learning-Test-Split-Methode, die anhand von Molinaro et al. (2005, S. 5) und Witten et al. (2011, Kap. 5.3) nur kurzen Eingang finden soll, ist die wohl am einfachsten vorzunehmende Partitionierung in Trainings- und Validierungsdaten. Allerdings verlangt es eine (subjektive) Verteilung der Masse oder anders ausgedrückt des Zuweisungsverhältnisses der Daten zu den zwei Subsets. Um gerade hinsichtlich selten vorkommender Klassen gewährleisten zu können, dass diese innerhalb der einzelnen Subsets in einem etwa gleich großen Verhältnis auftreten, kann auf Prozeduren der Stratifizierung und damit auf die s.g. stratifizierte Holdout-Methode zurückgegriffen werden. Hierbei bezeichnet Stratifizierung eine Methodik, die im Hinblick auf kategoriale Variablen sicherstellen sollen, dass alle vorkommenden Wert bzw. Kategorien in etwa dasselbe Verhältnis innerhalb von sowohl Trainings-, als auch Test-/Validierungsdatensatz wie im Gesamtdatensatz aufweisen.
Trotz der relativ simplen Handhabung und computationalen Einfachheit sollten besonders zwei Quellen der Verzerrung hervorgehoben werden: derjenige Bias, der durch die ausschließliche Zuweisung einer Beobachtung zu Validierungs- oder Trainingsdatensatz entsteht und die durch die geringe(re) Größe des Trainingsdatensatzes bzw. die nicht erschöpfende Nutzung des vorliegenden Datenmaterials erzeugte Verzerrung in der Modellkonstruktion. Somit empfiehlt es sich auf andere Resampling-Methoden - wie etwa die nachfolgenden - zurückzugreifen.

Kreuzvalidierung (CV)

Gemäß Zheng (2015, S. 22) ist die Kreuzvalidierung - abgeleitet von dem englischen Begriff „Cross-Validation“ (CV) - eine einfache Methodik für die Generierung von Test- und Validierungssets für den Prozess des Hyperparameter-Tunings. Es gibt unterschiedliche Arten, wobei die wohl bekannteste und am meisten genutzt die k-fache CV ist. Hierbei wird ein gegebener Trainingsdatensatz in k Einheiten geteilt. Dabei ist es Witten et al. (2011, S. 153) zufolge nicht zwingend erforderlich, dass die Subdatensätze die exakt gleiche Größe aufweisen, solange das Verhältnis der einzelnen auftretenden Klassen in den verschiedenen Sets in etwa gleich ist. Für ein gegebenes Hyperparameter-Setting wird nun immer eines der k Teile ausgelassen und als Validierungssatz verwendet. Entsprechend wird ein Modell auf Basis der $k-1$ Datensätze trainiert und anschließend über das währenddessen nicht betrachtete Subset evaluiert. Die zugehörige Performance für die festgesetzte Hyperparameter-Konstellation ergibt sich als das Mittel aus allen k Durchgängen. Dieser Prozess wird für alle möglichen Kombinationen

der Hyperparameter durchgeführt und anschließend dasjenige Hyperparameter-Setting ausgewählt, welches die höchste Güte in Anbetracht des k-fachen Mittels aufweist. Auch bei Anwendung der CV kann zusätzlich Stratifizierung Berücksichtigung finden.
Wenngleich dies nach wie vor innerhalb der Kreise von ML-Experten eine debattierte Thematik darstellt, hat sich die 10-fache CV doch zu einer Standardmethode bei praktischen Anwendungen entwickelt. Sollte die Annahme bestehen, dass auch die 10-fache CV keine ausreichend genaue Fehlerschätzung hervorbringt, so kann ausweitend auf die wiederholte (repeated) k-fache CV zurückgegriffen werden was also bedeutet, dass wiederholt k-fache CV betrieben wird und über diese Wiederholungen/Iterationen gemittelt wird. Beispielsweise wird durch 10-mal wiederholte 10-fache CV der entsprechende Lernalgorithmus 100 ausgeführt basierend auf jeweils 9/10 des Originaldatensatzes.
Eine sich aus der CV ergebende Resampling-Variante ist die Leave-one-Out-Kreuzvalidierung, abgekürzt auch bezeichnet als LOOCV. Dabei ist k gleich der Anzahl der Datenpunkte im Datensatz - hier also N-fache CV -, wobei entsprechend jede Instanz einmal ausgelassen wird und der Lernalgorithmus auf Basis der verbleibenden Einheiten trainiert wird. Dieser Spezialfall der CV gewinnt aus zwei Gründen an Attraktivität: zum einen wird in jedem Trainingsdurchlauf auf die größtmögliche Menge an Daten zurückgegriffen, zum anderen ist diese Prozedur deterministisch, nachdem kein zufälliges Ziehen involviert ist. Offensichtlich ist diese Methodik allerdings computational rechenintensiv und kann zudem nicht stratifiziert werden. Abgesehen davon erzielt im Vergleich zur LOOCV die k-fache CV i.A. bessere Ergebnisse der Testfehlerrate, was sich auf Basis des Bias-Varianz-Trade-Offs begründen lässt. Entsprechend kann gemäß Boulesteix et al. (2008, S. 87) durch LOOCV zwar zumeist ein nahezu unverzerrter Schätzer des Fehlers hervorgebracht werden, jedoch gleichzeitig auch eine hohe Varianz aufweisen, nachdem die Lernsets große Ähnlichkeit zueinander zeigen.

Bootstrap (BS)

Bootstrap (BS) stellt eine eng mit der CV verwandte Resampling-Methode dar und fußt auf der Idee von Jackknife, was essentiell der LOOCV entspricht. Auch die Konzepte von Bootstrap sollen nun folgenden in Anlehnung an Zheng (2015, S. 23), sowie Witten et al. (2011, Kap. 5.4) beschrieben werden.
Der wesentliche Unterschied der bei BS zum Vorschein kommt, ist, dass hierbei im Gegensatz zur CV mit Zurücklegen gezogen wird. Während also innerhalb Letzterer jeder der Beobachtungen nur einem einzigen der k Subdatensatz zugeteilt wird, ist dies beim BS nicht der Fall. Stattdessen wird aus den N Beobachtungen N-mal zufällig und jeder Datenpunkt mit gleicher Wahrscheinlichkeit aus dem Gesamtdatensatz gezogen, dem Trainingsset zugeteilt und anschließend wieder als Teil des Ausgangsdatensatzes angesehen, sodass dieser Datenpunkt potentiell wiederholt in den neu entstehenden Subdatensatz gelangen kann.
Warum wird ein derartiges Vorgehen gewählt? Man nimmt an, dass der beobachtete gan-

ze Datensatz eine Stichprobe aus der realen, aber nicht bekannten Verteilung ist - womit man wiederum auf den DGP Bezug nimmt. Jedoch steht mit dem gegebenen Datensatz ein Repräsentant der zugrundeliegenden Verteilung zur Verfügung und damit eine empirische Verteilung der Daten. Entsprechend werden durch die BS-Simulationen neue Stichproben aus der empirischen Verteilung gezogen, wobei diese jeweils mit Zurücklegen zu ziehen sind, da sich anderenfalls die empirische Verteilung nach jedem Zug ändern würde.
Es lässt sich zeigen, dass das erwartete Verhältnis von Beobachtungen, welche in die Teildatensätze gelangen, gegenüber denen, die nicht gezogen werden in etwa 2/3 ist. Nachdem nämlich für eine gegen unendlich strebende Wiederholung der BS-Prozedur eine Poisson-Verteilung mit $\lambda = 1$ approximiert wird, ergibt sich die Wahrscheinlichkeit für eine Beobachtung mindestens einmal gezogen zu werden als $1 - (1 - 1/n)^n \approx 1 - 1/e = 0.632$.
Eine Möglichkeit des Vorgehens wäre also das Modelle auf Basis der einzelnen Instanzen der aus BS-Prozeduren hervorgehenden Teildatensätze zu trainieren und anschließend die Ergebnisse mittels der nicht verwendeten Beobachtungen zu validieren, sowie diese abschließend zu mitteln. Die sich ergebenden Effekte sind denen der CV zumeist sehr ähnlich.
Wie Boulesteix et al. (2008, S. 88) unterstreicht sind jedoch BS-Schätzer hinsichtlich ihrer Fehlerrate nach oben verzerrt, nachdem die Modellkonstruktion nur auf circa 63.2% der zu beobachteten Daten beruht. Dies veranlasste schließlich die Konstruktion von Schätzprozeduren, welche die BS-Fehlerrate und die der Resubstitution, welche ihrerseits nach unten verzerrt ist, zu kombinieren und so entstanden das s.g. 0.632 BS, sowie das 0.632+ BS. In detaillierterer Form lassen sich diese zwei Erweiterung des BS's beispielsweise anhand von Boulesteix et al. (2008, S. 88) nachvollziehen. Es lässt sich zeigen, dass diese beiden Schätzmethoden einen niedrigeren Bias aufweisen als einfaches BS und sogar als das s.g. Subsampling, welches kurz in der nachfolgenden Untersektion Ansprache findet.

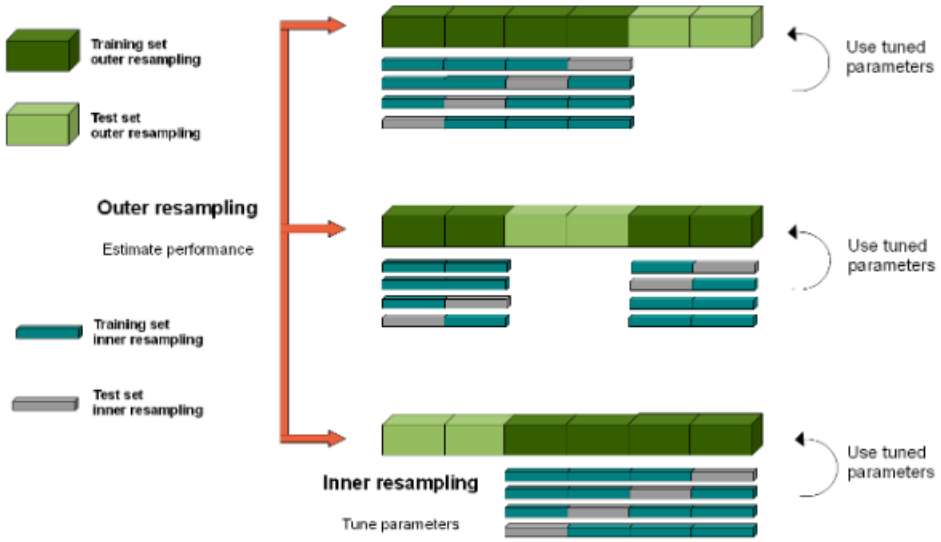
Subsampling bzw. Monte-Carlo Kreuzvalidierung (MCCV)

Die Monte-Carlo Kreuzvalidierung (MCCV), welche häufig auch als Random-Splitting oder Subsampling bezeichnet wird, kann als weitere Methodik des Resamplings aufgeführt werden. Insbesondere lässt sich für diese Methode eine niedrigere Varianz in der Schätzung der Fehlerrate erkennen als für beispielsweise die LOOCV, nachdem die durch MCCV hervorgebrachten Testsets eine deutlich geringere Korrelation aufweisen als die der LOOCV. Allerdings ist dafür bei MCCV ein nach oben gerichteter Bias zu erkennen, da Prädiktionsregeln nicht auf Basis aller N Beobachtungen erstellt werden. Genauere Beschreibung zu dieser Resampling-Methodik finden sich u.a. wieder bei Boulesteix et al. (2008). Nachfolgend soll nun das Resampling im Kontext von drei Subdatensätzen, wie dies im Bereich des Benchmarkings zumeist in Erscheinung tritt, verdeutlicht werden.

3.2.3 Genestetes Resampling

Bereits mehrfachen Anklang fand die Tatsache, dass Benchmark-Studien zumeist mit der Aufteilung des Gesamtdatensatzes in drei Teile - also Trainings-, Validierungs- und Testsets - einhergeht. Hinzu kommt das Vergleichen, sowie die Evaluation der auf Grundlage dieser Subdatensätze angepassten - mitsamt Hyperparameter-Tuning oder Kovariablen-Selektion - unterschiedlichen ML-Methoden. Insgesamt kristallisiert sich damit ein recht komplexer Prozess der Modellanpassung(en), einschließlich -selektion heraus. Nachdem Bischl et al. (2012, Kap. 3) folgend die optimalen Hyperparameter zumeist datenabhängig sind, können die Werte dieser Kontrollgrößen nicht ohne entsprechendes Heranziehen der zugrundeliegenden Daten festgesetzt werden. Andererseits dürfen die einzelnen Schritte der Modellselektion nicht auf Basis desselben Resampling-Sets durchgeführt werden, wie dieses für die Evaluation des Modells selbst genutzt wurden. Begründung hierfür stellen die potentiell stark verzerrten Schätzungen der Performance dar. Dieser Effekt wird häufig als „Training auf dem Testset“ bezeichnet, da durch die wiederholte Evaluierung des Modells auf Basis der Testdaten Informationen, welche deren Struktur beschreiben, Einzug in das Modell nehmen und so zu Overfitting führen.
Einen möglichen Ausweg aus der geschilderten Problemstellung biete das genestete Resampling. Hierbei gilt es den Prozess der Modellselektion als Teil der Modellanpassung zu erachten und aus diesem Grund für jeden der Trainingssets zu wiederholen. Nachdem diese Art der Modellselektion zumeist selbst einer Resampling-Methode bedarf, mündet der gesamte Prozess in einer genesteten Form des Resamplings. Entsprechend wird in der äußeren (*outer*) Schleife das Modell evaluiert bzw. die Performance mehrerer angewandten Algorithmen einander gegenüber gestellt, wohingegen in der inneren (*inner*) Schleife jeder Trainingssatz erneut dem Resampling unterzogen wird, um ein angemessenen Lerner bzw. das optimale Setting der Hyperparameter auszuwählen. Letzteres stellt also selbst ein Optimierungsproblem dar und soll mithilfe der nachfolgenden Sektion 3.3 besser zugänglich gemacht werden.
Zusammenfassend werden die optimierten Hyperparameter genutzt, um eine Modellanpassung anhand des kompletten Trainingsdatensatzes vorzunehmen und schließlich den GE, der sich durch Anwendung des Modells auf die Testdaten der äußeren Resampling-Strategie ergibt, zu berechnen. Die sich so ergebende Prozedur des genesteten Resamplings lässt sich bildlich anhand von Abb. 3.2 nachvollziehen. Hierbei ist in der äußeren Schleife eine 3-fache CV für das Hyperparameter-Tuning genutzt, sowie eine 4-fach CV innerhalb der inneren Schleifen. Entsprechend hat man in der äußeren Schleife drei Paare von Trainings-/Testsets. Zunächst wird für jedes äußere Trainingsset das Hyperparameter-Tuning auf Basis der inneren Schleife durchgeführt. Folglich ergibt sich ein Set ausgewählter Hyperparameter für jedes äußere Trainingsset. Anschließend wird der jeweilige Lerner an jeden äußeren Trainingssubdatensatz unter Nutzung der entsprechenden, ausgewählten Hyperparameter angepasst und die Performance evaluiert. Offensichtlich ist diese Art des Vorgehens computational sehr aufwendig,

doch Bischl et al. (2012) zufolge zum jetzigen Zeitpunkt mitunter die einzige Möglichkeit unverzerrte Ergebnisse bezüglich des GE's, sowie dem Vergleich der Performance verschiedener ML-Methoden sicherzustellen.



Quelle: (Bischl et al., 2016, Kap. *Nested Resampling*)

Abb. 3.2: Beispielhafte Veranschaulichung zum genesteten Resampling: 3-fache CV in äußerer Schleife und 4-fache CV in innerer Schleife (Hyperparameter-Tuning)

Innerhalb dieser Sektion wurden nun mehrfach s.g. Hyperparameter bzw. deren Tuning angesprochen. Aus diesem Grund scheint es erforderlich deren Zweck und das genauer Vorgehen beim Tuning genauer zu beleuchten. Man ziehe hierfür die nachfolgende Sektion 3.3 heran.

3.3 Hyperparameter und deren Tuning

Nachfolgend sollen im Rahmen dieser Sektion der Unterschied von Hyper- zu Modellparametern verdeutlicht werden, ebenso wie der Zugang zu dem bereits mehrfach angesprochenem Tuning ersterer Größen vermittelt werden soll. Die hier enthaltenen Erklärungen fußen im Wesentlichen auf Zheng (2015, S. 27-34).

3.3.1 Abgrenzung der Hyperparameter von Modellparametern

Zunächst gilt es eine klare Abgrenzung der Hyperparameter von Modellparametern offenzulegen. Dabei sind ML-Methoden als mathematische Funktionen bzw. Algorithmen zu verstehen, mittels derer im Rahmen des SL's der Zusammenhang zwischen einer Zielvariable und potentieller Einflussgrößen ausgemacht werden soll, wobei hierzu - innerhalb der Parametergestützten Modellierung - entsprechende Modellparameter angepasst werden. Diese Parameter werden innerhalb der Trainingsphase bestimmt und ergeben sich durch die Optimierungsprozedur zur Ermittlung desjenigen Modells mit der „besten" Anpassung an die gegebenen Daten.

Dagegen stellen Hyperparameter Größen zur Regularisierung dar und werden außerhalb der Trainingsphase ermittelt. Diese Regularisierung tritt je nach betrachtetem Modell in unterschiedlicher Form in Erscheinung. Beispielsweise in der Ridge-Regression oder bei Anwendung von Lasso durch die Gewichtung des Bestrafungsterms, bei Entscheidungsbäumen durch die Tiefe des Baumes bzw. die Anzahl der Knoten, beim Boosting durch die Anzahl der maximal zu betrachtenden Features, etc. Diese Auflistung kann nahezu beliebig fortgeführt werden und sollte im Wesentlichen verdeutlichen, dass Hyperparameter als Kontrollsequenzen der eigentlichen Algorithmen dienen und damit die Kapazität eines Modells steuern, wie etwa die Flexibilität oder die Anzahl der Freiheitsgrade. Allerdings steuern diese teilweise auch den Trainingsprozess selbst, da die Lernmethoden auf Optimierungsprozeduren - beispielsweise der Optimierung der Verlustfunktion - basieren und zum Teil eigene Hyperparameter beinhalten. Beispielhaft lässt sich etwa die maximale Anzahl an Bäumen aufführen, die in Forests oder geboosteten Entscheidungsbäumen Berücksichtigung finden dürfen.

3.3.2 Tuning der Hyperparameter

In Form der Hyperparameter steht eine große Anzahl festzusetzender Einstellungen aus, welche noch dazu aus einem breiten Intervall möglicher Werte ausgewählt werden müssen. Um dies zu bewerkstelligen, bedient man sich des Hyperparameter-Tunings, wobei man außerhalb des ML's auch von (Algorithmus-) Konfiguration spricht. Dieses Tuning kann als eine Art Meta- bzw. übergeordnete Lernaufgabe angesehen werden und hat - wie bereits verdeutlicht wurde - vor der Trainingsphase zu erfolgen. Hierbei wird bei dem Tuning für jedes Set möglicher Hyperparameter-Kombinationen ein Modell als innerer Optimierungsprozess trainiert und auf Basis dessen das beste Setting als Ausgabe des Tunings bestimmt. Dieser innere Trainingsprozess beruht auf der Evaluation der Ergebnisse von beispielsweise kreuzvalidierten Datensätzen. Damit sei auf die bereits beschriebene Thematik der Trainings- und Validierungsdatensätze aus Sektion 3.2 verwiesen. Das eigentliche Trainieren des Modells erfolgt schließlich erst im Anschluss an die Wahl der in diesem Sinne optimalen Hyperparameter.
Wie nun bereits deutlich wird, ist das Tuning der Hyperparameter als Optimierungsproblem eine komplexe Aufgabe. Insbesondere nachdem jeder Parameter auf einem vordefinierten Intervall gesucht werden muss und so schließlich durch Kombination unterschiedlicher Settings einen riesigen, wenn nicht gar unendlichen Suchraum - insbesondere auch im Hinblick auf kontinuierliche Hyperparameter - mit sich führt. Um dieser Problematik entgegenzutreten, betrachtet man nur bestimmte bzw. „vorab errechnete“ Punkte innerhalb des Suchraumes. Die Wahl der Punkte kann hierbei auf unterschiedliche Arten erfolgen und wird als Optimierungsprozedur bezeichnet. Hinsichtlich der Spezifikation des Optimierungsalgorithmus sind im Wesentlichen wohl „Grid Search“ und „Random Search“ geläufig, jedoch soll hier insbesondere auch die modell-basierte Optimierung (MBO) angeführt werden und zum Abschluss der Sektion noch einige weitere Optimierungsalgorithmen genannt werden.

Grid Search

Wie der Name der Methode - aus dem Englischen „Gitter-, Rastersuche“ - vermuten lässt, wird der Suchraum in ein Gitter unterteilt und das Tuning nur für die auf dem Gitter liegenden Punkte durchgeführt. Dies impliziert die Unterteilung des Suchraums in - die gleiche Entfernung voneinander aufweisende - Punkte, womit sich durch gedankliches Verbinden dieser eine Gitterform ergibt. Man ziehe hierzu beispielsweise den Hyperparameter der zu optimierenden Knotenanzahl in einem Entscheidungsbaum, der auf einem Gitter von $10, 20, 30, \ldots, 150$ Knoten gesucht wird, in Betracht.
Folglich kommt die Notwendigkeit zutage das Intervall des Suchraums und damit entsprechend Minimum und Maximum zu definieren. Sollte der resultierende optimierte Hyperparameter am Rande des Intervalls liegen, so empfiehlt es sich das Intervall in die entsprechende Richtung zu vergrößern und einen erneuten Suchlauf zu starten. Wenngleich Grid Search relativ einfach zu verstehen ist, so ist diese Methode gemäß Zheng (2015, S. 31) in Anbetracht der gesamten Computationszeit als äußerst aufwendig zu erachten, wobei dem aufgrund einfach vorzunehmender Parallelisierung entgegengesteuert werden kann.

Random Search

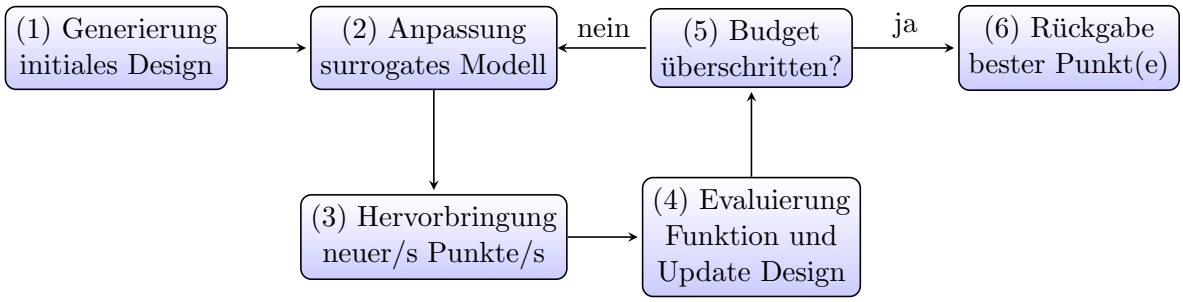
Random Search stellt eine Variation der vorhergehend beschriebenen Methodik dar. Anstatt über das gesamte Gitter hinweg wird dabei nur über eine zufällig ausgewählte Stichprobe an Punkten evaluiert. Damit ist das zufällige Suchen auch computational deutlich sparsamer als das Suchen über ein festes Gitter hinweg, nachdem mit dieser Methodik die zu evaluierenden Punkte nicht errechnet und so spezifisch ausgewählt werden müssen. Wenngleich man früher glaubte, dass Random dem Grid Search deutlich unterlegen ist, so konnte im Laufe der Entwicklungsgeschichte des ML's Bergstra und Bengio (2012) sogar zeigen, dass dem nicht zwangsweise so ist, sondern sogar nahezu gleiche Performance der beiden Methoden erreicht werden kann. Dies gilt insbesondere dann, wenn die Region nahe des Optimums relativ groß ist. Genauere Ausführungen, sowie mathematische Beweisführungen sollen hier nun aber nicht weiter fokussiert werden, sondern dahingehend auf Bergstra und Bengio (2012) verwiesen sein.
Zusammenfassend kann festgehalten werden, dass sich Random Search als Methode für das Hyperparameter-Tuning eignet, da es relativ leicht parallelisiert werden kann, jedoch im Vergleich zum Grid Search weniger Versuche für die Ermittlung des Optimums benötigt und dennoch nahezu gleich gute Ergebnisse liefert. Neben den zwei bisher aufgeführten Methodiken gibt es zudem noch s.g. schlaues/smartes Hyperparameter-Tuning. Allerdings sind inbegriffene Methoden zumeist kaum oder überhaupt nicht parallelisierbar und damit computational hoch aufwendig und sollen nun hier nicht weiter von Relevanz sein.

(Sequentielle) Modell-basierte [(S)MBO]

(Sequentielle) Modell-basierte Optimierung [(S)MBO] - der Bayes'schen Optimierung angehörend - entwickelte sich im Wesentlichen im Kontext von s.g. Blackbox-Problemen. Derartige Probleme sind gekennzeichnet durch eine Zielfunktion, zu der keinerlei Informationen vorliegen und die nicht analytisch beschrieben werden können, wobei insbesondere Definitions-, sowie Wertebereich der Zielfunktion unbekannt sind. Entsprechend beschreibt Stöcker (2007, Kap. 2.4) ein Blackbox-Optimierungsproblem folgendermaßen: „Gegeben sind eine Menge S von Punkten, die als Variablen oder im technischen Kontext als Parameterwerte bzw. Entscheidungsvariablen bezeichnet werden sowie eine oder mehrere Zielfunktionen. Gesucht ist nun derjenige Punkt bzw. die Menge an Variablen, die nicht dominiert werden, die Paretomenge: $\mathcal{P} = \{\boldsymbol{x}^\star\} = \arg\min_{x \in S}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x}))^T$“ (Stöcker, 2007, S. 16).
Wie anhand der vorangegangenen Erklärungen zu Blackbox-Problemen deutlich wird, kann die (S)MBO als eine Art von Optimierungsverfahren insbesondere auch im Sinne des Hyperparameter-Tunings angewandt werden und entwickelt sich heutzutage sogar immer mehr zur Standardtechnik der Optimierung innerhalb des Benchmarkings. Da die SMBO eine doch recht komplexe Methodik darstellt, soll in dieser Arbeit auf eine tiefergehende Beschreibung verzichtet werden und hierfür auf Stöcker (2007) oder insbesondere Bischl et al. (2017) verwiesen werden. Allerdings lässt sich die grundlegende Funktionsweise - gestützt auf die zuvor genannten Quellen - hier nun kurz umreißen.
Man gehe hierzu von $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ als eine willkürliche Blackbox-Funktion mit d-dimensionalen Input-Definitionsbereich $\mathcal{X} = \mathcal{X}_1, \ldots, \mathcal{X}_d$ und einen deterministischen Output y aus. Gleichzeitig nehme man jedes $\mathcal{X}_i$ - wobei $i = 1, \ldots, d$ - entweder als numerisch und begrenzt (z.B. $\mathcal{X}_i = [l_i, u_i] \subset \mathbb{R}$) oder als finites Set von s kategorialen Werten ($X_i = \{v_{i1}, \ldots, v_{is}\}$) an. Ohne Einschränkung der Allgemeinheit wird schließlich derjenige Input $\boldsymbol{x}^\star$ gesucht, für den nachstehende Gl. 3.4 erfüllt ist:

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \tag{3.4}$$

Innerhalb der SMBO wird zumeist davon ausgegangen, dass f selbst teuer in der Evaluierung ist und entsprechend die Gesamtzahl der Funktionen-Evaluation durch ein gewisses Budget limitiert ist. Aus diesem Grund stellen s.g. surrogate Modelle $\hat{f}$, die auf „billige“ Weise die teure Blackbox-Funktion f schätzen sollen und hierbei iterativ aktualisiert, sowie verfeinert werden, die zentrale Komponente der SMBO dar. Die generelle Funktionsweise ist in Form von Abb. 3.3 dargeboten, kann aber schrittweise auch anhand der sich im Anhang befindenden Sektion 8.2.3 noch etwas genauer nachvollzogen werden.
Neben alledem kann die MBO beispielsweise auch auf multi-objektive Optimierung oder das s.g. „Multi-Point Proposal“ ausgeweitet werden und bietet den klaren Vorteil, dass Input-Variablen jeglichen Typs bzw. Parametertypen berücksichtigt werden können. Insbesondere sollte auch noch darauf hingewiesen werden, dass sich die MBO nicht nur für ein-, sondern



Quelle: Bischl et al. (2017)

Abb. 3.3: Genereller Ansatz der SMBO-Prozedur

auch multi-kriterielle Optimierung eignet. Letzteres bedeutet, dass nicht nur eine Funktion minimiert wird, sondern mehrere Ziele verfolgt werden und entsprechend nicht ein einzelner Punkt die Lösung darstellt, sondern die Paretomenge.

Weitere Optimierungsmethoden

Da das Hyperparameter-Tuning letztlich nichts anderes als ein Optimierungsproblem darstellt, stehen neben den drei bisher aufgeführten Methoden, eine Reihe weiterer Kontrollmöglichkeiten für das Tuning bereit. In diesem Zuge trifft Bardenet (2012, Kap. 3) die Unterscheidung zwischen Modell-freiem und -basiertem Tuning, wobei sich für einige Ansätze sowohl die Zugehörigkeit zu einer wie zur anderen Klasse argumentieren ließe. Erstere, automatische Hyperparameter-Tuning-Prozeduren können neben Grid und Random Search auch gängige Optimierungs-Heuristiken zugeteilt werden. Dazu zählen etwa das bereits lange genutzte „Hill-Climbing“ oder weiter entwickeltere Heuristiken wie der „Nelder-Mead“ Algorithmus, genauso wie „Estimation of Distribution“ Algorithmen (EDA's) oder „ad hoc genetic“ Algorithmen, welche naturanaloge Optimierungsverfahren bezeichnen.
Des Weiteren sind im Kontext der Modell-freien Optimierung auch „Racing“ Prozeduren aufzuführen. Ausgehend von einem Grid mit finitem Set von j Kandidaten an Hyperparameter-Vektoren startet ein Racing-Algorithmus parallel j Optimierungen, also eine pro Hyperparameter-Vektor. Dabei wird die folgende Eliminationsregel angewandt: reguläre, statistische Tests werden genutzt, um die Kandidaten zu vergleichen und diejenigen zu verwerfen, die einen niedrigen Rang aufweisen. Dies bedeutet, dass deren Evaluation stoppt, um damit die Konzentration auf Hyperparameter-Vektoren guter Performance durch Reduktion der Unsicherheit zu richten. Beispielsweise nutzt das s.g. F-Race einen Friedman-Test mit einer Nullhypothese gemäß der alle Kandidaten gleichermaßen gute Performance aufweisen. Falls diese Hypothese abgelehnt wird, finden Tests basierend auf dem paarweisen Vergleich der Mittelwerte statt, sodass jeder Vektor von Hyperparameter-Kombination, der schlechtere Performance als der aktuell „beste“ Kandidat leistet, eliminiert werden kann.
Entsprechend können im Rahmen der Modell-freien Optimierung also auch das F-Race oder

dessen Erweiterung, das iterierte F-Race, genutzt werden. Zusammenfassend sollen hierbei automatisch Optimierungsalgorithmen konfiguriert werden, was basierend auf statistischen Annahmen zur Auswahl der besten Konfiguration aus einem Set möglicher Konfigurationen unter stochastischer Evaluation geschieht. Ausführliche Informationen - insbesondere zur iterierten Version des F-Races - finden sich beispielsweise bei Lopez-Ibanez et al. (2011).
Hinsichtlich des Rechenaufwands verzeichnen v.a. Modell-freie Optimierungsmethode große Zeitansprüche. Daher versuchen Modell-basierte Optimierungsansätze einfach zu evaluierende Modelle an die objektive Funktion anzupassen und mithilfe des entsprechenden Modells das nächste zu testende Set von Hyperparametern bereitzustellen. Gemäß der Technik, mit welcher das Modell an die objektive Funktion angepasst wird, kann folgende Gruppierung vorgenommen werden: Lineare, sowie Neuronale-Netzwerk- und Gauß-Prozess-Regression oder Bagging von Regressionsbäumen. Hierbei kann sich SMBO sowohl auf die vorletzte, wie auch auf letzterer genannter Technik stützen. Für weiterführende Informationen sei auf Bardenet (2012) verwiesen.

Wenngleich wohl noch eine Reihe weiterer Optimierungsmethoden für das Hyperparameter-Tuning denkbar sind, so soll die Thematik an dieser Stelle nicht weiter von zentralem Interesse sein, sondern stattdessen der Frage nachgegangen werden, wie ein Algorithmus nun im engeren Sinne „lernt“ bzw. wie sich die Güte der Performance von ML-Methoden berechnen lässt. Zur Klärung dieser Frage bedarf es zunächst der eingehenderen Beschreibung von Gütemaßen, welche den Schwerpunkt der nachfolgenden Sektion darstellen.

3.4 Gütemaße zur Beurteilung der Performance von ML-Methoden

Eine der fundamentalen Aufgaben bei der Generierung von ML-Methoden ist wohl die Evaluierung von deren Performance. Zentral ist dabei die Frage, nach der Messbarkeit des Erfolgs derartiger Algorithmen angewandt auf die dem Anwender vorliegenden Daten. Damit beruht eine Seite der Evaluierung auf statistischer Signifikanz, sowie Konfidenzintervallen. Andererseits ist dies aber auch gebunden an die verwendete Metrik, anhand derer Aussagen über die Güte eines Modells getroffen werden. Umso wichtiger ist es ein grundlegendes Verständnis von den einzelnen Gütemaße zu bekommen und so deren Funktionsweise, sowie Vor- und Nachteile ersichtlich zu machen. Aus diesem Grund ist diese Sektion der Erklärung einiger wichtiger Evaluationsmetriken gewidmet. Die nun noch folgenden Erklärungen - vor Beschreiben der einzelnen Gütemaße - stützen sich auf Ferri et al. (2008).
Zumeist sind Evaluationsmetriken an die Art der Lernaufgabe gebunden. So gilt es hinsichtlich des SL's v.a. zwischen Klassifikations- und Regressionsmethodiken zu unterscheiden. Letztere werden in dieser Arbeit keine tragende Rolle spielen und entsprechend wird nun auch in dieser Sektion der Fokus auf Performancemaße für Klassifikatoren gelegt. In diesem Sinne sei hinsichtlich der Regression und deren Opportunität der Evaluationsmetriken beispielsweise

auf F. E. Harrell (2015) verwiesen und hervorhebend der hier beschriebene MSE und das (korrigiertes) Bestimmtheitsmaß als mögliche Maße in Betracht gezogen.
Nachdem es gerade für Klassifikationsaufgaben eine ganze Bandbreite möglicher Performancemaße gibt, nimmt Ferri et al. (2008, S. 27) eine Einteilung in drei Gruppen vor:

- Metriken, die auf einem Schwellenwert (Threshold) basieren und ein qualitatives Verständnis des Fehlers vermitteln, beispielsweise die Accuracy (ACC), der F-Score und die Kappa-Statistik. Diese Maße bieten sich v.a. dann an, wenn ein Modell die Anzahl fehlerhafter Klassenzuteilungen minimieren soll.

- Metriken, die eine Beurteilung hinsichtlich dessen erlauben, wie gut ein Modell Beispieldaten einordnen kann (Separierbarkeit). In diesem Kontext lassen sich ROC-Analysen nennen, wobei insbesondere das AUC aufzuführen ist, welches im Fall von binärer Klassifikation äquivalent ist zur Mann-Whitney-Wilcoxon-Statistik und damit eng verbunden ist mit dem Konzept der Separierbarkeit. Für genaue Erklärungen bezüglich dieses Zusammenhangs sei beispielsweise auf Scheipl et al. (2018, Kap. 8) verwiesen.

- Metriken, die auf dem probabilistischen Verständnis des Fehlers beruhen und so z.B. die Abweichung von der wahren Wahrscheinlichkeit messen: mittlerer absoluter Fehler (MAE), mittlerer quadratischer Fehler (MSE), logarithmierter Verlust (logLoss), Maße zur Kalibrierung, etc. Entsprechend bieten sich diese Maße insbesondere dann an, wenn man die Reliabilität eines Klassifikators bestimmen will und damit auch zugänglich machen möchte, - nicht nur, ob eine fehlerhafte Einteilung erfolgt, sondern - ob eine falsche Klasse mit hoher oder niedriger Wahrscheinlichkeit gewählt wird.

Nachfolgend soll allerdings auch ein Maß vorgestellt werden, welches sich zwischen den letzten zwei genannten Metrik-Typen einordnen lässt, das s.g. partielle AUC (pAUC).
Die Art der vorgenommenen Typus-Einteilung der Evaluationsmetriken, die zuvor dargeboten wurde, lässt sich wohl am einfachsten folgendermaßen begründen: ein guter Klassifikator im Sinne der Separierbarkeit kann sehr schlechte Ergebnisse im Sinne der Genauigkeit aufweisen, falls ein ungeeigneter Threshold gewählt wurde. Andersherum kann eine Methode sehr gute Ergebnisse in der Klasseneinteilung für einen spezifischen Schwellenwert aufweisen, wohingegen diese bei Betrachtung eines anderen, ungeeigneten Thresholds beispielsweise durch Änderung des Kostenkontextes schlechte Performance leistet. Ausweitend ist wohl insbesondere der Unterschied zwischen guten Ranking-Leistungen und hohen Wahrscheinlichkeiten schwer zu erfassen: Ein Klassifikator kann durchaus viele korrekte Einordnung treffen, allerdings können sich ggf. die zugrundeliegenden Wahrscheinlichkeiten stark von den tatsächlichen Wahrscheinlichkeiten unterscheiden. In diesem Fall würde man dann von einer schlecht kalibrierenden Klassifikationsmethode sprechen.
Um das Verständnis für die Beurteilung der Performance einer ML Methode zu vertiefen, werden nachfolgend die wichtigsten Maße im Kontext der Klassifikation, jedoch auch einige für

den Fall von Regressionsaufgaben bereitgestellt. Ferri et al. (2008, S. 28) betonen in diesem Zusammenhang, dass wenngleich einige der Metriken starke Korrelationen untereinander aufweisen, zumeist unterschiedliche Größen im Fokus der Berechnung stehen und sich damit auch die Entscheidung verschiedener Metriken unterscheiden könne. Diese Differenzen vergrößern sich insbesondere für Mehrklassen-Probleme, Klassifikationsaufgaben mit stark imbalancierter Klassenverteilung, ebenso wie bei der Problematik einer geringen Datenbasis.

3.4.1 Accuracy (ACC) und Missklassifikationsfehler (MMCE)

Accuracy (ACC)

Eine wohl recht naheliegende Variante der Güte-Beurteilung stelle das Genauigkeitsmaß Accuracy (ACC) dar. Unter Nutzung der k-fachen Kreuzvalidierung wird innerhalb jeder Iteration das jeweilige Modell auf die Trainingsdaten angepasst und die Prädiktion auf Basis der Testdaten durchgeführt. Letzteres dient der Beurteilung der Güte, indem die Anzahl der korrekt vorhergesagter Ausprägungen des Outcomes im Verhältnis zu den insgesamt zu treffenden Klassifikationen gesetzt wird. Damit beschreibt das ACC den relativen Anteil korrekter Prädiktionen und lässt sich informell anhand des nachstehenden Quotienten in Gl. 3.5 beschreiben. Zusätzlich kann auch auf die s.g. Konfusionsmatrix Bezug genommen werden, wie diese in Tab. 3.1 dargestellt ist. Sollte es an Kenntnis über die Einträge der Konfusionsmatrix mangeln, so sei für deren Klärung auf die anschließende Untersektion 3.4.2 verwiesen und bis dahin die verwendete Notation in Gl. 3.5 hingenommen. Des Weiteren sei vermerkt, dass Ω hier die Gesamtzahl der insgesamt getroffenen Prädiktionen kennzeichnet.

$$ACC = \frac{\#korrekte\ Prädiktionen}{\#zu\ treffende\ Prädiktionen} = \frac{\#\mathrm{TP} + \#\mathrm{TN}}{\#\mathrm{TP} + \#\mathrm{FP} + \#\mathrm{FN} + \#\mathrm{TN}} = \frac{\#\mathrm{TP} + \#\mathrm{TN}}{\Omega} \quad (3.5)$$

Demgemäß spiegelt sich die optimale Performance eines Modells in einem ACC von 100 % wider und damit die gänzlich korrekte Klassifikation des Outcomes aller Testdaten. Entsprechend würde sich ein derartiges Modell sehr gut eignen, um Muster innerhalb der Trainingsdaten zu erkennen. Dagegen würde ein ACC von 50 % implizieren, dass für die Hälfte aller Testfälle eine richtige/falsche Klassifikation erfolgt, was ein sehr unbefriedigendes Ergebnis ist und rein zufälliger Klassenzuweisung gleichkommt.

Missklassifikationsfehler (MMCE)

Ein weiteres Gütemaß, der s.g. mittlere Klassifikatiosnfehler (Mean Missclassification Error = MMCE), lässt sich als Gegenstück zur Accuracy beschreiben. Folglich ist das MMCE als relativer Anteil falscher Zuordnungen an der Gesamtzahl zu treffender Klassifikationen definiert:

$$MMCE = \frac{\#\text{falsche Prädiktionen}}{\#\text{zu treffende Prädiktionen}} = \frac{\#\mathrm{FP} + \#\mathrm{FN}}{\Omega} = 1 - ACC \quad (3.6)$$

Zusammenfassend können die Gütemaße ACC und MMCE wohl als recht triviale und intuitive Werkzeuge zur Beurteilung bzw. dem Vergleich der Performance von Klassifikatoren ausgemacht werden. Jedoch bleiben hierbei einige Faktoren unberücksichtigt, wobei nämlich insbesondere zu bedenken gilt, dass nicht einzig die auf die Anzahl korrekter Zuweisungen geachtet werden sollte. Man nehme beispielsweise an, es lägen Daten zum Screening einer relativ seltenen Krankheit vor, die mit einer Prävalenz - gegeben durch $(\#TP + \#FN)/\Omega$ - von 5 % auftritt. Deklariert ein Klassifikator alle Patienten als gesund, so weist diese Methodik ein ACC von 95 % auf. Die Beurteilung anhand des Genauigkeitsmaßes ACC würde somit auf einen recht guten Klassifikator schließen lassen, doch in Wirklichkeit wird bei keinem Patienten die Krankheit entdeckt. Insbesondere ist das ACC - sowie entsprechend das MMCE - also empfindlich gegenüber schiefen Klassenverteilungen. Zudem wird von diesen Maßen gleiche Fehlerkosten für positive und negative Klassenzuweisungen angenommen, was in der Realität selten der Fall ist. Dies impliziert das Heranziehen weiterer Gütemaße für eine ausweitende Beurteilung der Performance.

3.4.2 ROC-Analysen und Fläche unterhalb der ROC-Kurve (AUC)

Ein weiteres Gütemaß für die Bewertung der Performance, sowie für den Vergleich unterschiedlicher Klassifikationsmethoden ist durch die Fläche unterhalb der ROC-Kurve (Area under the Curve = AUC) gegeben. Um dieses eingehender beschreiben zu können, bedarf es zunächst einiger Verdeutlichungen zur „Receiver Operating Characteristics Curve“ - oder kurz ROC-Kurve -, welche sich nachfolgend an Scheipl et al. (2018, Kap. 8) und Lohninger (2012, Kap. *ROC-Kurve* und Kap. *Gütemaße für Klassifikatoren*), sowie Hallinan (2014) orientiert.

ROC-Kurve

Die ROC-Kurve ergibt sich durch die Abtragung der Richtig-Positiv-Rate (TPR) gegen die Falsch-Positiv-Rate (FPR) für unterschiedliche Schwellenwerte (Thresholds). Dazu zunächst folgende Erklärung: hinsichtlich der eigentlichen, aber unbekannten Realität kann das Outcome eines Klassifikators entweder richtig oder falsch sein. Somit ergeben sich vier Kombinationsmöglichkeiten aus tatsächlicher und prognostizierter Klassenzugehörigkeit. Durch Abtragung dessen in Form einer Art Tabelle - wie diese in Tab. 3.1 zu sehen ist - entsteht die s.g. Konfusionsmatrix. Ihr kann generell entnommen werden, wie gut ein Klassifikator Zuordnungen treffen kann, wobei eine Zuweisung also richtig ist, wenn sie als „richtig-positiv“ oder „richtig-negativ“ deklariert ist. Folglich kann die Zahl der korrekten Klassifikationen an der Hauptdiagonalen abgelesen werden. Ein Großteil der aus der Konfusionsmatrix ableitbaren Performance-Maße finden sich im Anhang 8.2.2, sodass im weiteren Verlauf dieses Kapitels nur ein Teil der zur Verfügung stehenden Maße ihre Definition finden. Zudem finden sich im Anhang unter 8.2.4 ausweitende Erklärungen zur Thematik probabilistischer Klassifikatoren, sowie zum Threshold θ.

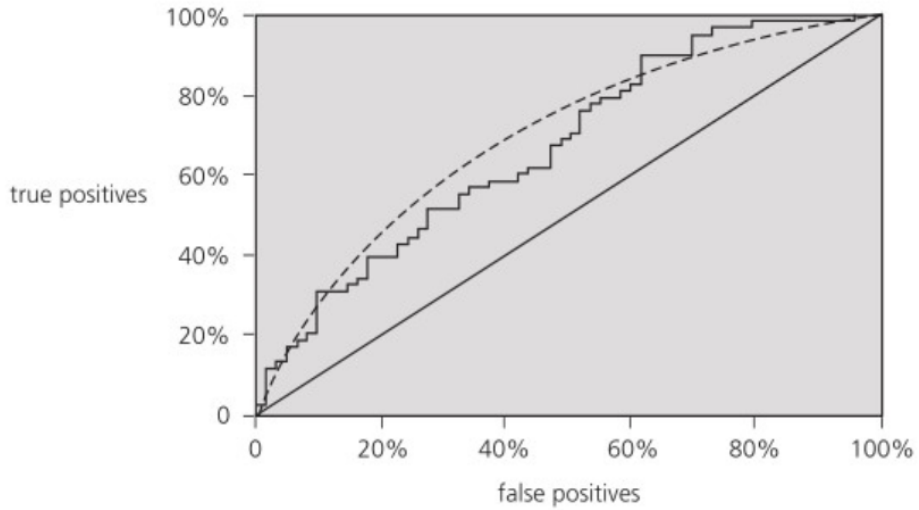
	Realität positiv (T+)	Realität negativ (T-)	*Summe*
Prädiktion positiv (P+)	richtig-positiv (#TP)	falsch-positiv (#FP)	PP = #TP + #FP
Prädiktion negativ (P-)	falsch-negativ (#FN)	richtig-negativ (#TN)	PN = #FN + #TN
Summe	RP = #TP + #FN	RN= #FP + #TN	Ω

Tab. 3.1: Konfusionsmatrix

Nun gilt es die Frage zu klären, wie sich mittels Tab. 3.1 die TPR, sowie die FPR errechnen lassen. Dieser ergeben sich jeweils als Anteil der richtig-positiv bzw. falsch-positiv Entscheidungen an der Gesamtzahl der tatsächlich positiven bzw. negativen Entscheidungen/Klassen. Die TPR ist teilweise wohl besser bekannt als Sensitivität/Empfindlichkeit und die FPR stellt das Gegenteil zur s.g. Spezifität dar, wobei somit jeweils gilt:

$$\text{TP-Rate (TPR; Sensitivität)} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}}$$
$$\text{FP-Rate (FPR; 1-Spezifität)} = \frac{\#\text{FP}}{\#\text{TN} + \#\text{FP}}$$

Der ROC-Raum lässt sich nun also über die FPR und TPR - jeweils als x- bzw. y-Achse - in Abhängigkeit des jeweiligen Schwellenwertes θ aufspannen und bildet damit den Trade-Off zwischen Nutzen (richtig-positiv) und Kosten (falsch-positiv) ab. Im Falle von Analysen endlich vieler Beobachtungen, stellt sich die ROC-Kurve in Form einer Stufenfunktion dar. Entsprechend nähert man sich mit einer gegen unendlich strebenden Zahl an Beobachtungen der eigentlich namensgebenden ROC-Kurve an. Für eingehendere Beschreibung zur Konstruktion der ROC-Kurve sei auf Anhang 8.2.4 verwiesen und für eine visuelle Darstellung der besagten Kurve Abb. 3.4 herangezogen. Grob formuliert verursacht hier das Herabsetzen des Schwellenwertes eine Bewegung von der „konservativen“ zur „liberalen“ Seite des Graphen. Ersteres meint, dass positive Einordnungen nur unter deutlichen dafür sprechenden Anzeichen getroffen werden, sodass wenige falsch-positiv Fehler auftreten, allerdings häufig auch eine niedrige richtig-positiv Rate (Sensitivität) die resultierende Konsequenz ist. Umgekehrt bezeichnet der „liberale“ Gedanke, positive Prädiktionen bereits unter geringer Annahme dazu, sodass fast alle positiven Klassen als solche erkannt werden, jedoch eine hohe falsch-positiv Rate damit einhergeht. [vgl. Fawcett (2005, S. 863)]
Generell sei vermerkt, dass eine Erhöhung der Sensitivität oder der Spezifität umgekehrt proportional zueinander sind, wodurch mit der Erhöhung des einen Maßes eine Verringerung des anderen impliziert wird. Dies lässt sich auch mathematisch beweisen, soll hier aber nicht weiter ausgeführt werden. Mit diesem Zusammenhang gilt allerdings auch, dass die Erhöhung der TPR eine Steigerung der FPR mit sich führt. Gleichzeitig ergibt sich, dass mit geringer werdendem Schwellenwert die Zahl der positiven bzw. mit höherem Schwellenwert die Anzahl der negativen Klassenzuweisungen steigt.



Quelle: (Witten et al., 2011, Kap. 5.7)

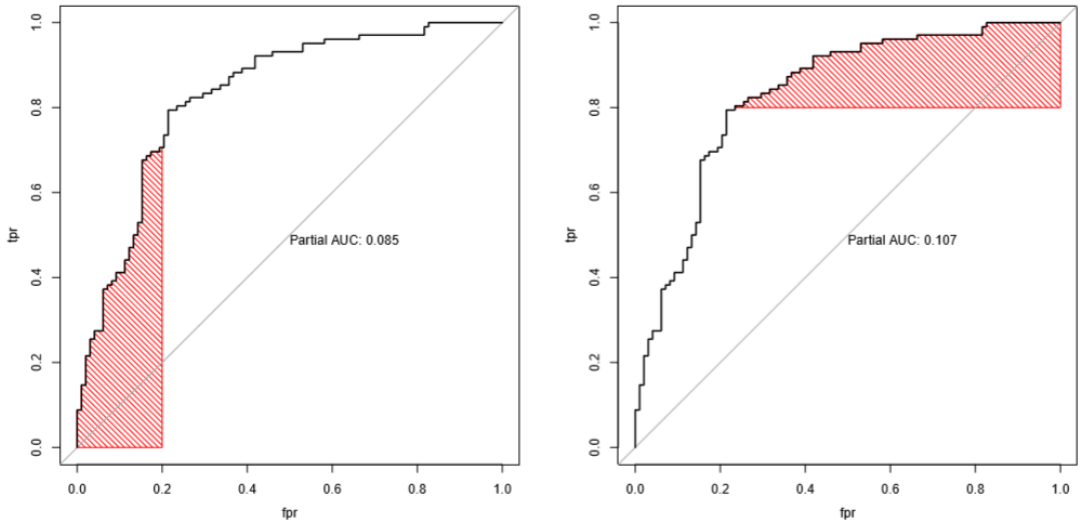
Abb. 3.4: Beispielhafte Veranschaulichung zur ROC-Kurve: Entstehung der Stufenfunktion durch Verschiebung des Thresholds θ

Bisher ist allerdings die Frage, welcher Threshold gewählt werden sollte, um eine möglichst „optimale“ Klassifikation der Daten zu erreichen, nicht eindeutig geklärt. Angestrebt wird i.A. natürliche ein Threshold, welcher mit möglichst vielen richtig-positiv und möglichst wenigen falsch-positiv Zuteilungen einhergeht. Doch allgemein gesprochen kann ein derartiges θ nicht direkt bestimmt werden, sondern hängt von der Problemstellung an sich ab und verlangt eine individuelle, an die jeweilige Situation angepasste Festsetzung dieses Schwellenwertes. Beispielsweise nimmt man wohl auch bei der Erkennung potentieller Krebspatienten eine recht hohe falsch-positiv Rate bei Screening-Untersuchungen in Kauf, um letztlich mit hoher Sicherheit alle möglicherweise Erkrankten zu erkennen. Zur ausweitenden Klärung der angesprochenen Thematik kann v.a. die Untersektion 8.2.4 des Anhangs genutzt werden, in der s.g Iso-Accuracy-Linien (IAL's) herangezogen werden, um den Zusammenhang zwischen dem ACC und der TPR bzw. FPR auszumachen.
Insbesondere scheint sich die zuvor genannte Problematik noch zu verstärken in Anbetracht schiefer Verteilungen der Outcome-Variable. Doch hier tritt nun ein großer Vorteil von ROC-Graphen in Erscheinung: diese ermöglichen die Visualisierung und den Zugang zur Performance von Klassifikatoren ohne Rücksichtnahme auf die Klassenverteilung, sowie Fehler-Kosten. Die ROC-Kurve ist folglich invariant im Hinblick auf diese Einsatzbedingungen, der Klassenschiefe und Fehler-Kosten. Mit Änderung dieser Bedingungen fallen ggf. andere Ausschnitte in den Interessensbereich, doch der ROC-Graph an sich bleibt derselbe. Dies lässt sich dadurch begründen, dass die Kurve von TPR und FPR bestimmt wird, wobei jede Dimension einem strikten spaltenorientierten Verhältnis entspricht und damit nicht von der Klassenverteilung abhängt.

Die bisherigen Beurteilungsmöglichkeiten der Performance basierend auf der ROC-Kurve - einschließlich der im Anhang 8.2.4 präsentierten „konvexen Hülle“ (ROCCH) - beschränken sich auf den zweidimensionalen Raum. Insbesondere zum Vergleich von Klassifikatoren kann es aber auch sinnvoll sein, eine Einstufung anhand eines skalaren Wertes durchzuführen, zumal es häufig auch schwierig ist graphisch - anhand der ROCCH - die Dominanz eines Klassifikators gegenüber einem anderen auszumachen und daher nun auf das s.g. AUC übergeleitet werden.

AUC und pAUC

Ein Gütemaß zur Beurteilung der Qualität eines betrachteten Klassifikators lässt sich nun direkt aus der ROC-Kurve ableiten, ebenso wie dies ein Mittel darstellt, um verschiedene Methodiken miteinander zu vergleichen. Insbesondere ist dieses Maß aber auch unabhängig von der Wahl eines spezifischen Thresholds. Die Rede ist von der Fläche unterhalb der ROC-Kurve - zu Englisch „Area Under the (ROC-)Curve“ - welche abgekürzt als AUC bezeichnet wird. Damit ergibt sich das AUC als Integral der ROC-Funktion im Intervall $[0, 1]$. Folglich kann dieses Gütemaß Werte zwischen 0 und 1 annehmen ($AUC \in [0, 1]$), wobei generell gilt, dass ein Klassifikator eine umso höhere Performance aufweist, je höher das zugehörige AUC ist. Entsprechend ist ein perfekter Klassifikator gekennzeichnet durch $AUC = 1$, wohingegen zufälliges Raten die Winkelhalbierende im ROC-Raum hervorbringt und damit $AUC = 0.5$ ausmacht, wobei man dann vom nicht-diskriminanten AUC spricht. Schlechte Klassifikatoren sind entsprechend solche, die ein $AUC \in [0, 0.5)$ aufweisen.
Wie deutlich werden sollte, kann das AUC auch als Wahrscheinlichkeit interpretiert werden. Die ROC-Kurve trägt die Wahrscheinlichkeit einer positiven Prädiktion in den Teilpopulationen der tatsächlich positiven und den in Wirklichkeit negativen Beobachtungen gegeneinander ab. Somit stellt das AUC die Wahrscheinlichkeit dar, dass bei einem zufällig ausgewählten Paar, bestehend aus einer wahren positiven und einer wahren negativen Beobachtung, die korrekte Klassenzuweisung in positiv und negativ gelingt. Damit kann die direkte Verbindung zur Mann-Whitney-Wilcoxon-Statistik hergestellt werden. [vgl. Wehberg et al. (2007, S. 335)]
Es gibt jedoch auch Fälle, in denen es sinnvoll ist nicht das gesamte AUC zu betrachten, sondern nur einen bestimmten Teilbereich unterhalb der ROC-Kurve. In dieser Situation spricht man dann vom partiellen AUC (pAUC), dessen Anwendung zum Beispiel dann zum Tragen kommt, wenn die FPR von größerer Bedeutung ist als die TPR oder umgekehrt. Zur Definition dieser Region bedarf es zweier Komponenten c_1, c_2, welche die Koordinaten/Abschnitte von TPR und FPR wiedergeben sollen, wobei gilt $0 \leq c_1 < c_2 \leq 1$. So kann dann - wie in Abb. 3.5 beispielhaft dargestellt - der Fokus auf Bereiche mit niedriger FPR ($c_1 = 0, c_2 = 0.2$) oder hoher TPR ($c_1 = 0.8, c_2 = 1$) gelegt werden. Damit ist schließlich das pAUC auf einen Definitionsbereich von $pAUC \in [0, c_2 - c_1]$ begrenzt. Weil dies Schwierigkeiten in der Interpretation mit sich bringt, kann es sinnvoll sein, das pAUC entsprechend zu korrigieren, sodass der Definitionsbereich wiederum auf dem Intervall $pAUC_{korr} \in [0, 1]$ angesiedelt ist und somit



Quelle: (Scheipl et al., 2018, Kap. 8, S.49)

Abb. 3.5: Beispielhafte Veranschaulichung zum pAUC: Fokussierung von links: niedriger FPR ($c_1 = 0, c_2 = 0.2$); rechts: hoher TPR ($c_1 = 0.8, c_2 = 1$)

dann 0.5 den Wert des nicht-diskriminanten AUC's darstellt und 1 den maximalen. Dies kann gemäß McClish (1989) folgendermaßen bewerkstelligt werden:

$$pAUC_{korr} = \frac{1 + \frac{pAUC - min}{max - min}}{2} \tag{3.7}$$

mit pAUC = *Unkorrigiertes partielles AUC*

min, max = *Wert des nicht-diskriminaten bzw. maximal möglichen AUC's in Region*

Abschließend sollte noch einmal kurz festgehalten werden, dass sich die dargestellten ROC-Analysen bzw. Bewertungen anhand des (p)AUC's sowohl für die Bestimmung eines Thresholds θ heranziehen lassen, als auch für die Beurteilung der Performance eines Klassifikators. Ebenso können mit diesem Werkzeug zwei oder mehrere Klassifikations-Methoden einander gegenüber gestellt werden, indem deren Leistung für einen spezifischen Threshold und damit punktweise oder über die globale Performance und entsprechend mittels des AUC's - bzw. beschränkt auf einen bestimmten Teilbereich über das pAUC - beurteilt wird.
Es sei zusätzlich noch kurz auf den Zusammenhang des AUC's mit dem s.g. Gini-Koeffizient hingewiesen, wobei sich diese Beschreibung an Rezac und Rezac (2011, S. 490 ff.). Dieser lässt sich eigentlich in den Kontext von Analysen mittels der s.g. Lorenz-Kurve (oder Power-Kurve) einordnen, welche eine (graphische) Methode zur Bestimmung der Trennschärfe darstellt und insbesondere als Maß der Disparität in der Ökonomie verbreitet ist. Wenngleich die ROC- und Lorenz-Kurve nicht miteinander verwechselt werden sollten, können doch gewisse charakteristische Kennzahlen dieser beiden Maße ineinander übergeführt und so auch der

Zusammenhang zwischen dem Gini-Koeffizienten und dem AUC hergestellt werden. Dabei errechnet sich ersterer als Konzentration bzw. Fläche zwischen der Winkelhalbierenden und der Lorenzkurve. Überleitend kann dann der Gini-Koeffizient G in Abhängigkeit vom AUC - wie in Gl. 3.8 dargestellt - errechnet werden. Demzufolge können also Werte auf dem Intervall $[0, 1]$ angenommen werden.

$$G = 2 \cdot AUC - 1 \tag{3.8}$$

3.4.3 Mittlerer quadratischer Fehler (MSE) bzw. Brier-Score

Nachfolgend sollen nun auch Gütemaße für die Beurteilung der Performance von Lernern, die sich hinsichtlich der Prädiktion kontinuierlicher Outcomes eignen, beleuchtet und mithilfe von Swalin (2018) und Jordan (2017) zugänglich gemacht werden. Die Evaluation von Metriken im Regressionsfall verlangt ein anderes Vorgehen als bisher für Klassifikationen - insbesondere des binären Typus - gezeigt wurden. In diesem Kontext das wohl bekannteste Maß ist der s.g. mittlere quadratische Fehler (MSE), auch Brier-Score genannt. Dieser stellt den Mittelwert der quadrierten Abweichungen bzw. Differenzen zwischen prognostiziertem und wahrem Wert der betrachteten Zielvariable dar. Häufig wird stattdessen auch die Stichproben-Standardabweichung der Differenz zwischen geschätztem und eigentlichem Outcome betrachtet und damit die Wurzel aus dem MSE. Aus diesem Grund spricht man für letzteres von dem s.g. RMSE, was auf die englische Bezeichnung „Root Mean Squared Error“ zurückzuführen ist.

$$MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 \quad \text{und} \quad RMSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2} \tag{3.9}$$

3.4.4 (Korrigiertes) Bestimmtheitsmaß R^2

Diese Untersektion widmet sich dem s.g. Bestimmtheitsmaß R^2, wobei hier für eingehende Beschreibungen - neben den bereits genannten Quellen - insbesondere auch auf Fahrmeir et al. (2009, S. 89 ff.) zurückgegriffen wird. Diese Metrik stellt eine Möglichkeit dar die Anpassungsgüte von Regressionsmodellen zu beurteilen, indem ermittelt wird, wie gut die ausgewählten, unabhängigen Variablen die Variabilität innerhalb der abhängigen Variable beschreiben. Entsprechend beruht das Gütemaß auf der Quadratsummenzerlegung - also der Aufteilung in erklärte und nicht-erklärte Quadratsumme - und beschreibt welcher Anteil der Streuung innerhalb der Daten durch das verwendete Regressionsmodell erklärt werden kann. Folglich lässt sich die Bezeichnung mit R^2 ableiten von der Tatsache, dass das Bestimmtheitsmaß dem quadrierten Korrelationskoeffizienten entspricht. So ergibt sich R^2 schließlich als Anteil erklärter Variation an der Gesamtstreuung bzw. andersherum dargestellt als 1 abzüglich der Anteil nicht-erklärter Variation (SQR: „Sum of Squared Residuals“) an der Gesamtstreuung (SQT: „Sum of Squared Total deviation“), was durch Gl. 3.10 zum Ausdruck kommt.

$$R^2 = 1 - \frac{SQR}{SQT} = 1 - \frac{\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}{\frac{1}{N}\sum_{n=1}^{N}(y_n - \bar{y}_n)^2} = 1 - \frac{\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}{\sum_{n=1}^{N}(y_n - \bar{y}_n)^2} \tag{3.10}$$

Anhand der vorhergehenden Gleichung sollte ersichtlich werden, dass im Zähler des Quotienten der MSE bzw. Mittelwert der quadrierten Residuen (s. Gl. 3.9) bezeichnet wird, während der Nenner die Varianz der y-Werte umfasst. Schlussfolgernd impliziert ein hoher MSE ein niedriges R^2 und entsprechend ein schlecht an die Daten angepasstes Regressionsmodell. Angestrebt wird also ein möglichst kleiner MSE und damit ein hoher Wert des Bestimmtheitsmaßes. Dabei liegt der Wertebereich dieses Gütemaßes also im Intervall $[0, 1]$, wobei sich 1 ergibt für den Fall, dass $MSE = 0$ und damit alle Datenpunkte des Streudiagramms von der Regressionskurve durchlaufen werden. Andersherum impliziert $R^2 = 0$, dass das Modell keinerlei Beitrag zur Erklärung der Gesamtstreuung leisten kann.
Da das Bestimmtheitsmaß mit der Aufnahme weiterer möglicher Einflussvariablen - unabhängig davon, ob diese tatsächlich einen Beitrag zur Erklärung der Varianz verzeichnen können oder nicht - wächst und damit die Gefahr der Überanpassung (Overfitting) zunimmt, wird zumeist das s.g. korrigierte bzw. adjustierte Bestimmtheitsmaß für eine Beurteilung der Performance herangezogen. Dieses Gütemaß beinhaltet einen Penalisierungsterm, der die Aufnahme zusätzlicher Variablen bestraft. Anstelle der Quadratsummen werden hier die mittleren Abweichungsquadrate genutzt und damit die jeweiligen Freiheitsgrade berücksichtigt. Entsprechend ergeben sich diese als $MQT = SQT/(N-1)$ und $MQR = SQR/(N-k)$, womit sich schließlich Gl. 3.11 für das korrigierte Bestimmtheitsmaß ausmachen lässt. Hierbei bezeichnet k die Anzahl der im Modell inbegriffenen Regressoren bzw. die berücksichtigten Variablen mitsamt einer Konstanten (Intercept). [vgl. Timischl (2013, S. 338)]

$$R^2_{adj} = 1 - \frac{MQR}{MQT} = 1 - \frac{N-1}{N-k} \cdot \frac{SQR}{SQT} = 1 - (1 - R^2) \cdot \frac{N-1}{N-k} \tag{3.11}$$

mit k = Anzahl der Regressoren im betrachteten Modell (einschließlich Intercept)

Demzufolge sollte eine Variable genau dann in das Modell aufgenommen werden, wenn der gewonnene Erklärungswert größer dem Abschlag durch die Bestrafung ist und damit das ermittelte R^2_{adj} gegenüber demjenigen vor Aufnahme dieser Einflussgröße steigt. Das korrigierte Bestimmtheitsmaß findet heutzutage vielfache Anwendung und ist aus dem zuvor genannten Grund gegenüber dem Bestimmtheitsmaß ohne Penalisierung zu bevorzugen. Allerdings führt Fahrmeir et al. (2009, S. 161) auf, dass auch das korrigierte Maß eine noch zu geringe Bestrafung neu aufgenommener Kovariablen mit sich führt, nachdem R^2_{adj} bereits dann steigt, wenn eine Variable mit einem t-Wert betragsmäßig größer als 1, in das Regressionsmodell einbezogen wird. So wird argumentiert, dass bereits Einflussgrößen mit einem p-Wert von ca. 0.3 in das Modell aufgenommen würden. Daher raten Fahrmeir et al. (2009) z.B. die Verwendung von Mallows' C_p an, in dessen Richtung diese Arbeit nun aber nicht weiter vertieft wird.

4 Ausgewählte ML-Methoden

Bisher wurde lediglich von ML-Methodiken gesprochen, doch gab es noch keinerlei Erklärungen zu einer derartigen Methode. Dieses Kapitel beinhaltet daher nun Beschreibungen ausgewählter Methoden, die größtenteils auch innerhalb der durchgeführten Benchmark-Studie und damit in Sektion 6.2 ihre Anwendung finden. In diesem Rahmen folgen nun: k-Nearest-Neighbors (kNN), Support Vector Machines (SVMs), Entscheidungsbäume mit besonderem Fokus auf Bäume konditionaler Inferenz (cTrees), sowie die auf diesen aufbauenden Random Forests, welche als cForest bezeichnet werden, ebenso wie GLM's, die sich auf Boosting Prozeduren stützen oder auf penalisierter Maximum Likelihood Schätzung beruhen.

4.1 k-Nearest Neighbors (kNN)

Die nicht-parametrische Methodik k-Nearest Neighbors (kNN) stellt eine recht intuitive Möglichkeit dar, um sowohl Klassifikations-, als auch Regressionsaufgaben zu bewerkstelligen. Dies impliziert, dass die Funktionsweise dieses Algorithmus derart konstruiert ist, dass keinerlei Annahmen über die funktionale Form des zu lösenden Problems getroffen werden müssen.
Die folgende, zugehörige Sektion soll entsprechende Erklärungen und Konzepte zu dem kNN-Algorithmus vermitteln, wobei dies unter Zuhilfenahme von Brownlee (2016) und Scheipl et al. (2018, Kap. 2) geschieht. Wie anhand der Namensgebung bereits ersichtlich wird, beruht das Grundkonzept hierbei im Durchsuchen des Trainingsdatensatzes nach den k ähnlichsten Instanzen - welche aufgrund der Ähnlichkeit als „Nachbarn" bezeichnet werden - zu einer neuen Instanz $\boldsymbol{x}_\bullet$, sodass eine anschließende Prädiktion von $\hat{y}_\bullet$ in Form einer Zusammenfassung der Output-Variablen dieser k nächsten Nachbarn ermöglicht wird. Diese Zusammenfassung kann in Regressionsfällen beispielsweise der Mittelwert oder Median aus den k Outcome-Werten sein oder bei Klassifikationen einen Klassenzuweisung basierend auf der am häufigsten auftretenden Klasse innerhalb der k Nachbarn.

Distanzmaße

Um diejenigen k Instanzen bestimmen zu können, die die größten Übereinstimmungen mit einer neuen Beobachtung aufweisen, benötigt man ein Ähnlichkeits- bzw. Distanzmaß. Als wohl bekanntestes Maß - für reellwertige Input-Variablen - ist hier die euklidische Distanz anzuführen. Diese errechnet sich als Quadratwurzel aus der Summe der quadrierten Differenzen zwischen einem neuen Datenpunkt $\boldsymbol{x}_\bullet$ und einer bereits existierenden Beobachtung $\boldsymbol{x}_i$ über alle Input-Attribute p. Anhand dieser Berechnung werden schließlich diejenigen k Datenpunkte als $\boldsymbol{x}_\bullet$ am nächsten gewählt, die die insgesamt k geringsten Distanzen aufweisen.

$$d_{euklidisch}(\boldsymbol{x}_\bullet, \boldsymbol{x}_i) = \sqrt{\sum_p (x_{\cdot p} - x_{ip})^2} \tag{4.1}$$

Einige weitere bekannte Distanzmaße sind die nachfolgend gelistet, wobei die mathematische Formulierung dieser Distanzmaße von Sayad (o. J.) abgeleitet ist. Theoretisch gesehen können zahlreiche weitere Maße - etwa die Mahalanobis- oder Jaccard-Distanz - in Betracht gezogen werden können. Die Wahl des letztlich verwendeten Maßes sollte in Abhängigkeit von den Eigenschaften der beobachteten Daten getroffen werden. In experimenteller Form können natürlich verschiedene Distanzmaße verglichen werden und unterschiedliche Werte für k einander gegenüber gestellt werden, um das geeignetste Modell zu bestimmen. Dies geschieht wiederum im Rahmen des Hyperparameter-Tunings, welches sich in Sektion 3.3.2 vertieft findet.

- Hamming-Distanz: Distanz zwischen zwei binären Vektoren (also $x_{\bullet p}, x_{ip} \in \{0, 1\}$) $d_{hamming}(\boldsymbol{x}_\bullet, \boldsymbol{x}_i) = \sum_p |x_{\bullet p} - x_{ip}|$, wobei $d = 0$, falls $x_{\bullet p} = x_{ip}$ und $d = 1$ sonst;
- Manhattan- bzw. City-Block-Distanz: Distanz zwischen zwei reellwertigen Vektoren mit Berechnung der Summe von deren absoluten Differenz $d_{manhattan}(\boldsymbol{x}_\bullet, \boldsymbol{x}_i) = \sum_p |x_{\bullet p} - x_{ip}|$;
- Minkowski-Distanz: Generalisierung von euklidischer und Manhattan-Distanz $d_{minkowski}(\boldsymbol{x}_\bullet, \boldsymbol{x}_i) = \left[\sum_p (|x_{\bullet p} - x_{ip}|)^q\right]^{1/q}$.

Es sei angemerkt, dass die euklidische Distanz sich insbesondere anbietet, wenn die Input-Variablen ähnlichen Types sind, also beispielsweise Längen- und Höhenmessungen, und die Manhattan-Distanz bei unterschiedlichen Typen, wie etwa Variablen zu Alter, Geschlecht, Größe, etc.

Nachbarschaft und Prädiktionen

So kann schließlich die k-Nachbarschaft von $\boldsymbol{x}_\bullet$ - bezeichne dies als $N_k(\boldsymbol{x}_\bullet)$ - definiert werden als dasjenige Set der k nähesten Punkte zu x und mathematisch wie in Gl. 4.2 formuliert werden.

$$N_k(\boldsymbol{x}_\bullet) = \{(\boldsymbol{x}_i, y_i) : |[l : d(\boldsymbol{x}_\bullet, \boldsymbol{x}_l) \leq d(\boldsymbol{x}_\bullet, \boldsymbol{x}_i)]| \leq k\} \tag{4.2}$$

Aus Basis der vorangehenden Beschreibung können schließlich auch Prädiktionen erfolgen. Im Falle des Regressionskontextes geschieht dies zumeist über das arithmetische Mittel. Damit kann die Prädiktion des Outcomes für einen neuen Datenpunkt $\boldsymbol{x}_\bullet$ in Abhängigkeit von den k nächsten Nachbarn dann - wie in Gl. 4.3 dargestellt - durchgeführt werden.

$$\hat{y}_\bullet = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x}_\bullet)} y_i \tag{4.3}$$

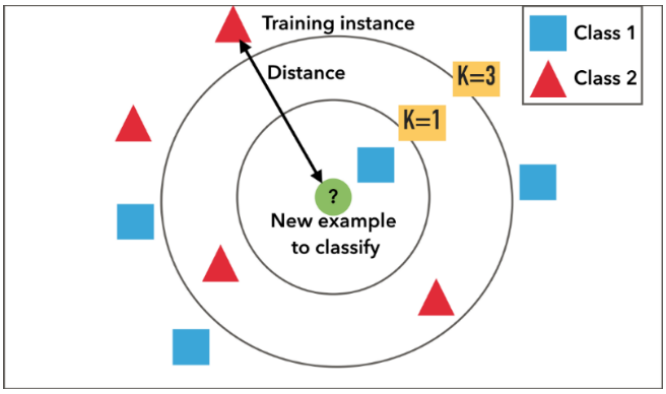
Im Gegensatz dazu macht man im Klassifikationsfall zumeist vom „Majority Vote“ Gebrauch und verwendet entsprechend diejenige Klasse g als Prädiktion, welche bei den k nächsten Nachbarn am häufigsten aufgetreten ist. Die zugehörige mathematische Formulierung findet sich in Gl. 4.4.

$$\hat{y}_\bullet = \arg\max_g \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x}_\bullet)} \mathbb{I}(y_i = g) \tag{4.4}$$

Ebenso kann dann auf einfache Weise hierfür die posteriori Wahrscheinlichkeit errechnet bzw. geschätzt werden als:

$$\hat{\pi}_g(\boldsymbol{x}_\bullet) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x}_\bullet)} \mathbb{I}(y_i = g) \tag{4.5}$$

Um sich die bisherigen Erklärungen zur kNN-Methode im Klassifikationsfall nun einmal bildhaft vor Augen führen zu können, sei Abb. 4.1 herangezogen. Angestrebt wird die Prädiktion der Klasse einer neuen Beobachtung, welche sich als grüner Punkt im zweidimensionalen Raum - entsprechend also zwei betrachtete Features/Einflussgrößen - zwischen den übrigen Datenpunkten mit bekannter Klasse 1 oder 2 findet. Sollte nur ein direkter Nachbar ($k = 1$) in Betracht gezogen werden, so würde für die neue Beobachtung Klasse 1 prognostiziert werden. Werden dagegen $k = 3$ der Distanz-mäßig nächsten Nachbarn betrachtet, so würde gemäß dem Majority Vote eine Prognose von Klasse 2 für $\hat{y}_\bullet$ getroffen.



Quelle: Bronshtein (2017)

Abb. 4.1: Beispielhafte Veranschaulichung zu kNN: Prädiktion in Abhängigkeit von der Anzahl in Betracht gezogener Nachbarn k

Generell kann es gewünscht sein Nachbarn, die näher an dem interessierenden Punkt liegen, auch mit höherer Gewichtung zu berücksichtigen als diejenigen Datenpunkte, die zwar innerhalb der Nachbarschaft liegen, jedoch eine größere Distanz zu $\boldsymbol{x}_\bullet$ aufweisen. Wie dadurch schon anklingt greift man hierfür auf Gewichte zurück, die beispielsweise die folgende Erscheinung haben können: $w_i \propto 1/d(\boldsymbol{x}_\bullet, \boldsymbol{x}_i)$. Hinsichtlich dieser Thematik sei auf Kernmethoden verwiesen und damit z.B. auf Scheipl et al. (2018, Kap. 9) oder Hastie et al. (2008, Kap. 6), wobei diesem in nachfolgender Sektion 4.2 noch Anklang finden.

Zusätzliche Anmerkungen

Generell wurden die Grundzüge bzw. kNN als solches schon vor langer Zeit entwickelt und aus diesem Grund sind dieser Methodik in unterschiedlichen Disziplinen auch verschiedene Namen zu eigen. Dazu zählt beispielsweise der Begriff „Instance-Based Learning“, was sich in der Tatsache begründet, dass die rohen Trainingsdaten bzw. Instanzen aus dem Trainingsset verwendet werden um Prädiktionen zu schaffen. Gleichzeitig sollte hierdurch aber auch der hohe computationale Aufwand ersichtlich werden, der mit zunehmender Anzahl der Trainingsdaten ebenfalls zunimmt. Genau genommen wird das Modell also gar nicht trainiert, sondern alle Arbeit findet dann statt, wenn eine Prädiktion verlangt wird. Dies verschafft der kNN-Methode auch den Namen „Lazy Learning“ und soll insbesondere bei hoher Dimensionalität der Daten die starke Verlangsamung des Algorithmus zum Ausdruck bringen. Dazu sei vermerkt, dass kNN durchaus gute Leistungen erbringen kann, falls die Anzahl der Input-Variablen p relativ klein ist, sich jedoch deutliche Schwierigkeiten ergeben, wenn diese Zahl sehr groß ist. Dies kann als Dimensionen-Problem angesehen werden: jede betrachtete Einflussgröße entspricht einer Dimension im p-dimensionalen Input-Raum. Mit Zunahme der Dimensionen steigt das Volumen dieses Raumes um eine exponentielle Rate. So können dann auch Datenpunkte, welche recht viel Ähnlichkeit miteinander haben, in diesen hohen Dimensionen große Abstände/Distanzen voneinander aufweisen. Selbst wenn dies zunächst nicht intuitiv erscheint - insbesondere da die Vorstellung von Distanzen, die über den zwei- bzw. drei-dimensionalen Raum hinaus gehen, vom Menschen kaum greifbar sind - stellt das ein häufig auftretendes Problem dar und wird als “Fluch der Dimensionen“ bezeichnet.
Doch auch die Anzahl k der betrachteten Nachbarpunkte kann einen großen Einfluss auf die Genauigkeit in Prognosen durch Nutzung der kNN-Methodik haben. Je kleiner k gewählt ist, desto glatter, aber auch „verwackelter“ werden die Entscheidungsgrenzen. Generell erfährt die Genauigkeit der Prädiktionen auch eine Abwertung durch die Anwesenheit von Stör- bzw. unwichtigen Features oder wenn die Skala von diesen Attributen nicht konsistent mit deren Wichtigkeit ist. So sei nun zu einer weiteren Methodik, die an der räumlichen Aufteilung der Features für die Klassifizierung einer Zielgröße ansetzt, hingeführt, welche als „Support Vector Machines“ (SVM) bezeichnet wird und innerhalb der nachfolgenden Sektion Erklärung findet.

4.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) - im Deutschen auch als Stützvektormaschine bezeichnet - sind lineare Klassifikatoren, deren ursprüngliche Entwicklung auf binäre Klassifikation abzielt. Hierbei werden die Trainingsobjekte als Vektoren im mehrdimensionalen Vektorraum erachtet und je nach Klassenzugehörigkeit mit -1 oder 1 kenntlich gemacht. Ziel ist es nun diejenige Trennlinie (im zweidimensionalen Fall) bzw. Hyperebene (im mehrdimensionalen Fall) zu finden, die die bestmögliche Trennung der beiden Klassen bereitstellt. Sollte es möglich sein

mittels eines linearen Klassifikators die beiden Klassen gänzlich voneinander zu unterscheiden, so spricht man von linear, anderenfalls von nicht-linear separierbaren Daten.
Die nachfolgenden Erklärungen sollen einen groben Überblick über die Funktionsweise der SVM geben. Um dies zu bewerkstelligen wird insbesondere auf Scheipl et al. (2018, Kap.9), Hastie et al. (2008, Kap. 12) und Zhou (2012, Kap. 1.2.6) zurückgegriffen. Hierbei gehe man der Einfachheit halber von angestrebter, binärer Klassifikation aus, wobei natürlich auch Mehr-Klassen-Probleme denkbar sind.

4.2.1 Darstellung der Grundlagen anhand linearer Separation

Insofern eine Menge an Daten linear trennbar ist, ergibt sich eine große - wenn nicht gar unendliche - Auswahl möglicher Hyperebenen, welche diese lineare Separation vornehmen können. Aus diesem Grund gilt es diejenige zu bestimmen, die den größtmöglichen Grenzspalt zwischen den Objekten der einzelnen Klassen zieht. Hierzu sei vermerkt, dass diejenigen Objektvektoren mit dem geringsten Abstand zu eben diesem Spalt als Stütztvektoren bezeichnet werden. Der Abstand, welchen diese Vektoren zu dem Spalt messen, bezeichnet sich als „Margin“ , was sich mit „Rand“ übersetzen lässt. Demgemäß handelt es sich bei der gesuchten Hyperebene um die s.g. „Maximum Margin Hyperplane“ (MMH), wobei die Suche nach eben dieser Hyperebene letztlich ein Optimierungsproblem darstellt. Damit sieht man sich angesichts eines quadratischen Kriteriums mit linearer Ungleichheit als Nebenbedingung, was synonym auch als beschränktes Optimierungsproblem bezeichnet wird. Zur Lösung dessen werden i.A. Langrange-Multiplikatoren/-Funktionen eingeführt. Genauere Ausführungen hierzu können beispielsweise Hastie et al. (2008, Kap. 4.5.2) entnommen werden.
Die entsprechende Lösung ist in der linken Grafik von Abb. 4.2 durch die gelb markierte MMH bereitgestellt. Die Entscheidungsgrenze ist anhand der durchgezogene Linie markiert, wohingegen der maximale Rand - hier $2M = ||\beta||^{-1}$ - durch die gestrichelten Linien gekennzeichnet ist. In diesem Falle bilden drei Stützpunkte die Basis für die Margins und lassen sich als diejenigen Punkte auf der gestrichelten Linie identifizieren. Mathematische Ausführungen sollen nun mit dem Übergang zu nicht-linear separierbaren Daten Einzug nehmen.

4.2.2 Ausweitung auf nicht-lineare Separation

Man nehme im Folgenden einen linearen Klassifikator $y = sign(\boldsymbol{w}^T\boldsymbol{x}_i + b)$ an - was nunmehr mit $\langle \boldsymbol{w}, b \rangle$ abgekürzt wird -, wobei $\boldsymbol{w}$ einen Richtungs-/Normalenvektor bezeichnet und b als Bias oder Offset aufzufassen ist. Zudem wird über $f(\boldsymbol{x}_i) = \boldsymbol{w}^T\boldsymbol{x}_i + b$ die Hyperebene selbst definiert. Damit kann schließlich der s.g. Hinge-Verlust herangezogen werden, um die Anpassung an die Daten zu beurteilen:

$$L(\gamma) = \max\{0, 1 - \gamma\} = \sum_{i=1}^{m} \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\} \tag{4.6}$$

Des Weiteren kann man auf die euklidische Distanz zurückgreifen, um den Abstand einer Instanz $\boldsymbol{x}_i$ zur Hyperebene auszudrücken:

$$d_{euklidisch}(f, \boldsymbol{x}_i) = \frac{|\boldsymbol{w}^T \boldsymbol{x}_i + b|}{||\boldsymbol{w}||} \tag{4.7}$$

Unter zusätzlicher Berücksichtigung der Restriktion $|\boldsymbol{w}^T \boldsymbol{x}_i + b| \geq 1$ für jede der Instanzen i, ist die minimale Distanz zur Hyperebene also gegeben durch $||\boldsymbol{w}||^{-1}$. In dieser Distanz von der Hyperebene entfernt liegen also die Stützvektoren und demzufolge maximieren SVMs dann $||\boldsymbol{w}||^{-1}$ durch Lösen von Gl. 4.8, wobei stattdessen auch $0.5 \cdot ||\boldsymbol{w}||^2$ minimiert werden kann, und wiederum von einem beschränkten Optimierungsproblem auszugehen ist. Die Lösung zu diesem konvexen, quadratischen Optimierungsproblem wird als „Soft Margin SVM“ bezeichnet. Hierbei finden nun die Fehler ξ_i, die durch die Separation auftreten Berücksichtigung. Hierbei misst die s.g. Slack Variable ξ_i den Abstand Gleichzeitig wird der Parameter C eingeführt, der das Fehlergewicht steuert und entsprechend den Abstand zur Hyperebene multipliziert. In der Literatur finden sich für diesen unterschiedlich Bezeichnungen wie etwa Trade-Off-, Regularisierungs- oder „Complexity Control“ Parameter.

$$\langle \boldsymbol{w}^\star, b^\star \rangle = \underset{\boldsymbol{w},b,\xi}{\arg\min} \frac{||\boldsymbol{w}||^2}{2} + C \sum_{i=1}^{m} \xi_i, \tag{4.8}$$

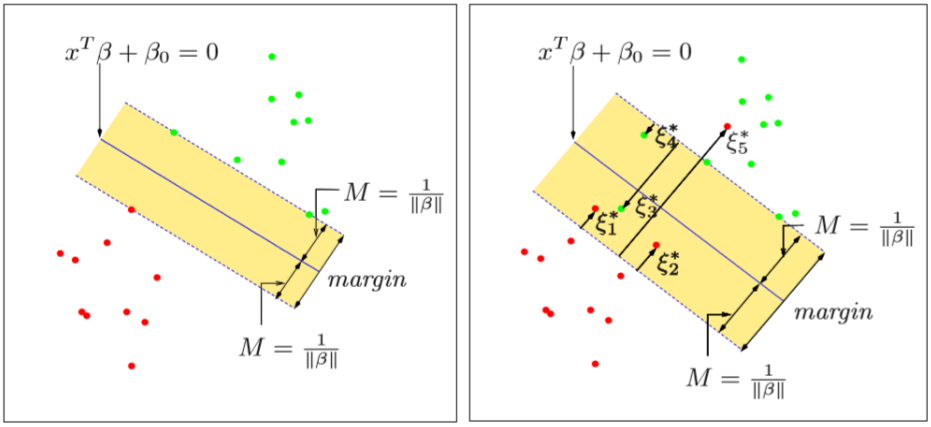
$$\begin{aligned} so\ dass \quad & y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad \text{für } \forall i = 1, \ldots, m \end{aligned}$$

Durch letztere Bedingung aus Gl. 4.8 wird das Fehlergewicht C derart festgesetzt, dass ein Kompromiss zwischen einem großen Spalt mit großen Fehlern und einem kleinen Spalt mit kleinen Fehlern realisiert wird. Innerhalb dieser Soft Margin SVMs gilt es also nun prinzipiell zwischen drei Typen von Trainingsinstanzen zu unterscheiden, welche sich wiederum allesamt in der rechten Grafik von Abb. 4.2 wiederfinden.

- Beobachtungen, die keine Stützvektoren darstellen und damit einen Margin größer 1 aufweisen. Diese bedürfen keiner weiteren Berücksichtigung und können entfernt werden.
- Einige Stützvektoren befinden sich direkt auf dem Rand der Hyperebene und weisen einen Margin von exakt 1 auf.
- Wieder andere Stützvektoren sind s.g. „Margin Violators“ und kennzeichnen sich durch eine Margin kleiner 1. Gleichzeitig weisen diese dann eine zugehörige Slack Variable $\xi_i > 0$ oder im Falle von Missklassifikation $\xi_i \geq 1$.

Zurückkommend auf die rechte Grafik von Abb. 4.2 lässt sich nunmehr Folgendes festhalten: die durchgezogene Linie markiert wiederum die Entscheidungsgrenze und entspricht der Lösung von $\boldsymbol{w}^T \boldsymbol{x} + b = 0$. Gleichzeitig erfahren diejenigen Punkte, die mit einem $\xi_i^\star$ versehen

sind, eine Zuteilung zur falschen Seite des Randes in einem Umfang von hier $\xi_i^\star = M \cdot \xi_i$. In der bisherigen Schreibweise ist dies folgendermaßen gleichzusetzen: die gestrichelten Linien und damit die äußerste Grenze des Randes lässt sich über Lösen der von $|\boldsymbol{w}^T\boldsymbol{x} + b| = 1$ ermitteln und entsprechend ergibt sich der Abstand dieser beiden Linien als $2/||\boldsymbol{w}||$. Der Rand wird dabei definiert als maximales Subjekt zu einem Gesamtbudget von $\sum \xi_i \leq const$, wobei $\sum \xi_i$ die absolute Distanz der Punkte auf der falschen Seite des Margins widerspiegelt.



Quelle: (Hastie et al., 2008, S. 418)

Abb. 4.2: Beispielhafte Veranschaulichung zu SVMs
links: Darstellung der Hyperebene für den Fall linearer Separierbarkeit;
rechts: Darstellung der Hyperebene für den Fall nicht-linearer Separierbarkeit

Wenn nun die Anmerkung aus der vorhergehenden Untersektion erneut aufgegriffen wird und man in diesem Zuge noch einmal auf Langrange-Funktionen zu sprechen kommt, so lässt sich die in Gl. 4.8 gegebene Darstellung als s.g. primale Form der Optimierung deklarieren. Die s.g. duale Form kann dagegen anhand der Langrange-Multiplikatoren und der Karush-Kuhn-Tucker-Bedingungen ermittelt werden, indem der Normalenvektor $\boldsymbol{w}$ umgeschrieben wird zu einer Linearkombination aus den Trainingsinstanzen.

$$\boldsymbol{\alpha}^\star = \arg\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle, \tag{4.9}$$

$$so\ dass \quad \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \quad für\ \forall i = 1, \ldots, m$$

In der vorhergehenden Gl. 4.9 bezeichnen $\langle \cdot, \cdot \rangle$ das Skalarprodukt. Entsprechend kann nun die Lösung von $\boldsymbol{w}^\star$ in dualer Form nun folgendermaßen dargeboten werden:

$$\boldsymbol{w}^\star = \sum_{i=1}^{m} \alpha_i^\star y_i \boldsymbol{x}_i \tag{4.10}$$

Das Skalarprodukt zwischen den Instanzen $\boldsymbol{x}$ und $\boldsymbol{w}^\star$ errechnet sich schließlich wie in Gl. 4.11.

$$\langle \boldsymbol{w}^\star, \boldsymbol{x} \rangle = \sum_{i=1}^{m} \alpha_i^\star y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle \tag{4.11}$$

Dieses Vorgehen innerhalb der SVMs gerät an seine Grenzen, wenn derartige lineare Klassifikatoren genutzt werden, um nicht-lineare Daten zu separieren. Offensichtlich können dann die Klassen nur schwer voneinander getrennt werden. Ein möglicher Ansatz besteht in der Überführung der Datenpunkte in einen höher-dimensionalen Feature-Raum, in welchem die Daten linear separierbar werden. Gemäß Zhou (2012, S.11) tendiert diese Art des Lernprozesses dazu stark zu verlangsamen und zusätzlich sieht man sich mit der Problematik konfrontiert, dass das Gleichungssystem gar unlösbar werden kann, nachdem sich Schwierigkeiten mit der Berechnung des Skalarproduktes im hoch-dimensionalen Raum offenbaren können.

4.2.3 Kernelfunktionen

Stattdessen lässt sich auf die nicht-lineare Erweiterung mittels s.g. Kernelfunktionen zurückgreifen, welche abgekürzt auch als Kerne bezeichnet werden. Der Feature-Raum (FS = „Feature Space“), welcher sich von diesen Kernen ableitet wird „Reproducing Kernel Hilbert Space“ (RKHS) genannt. Hierbei entspricht ein Skalarprodukt im RHKS der Kernel-Abbildung des Skalarproduktes der Instanzen im originalen, niedriger-dimensionalen Feature-Raum. Bezeichne ϕ die Abbildung des originalen FS zu einem höher-dimensionalen Raum und K eine Kern, dann kann die nachfolgende Gl. 4.12 geltend gemacht werden, wobei lediglich die duale Form des Skalarproduktes der Optimierung durch die Kernelfunktion ersetzt wird.

$$K(\boldsymbol{x}_{,}\boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle \tag{4.12}$$

Gemäß Cristianini und Shawe-Taylor (2000) ist Mercer's Theorem folgend jede positiv semidefinte, symmetrische Funktion eine Kernelfunktion. In diesem Sinne sind nachfolgend in Tab. 4.1 drei bekannte Kerne dargeboten. Sobald der Kernel-Trick benutzt wurde spricht man von Kernmethoden, sodass sich entsprechend auch im Falle von SVMs ein derartiger Spezialfall etablieren lässt.

Bezeichnung des Kerns	**Mathematische Formulierung**
Linearer Kern	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$
Polynomialer Kern	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle^d$
Gaußkern	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$

Tab. 4.1: Bekannte Kernelfunktionen zur potentiellen Anwendung innerhalb von SVMs, wobei d den Grad des Polynoms und σ die Standardabweichung bezeichnen

Es lässt sich wohl unbeirrt die Behauptung aufstellen, dass SVMs eine der bekanntesten Methodiken innerhalb des ML's darstellen. Eine weitere unwegdenkbare und zugleich deutlich intuitivere Methode stellen wohl Entscheidungsbäume dar. Die Erläuterungen hierzu mit Fokus auf Bäume, die auf konditionaler Inferenz beruhen, stellen den Schwerpunkt der nachfolgenden Sektion dar.

4.3 Entscheidungsbäume basierend auf konditionaler Inferenz (cTree)

4.3.1 Entscheidungsbäume im Allgemeinen

Aufgrund der verhältnismäßig einfachen Handhabung, sowie der Transparenz und gleichzeitig relativ hohen Performance haben sich Entscheidungsbäume zu populären Methodiken des ML's entwickelt. Hierbei soll mittels aufeinanderfolgender, zumeist binärer Aufteilungen die zunächst heterogene Datengesamtheit in möglichst homogene Gruppen unterschieden werden. Für derartige Entscheidungen kann eine grafische Darstellung anhand eines Baumes vorgenommen werden, welcher sich in einem Wurzel-Knoten (Root Node) beginnend, von oben nach unten absteigend in Form von Ästen aufspaltet. Unter der Annahme binärer Entscheidungen bringt dabei ein Elternknoten (Parent Node) jeweils zwei in sich homogenere Tochterknoten (Child Nodes) hervor. Gleichzeitig spricht man von inneren Knoten (Internal Nodes), insofern von einem Knoten Äste abgehen, und von terminalen Knoten (Terminal Nodes), falls sich ein Knoten nicht weiter aufspaltet und damit einen Endpunkt des Baumes markiert. Diesen Endknoten wird schließlich - abhängig von Modal- bzw. Mittelwert (je nach Skalenniveau und damit Klassifikations- oder Regressionsbaum ausmachend) - ein zugehöriger Wert der Zielvariable zugeteilt. Somit ist durch die Erstellung eines Entscheidungsbaumes unmittelbar ein Regelwerk zur Prognose der abhängigen Variable bei der Übergabe neuer Daten gegegeben.
Grundsätzlich gilt es nun zwischen zwei Typen von Entscheidungsbäumen zu unterscheiden: die einen lassen nur binäre Splits zu, wohingegen andere maximal so viele Zweige pro Aufteilungspunkt erlauben, wie die diesem Knoten zugehörige Variable Kategorien aufweist. Im weiteren Verlauf dieser Arbeit werden ausschließlich Bäume ersteren Typus betrachtet. Um letztlich an einem Knotenpunkt ausmachen zu können, welche Variable für die Aufteilung herangezogen wird, können nunmehr Unterschiede zwischen verschiedenen Algorithmen der Entscheidungsbäume erkannt werden. Diese Kriterien können sich einerseits in Informationsmaßen, die als Maß der „Unreinheit" (s.g. Impurity) dienen, widerspiegeln und andererseits durch statistische Tests zum Ausdruck kommen. Die Entscheidungen an jedem inneren Knoten bzw. Split werden dabei stets so gewählt, dass die Aufteilung in möglichst trennscharfe Gruppen, die in sich möglichst „rein" bzw. homogen sind, erfolgt.
Innerhalb der Entscheidungsbäume, welche auf statistischen Tests beruhen, ist gleichzeitig ein Kriterium gegeben, welches „automatisch" die weitere Verästelung stoppen, wenn das betrach-

tete Maß sich durch weitere Aufspaltung nicht signifikant verbessert. Dagegen ist anhand von Informationsmaßen keine Aussage möglich, ob sich durch weitere Abzweigungen signifikante Besserungen der Kriterien einstellen. Entsprechend werden derartige Bäume meist solange erweitert, bis ein Stopp-Kriterium erreicht ist oder - im Extremfall - alle für das Training herangezogenen Beobachtungen einem terminalen Knoten ausmachen. Da damit starkes Overfitting an die Daten vorliegt, werden diese Bäume im Zuge des s.g. Prunings zurückgestutzt. Hierbei gilt es den Trade-Off zwischen Modellgüte und -komplexität zu optimieren. Wenngleich das Overfitting mittels des Prunings eingedämmt werden kann, sind derartige Bäume zugleich anfällig für Verzerrungen in der Variablenselektion hin zu unabhängigen Variablen mit vielen Splitmöglichkeiten oder fehlenden Werten. Einen Ausweg stellt beispielsweise der Entscheidungsbaum basierend auf konditionaler Inferenz und damit auf statischen Tests beruhend - im Weiteren bezeichnet mit cTree (Conditional Inference Tree) - bereit und findet nun folgend eingehende Beschreibung.

4.3.2 cTree: Bäume basierend auf konditionaler Inferenz

Die zuvor genannte Problematik des Selektionsbias, welcher sich u.a. für den CART-Algorithmus erkennen lässt, wollen Hothorn, Hornik und Zeileis (2006) entgegentreten durch die Schaffung eines einheitlichen Rahmens für rekursive Partitionierung, bei welcher zunächst Regressionsbäume in die wohldefinierte Theorie von Prozeduren bedingter Inferenz eingebettet werden. Ihrer kennzeichnenden Eigenschaft zu Dank wird in Hinblick auf diese auch häufig von unverzerrter rekursiver Partitionierung gesprochen. Mit Verweis auf Klassifikations- und Regressionsbäume beschrieben von Breiman et al. (1984) sind die nachfolgenden Abschnitte derart konstruiert, dass insbesondere Unterschiede in den Algorithmen von CART und cTree hervorgehoben werden.

Grundlagen und Abgrenzung zu CART

Grundlegende Eigenschaften von cTree sind die rekursive Partitionierung mitsamt Stopp-Kriterien basierend auf multiplen Testverfahren (Early Stopping). Entsprechend beruht letzteres, sowie die Variablenselektion und die Suche der Splitpunkte auf konditionaler Inferenz. Gleichzeitig kann die Methodik von cTree für jegliche Art von Regressionsproblem angewandt werden, was letztlich bedeutet, dass dieses nicht nur nominal, ordinal oder numerisch sein kann, sondern beispielsweise zensiert oder multivariat, sowie auch die Kovariablen ein beliebiges Skalenniveau aufweisen können. Insbesondere bezeichnen Hothorn, Hornik und Zeileis (2006) die bedingte Verteilung des statistischen Maßes, welches die Assoziation zwischen Kovariablen und Response misst, als Grundlage für eine unverzerrte Auswahl zwischen Variablen unterschiedlichen Skalenniveaus.
Vorrangiges Ziel bestand in der Entwicklung rekursiver Partitionierung in Richtung der Forderung eines statistischen Ansatzes, der die verteilungsbedingten Eigenschaften in die jewei-

lige Messmethodik einbezieht. Die in diesem Kontexte einwickelte cTrees sind gekennzeichnet durch die Einbettung in die wohldefinierte Theorie von

1. Permutationstests, welche ihre Begründung in Strasser und Weber (1999) finden, sowie
2. Tests auf Parameter-Instabilität bei (parametrischen) Regressionsmodellen

Insbesondere sind die Variablenselektion, sowie die Suche nach einem möglichen Splitpunkt zwei voneinander getrennte Schritte, wobei jedoch alle Entscheidungen auf Hypothesen- bzw. genauer Permutationstest beruhen. Die Problematik lässt sich also so beschreiben, dass die Verteilung des - möglicherweise multivariaten - Response $\mathbf{Y} \in \mathcal{Y}$ auf einen P-dimensionalen Kovariablenvektor $\boldsymbol{X} = (X_1, \ldots, X_P) \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P$ zurückgeführt werden. Zusammenfassend soll somit Gl. 4.13 bzw. f basierend auf einem n Beobachtungen umfassenden Lern-Datensatz $\mathcal{L}_n = \{(\boldsymbol{Y}_i, X_{1i}, \ldots, X_{Pi}; i = 1 \ldots, n)\}$ mit einer möglichen Anzahl von $\boldsymbol{w} = (w_1, \ldots, w_n)$ Fällen ermittelt werden. Hierbei wird angenommen, dass die bedingte Verteilung $\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X})$ des Response gegeben den Kovariablen in $\boldsymbol{X}$ von einer Funktion f der Kovariablen abhängt.

$$\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X}) = \mathbb{P}(\boldsymbol{Y}|X_1 \ldots, X_P) = \mathbb{P}(\boldsymbol{Y}|f(X_1, \ldots, X_P)) \tag{4.13}$$

Ein generischer Algorithmus für die Formulierung der rekursiven Partitionierung kann nun wie nachfolgend gelistet formuliert werden. Dabei wird jeder Knoten des Baumes repräsentiert durch einen Vektor von Gewichten, dessen Eintrag nicht-Null ist, wenn die entsprechenden Beobachtungen Elemente des Knotens sind und Null anderenfalls.

1. Teste für die Fälle/Gewichte $\boldsymbol{w}$ die globale Hypothese der Unabhängigkeit zwischen jeder der P Kovariablen und dem Response. Stoppe, sobald eine der Hypothesen nicht ablehnt werden kann. Wähle anderenfalls diejenige Kovariate $X_{p^\star}$, welche die stärkste Assoziation mit $\boldsymbol{Y}$ aufweist.
2. Wähle ein Set $A^\star \subset \mathcal{X}_{p^\star}$, sodass $\mathcal{X}_{j^\star}$ in zwei disjunkte Sets $A^\star$ und $\mathcal{X}_{p^\star} \setminus A^\star$ aufgeteilt wird. Dabei definieren die Gewichte $\boldsymbol{w}_{links}$ und $\boldsymbol{w}_{rechts}$ die zwei Subgruppen mit $w_{links,i} = w_i \cdot \mathbb{I}(\mathcal{X}_{p^\star i} \in A^\star)$ und $w_{rechts,i} = w_i \cdot \mathbb{I}(\mathcal{X}_{p^\star i} \notin A^\star)$ für alle $i = 1, \ldots, n$.
3. Wiederhole rekursiv die Schritte 1 und 2 mit jeweils modifizierten Gewichten der Fälle $\boldsymbol{w}_{links}$ und $\boldsymbol{w}_{rechts}$.

Aus diesem generischen Algorithmus wird die bereits genannte separate Betrachtung der Variablenselektion und Split-Prozedur in Schritt 1 und 2 ersichtlich, welcher die Schlüsselrolle in der Interpretierbarkeit von Baumstrukturen ohne die Problematik der systematischen Tendenz hin zu Kovariablen mit vielen möglichen Splits oder einer großen Anzahl fehlender Werte zukommt. Der erste Schritt beschreibt hierbei die Variablenselektion.

Variablenselektion und Stopp-Kriterium

Im ersten Schritt des dargestellten generischen Algorithmus sieht man sich der Problematik der Unabhängigkeit gegenüber und damit der Frage, ob eine der zur Verfügung stehenden Kovariablen überhaupt Informationen über den Response beinhaltet. Entsprechend wird in jedem durch die Fall-Gewichte $\boldsymbol{w}$ identifizierten Knoten die globale Nullhypothese der Unabhängigkeit formuliert als $H_0 = \bigcap_{p=1}^{P} H_0^p$ mittels der partiellen Hypothesen $H_0^p : \mathbb{P}(\boldsymbol{Y}) = \mathbb{P}(\boldsymbol{Y}|X_p)$ geprüft. Letztere Hypothese beschreibt also die Annahme, dass Variable X_j keinen Einfluss auf $\boldsymbol{Y}$ besitzt bzw. entsprechend durch die globale Hypothese die Annahme, dass keine der p potentiellen Größen einen Einfluss auf $\boldsymbol{Y}$ nimmt. Sollte es nicht möglich sein das globale H_0 zu einem prespezifizierten Signifikanzniveau α abzulehnen, so wird die Rekursion gestoppt. Angemerkt sei an dieser Stelle, dass aufgrund der Problematik des multiplen Testens eine Korrektur des jeweiligen α-Niveaus vorzunehmen ist. Für den Fall, dass H_0 abgelehnt wird, wird die Assoziation zwischen $\boldsymbol{Y}$ und jeder der Kovariaten X_p durch Teststatistiken oder p-Werte gemessen, die eine Abweichung von den partiellen Hypothesen H_0^p indizieren.
Die Assoziation zwischen $\boldsymbol{Y}$ und X_p kann schließlich über eine lineare Statistik der nachstehenden Form gemessen werden, wobei sich eine derartige Formulierung von Strasser und Weber (1999) ableitet:

$$\mathbb{T}_p(\mathcal{L}_n, \boldsymbol{w}) = vec(\sum_{i=1}^{n} w_i \cdot g_p(X_{pi}) \cdot h[\boldsymbol{Y}_i, (\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n)]^T) \in \mathbb{R}^{p_j q} \tag{4.14}$$

$$\begin{aligned} mit\ g_p &= Transformation\ von\ X_p; mit g_p : \mathcal{X}_p \to \mathbb{R}^{v_p} \\ h_j &= Einflussfunktion\ h : \mathcal{Y} \times \mathcal{Y}^n \to \mathbb{R}^q \\ vec &= Transformation\ der\ v_p \times q\text{-}Matrix\ in\ einen\ v_p q\text{-}Spaltenvektor \end{aligned}$$

Gleichzeitig sei angemerkt, dass die Einflussfunktion $h(\cdot)$ von dem Response $(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n)$ auf Permutations-symmetrische Weise abhängt. Für die Wahl von $h(\cdot)$, sowie $g_p(\cdot)$ sei Hothorn, Hornik und Zeileis (2006, Kap. 4) herangezogen. Die Verteilung von $\mathbb{T}_p$ unter H_0^p ist dabei abhängig von der gemeinsamen Verteilung von $\boldsymbol{Y}$ und X_p, welche zumeist in allen Situationen praktischer Anwendung unbekannt ist. Jedoch kann man sich dieser Abhängigkeit zumindest unter der Nullhypothese entledigen, indem die Kovariablen fixiert werden und auf alle möglichen Permutationen des Response bedingt wird. Dieses Prinzip führt letztlich zu Prozeduren, die als Permutationstests bekannt sind. Für eingehende Erklärungen des sich hierzu ergebenden bedingten Erwartungswertes $\mu_p \in \mathbb{R}^{v_p q}$ und Kovarianz $\Sigma_j \in \mathbb{R}^{v_p q \times v_p q}$ sei auf Hothorn, Hornik und Zeileis (2006, S. 5), sowie Strasser und Weber (1999) verwiesen.
Schließlich kann nun der bedingte Erwartungswert und Kovarianz genutzt werden, um die lineare Statistik $\mathbb{T} \in \mathbb{R}^{vq}$ aus Gl. 4.14 für $v \in \{v_1, \ldots, v_P\}$ zu standardisieren. Letztlich kann dann für die beobachtete, (multivariate) lineare Statistik $\mathbf{t}$ eine (univariate) Teststatistik c zum Prüfen von H_0 konstruiert werden, die beispielsweise die in Gl. 4.15 dargestellte Form

annehmen kann. Hiermit ist das Maximum der absoluten Werte der standardisierten, linearen Statistik fokussiert.

$$c_{max}(\mathbf{t}, \mu, \Sigma) = \max_{k=1,\ldots,vq} \left| \frac{(\mathbf{t} - \mu)_k}{\sqrt{(\Sigma)_{kk}}} \right| \tag{4.15}$$

Bedeutend ist hier anzumerken, dass Teststatistiken $c(t_p, \mu_p, \Sigma_p)$ mit $p = 1, \ldots, P$ nicht direkt bzw. auf unverzerrte Weise miteinander verglichen werden können, solange nicht alle Kovariablen auf der gleichen Skala gemessen sind - also $v_1 = v_p$, $p = 2, \ldots, P$ gilt. Um folglich eine unverzerrte Variablenselektion zu gewährleisten, muss eine Betrachtung auf Ebene der p-Werte erfolgen, da für diese die genannte Problematik nicht besteht. Entsprechend wird dann in Schritt 1 diejenige Variable $X_{p\star}$ ausgewählt, die den geringsten p-Wert aufweist.

Splitkriterien

Nachdem in Schritt 1 nun die Variablenselektion erfolgt ist, kann Schritt 2 und damit die Wahl des Splitpunktes anhand unterschiedlicher Splitkriterien - etwa Gini-Index oder SSE - erfolgen. Abgesehen von binären Splits kann innerhalb der cTrees auch eine Aufteilung der Knoten in mehr als zwei Tochterknoten erfolgen. Problematisch an den meisten Splitkriterien ist jedoch, dass diese insbesondere im Hinblick auf relatives Skalenniveau der Responsevariable nicht geeignet bzw. anwendbar sind. Aus diesem Grund ist es sinnvoll, die Nutzung der zuvor beschriebenen Permutationstests in diesem Kontext auszuweiten und binäres Splits für die jeweils ausgewählte Einflussgröße $\mathbf{X}_{p^\star}$ zu bestimmen. Die Güte eines Splits wird dabei anhand von Zwei-Stichproben linearen Statistiken evaluiert. Diese lassen sich als Spezialfall der linearen Statistik aus Gl. 4.14 erkennen und können für alle mögliche Subsets A des Stichprobenraums $\mathcal{X}_p^\star$ folgendermaßen dargestellt werden:

$$\mathbb{T}_{p^\star}^A(\mathcal{L}_n, \boldsymbol{w}) = vec\Big(\sum_{i=1}^{n} w_i \mathbb{I}(X_{p^\star i} \in A) \cdot h(\boldsymbol{Y}_i, (\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n))^T\Big) \in \mathbb{R}^q \tag{4.16}$$

Hieraus kann dann der beste Split abgeleitet werden als

$$A^\star = \arg\max_A c(\mathbf{t}_{p^\star}^A, \mu_{p^\star}^A, \Sigma_{p^\star}^A) \tag{4.17}$$

Nachfolgend seien noch einige zusätzliche Anmerkung zu den Entscheidungsbäumen basierend auf konditionaler Inferenz gemacht.

Zusätzliche Anmerkungen

Ebenso wie innerhalb des CART-Algorithmus kann auch bei Anwendung von cTrees mit fehlenden Werten umgegangen werden. Sollte eine Beobachtung X_{pi} der Kovariable X_p einen fehlenden Eintrag aufweisen, wird für die Berechnung von $\mathbb{T}_p(\mathcal{L}_n, \boldsymbol{w})$ das zugehörige Fall-Gewicht w_i auf Null gesetzt. Sollte ein Split in X_p vorgenommen werden, so findet sich für

$\mathbb{T}_p^A(\mathcal{L}_n, \boldsymbol{w})$ dasselbe Vorgehen. Sobald ein Split $A^\star$ in X_p besteht, können surrogate Splits durch Suchen von Trennlinien, die nahezu die gleiche Aufteilung der Beobachtungen hervorbringen, entwickelt werden.
Ein zusätzliches Augenmerk sollte auch auf die Wahl des α-Parameters gelegt werden. Dieser lässt sich auf zwei unterschiedliche Arten interpretieren: zum einen als prespezifiziertes, nominales Level der zugrundeliegenden Assoziationstests oder als einfacher Hyperparameter, der mitentscheidend ist für die Größe des Baumes. Folglich kontrolliert α im ersteren Sinne die Wahrscheinlichkeit einer fälschlichen Ablehnung von H_0 in jedem Knoten. Andererseits kann dieser Parameter wie gesagt auch als Hyperparameter angesehen werden, was anhand der nachfolgenden Erläuterung verdeutlicht werden soll. Auch wenn die Testprozeduren, welche für die Erstellung des Baumes herangezogen werden, allgemein Tests auf Unabhängigkeit sind, so weisen diese dennoch nur für eine spezifische Richtung der Abweichung von der Unabhängigkeit - was durch die Wahl von $g(\cdot)$ und $h(\cdot)$ gesteuert wird - eine hohe Power auf und zeigen eine geringe Power für jede andere Richtung der Diskrepanz. Folglich könnte man die Strategie verfolgen möglichst jeglichen Typ von Abhängigkeit zu erkennen, indem man das Signifikanzlevel entsprechend hoch setzt. Um durch einen großen α-Wert hervorgerufenes Overfitting zu vermeiden, könnte schließlich auf eine Art des Prunings zurückgegriffen werden. Allerdings geht für Fälle dergleichen die Interpretation von α als nominales Signifikanzlevel von konditionalen Testprozeduren verloren. So jedoch wird α zu einem Hyperparameter der in Richtung bestimmter Risiko-Schätzungen optimiert werden soll.
Zurückgreifend auf vorhergehende Erläuterungen werden also multiple Testprozeduren angewandt, um eine fehlende Assoziation zwischen Kovariablen und Response auszumachen, und somit einen Abbruch der Rekursion einzuleiten. Hothorn, Hornik und Zeileis (2006) zeigten, dass dieses statistisch motivierte Stopp-Kriterium - implementiert über Hypothesentests - zu Regressionsmodellen führte, deren prädiktive Performance gleichzusetzen ist mit der von optimal gestutzten Bäumen.
Ganz allgemein gesprochen weisen gemäß James et al. (2013, S. 316) allerdings Entscheidungsbäume - also sowohl CART, als auch cTree - ein geringeres Level prädiktiver Genauigkeit auf, als andere zur Verfügung stehende ML-Methoden, mittels derer Regressions- und Klassifikationsproblemen entgegengetreten werden kann. Zudem sind Bäume zumeist nicht besonders robust, was bedeutet, dass mit geringfügigen Änderungen in den Daten große Änderungen in dem resultierenden Baum auftreten können. Diesen Nachteilen kann durch Erweiterungen wie Bagging, Boosting oder Random Forests (RF) Abhilfe geschafft werden. Ersteres leitet sich von dem Fachwort „Bootstrap Aggregation“ ab und eignete sich insbesondere für Prozeduren mit hoher Varianz, aber niedrigem Bias - also v.a. auch im Hinblick auf Entscheidungsbäume. [vgl. Hastie et al. (2008, S.587)] In der nachfolgenden Sektion soll nun insbesondere genauer auf die bereits namentlich genannten RFs eingegangen werden.

4.4 Ensemblemethoden

4.4.1 Ensemblemethoden im Allgemeinen

Die folgende Einführung in die Grundkonzepte der Ensemblemethoden ist hauptsächlich nach Zhou (2012, Kap. 1.4) ausgerichtet. Bis hier hin sollte bereits ersichtlich geworden sein, dass es eine Reihe von ML-Methoden gibt, mittels derer eine Lernaufgabe mehr oder minder erfolgreich bewältigt werden kann. Hierbei kann es gerade bei den „einfachen“ oder Basis-Methoden vorkommen, dass deren Performance als insuffizient zu erachten ist. So wurde beispielsweise bereits in der vorhergehenden Sektion 4.3 angeführt, dass Entscheidungsbäume anfällig hinsichtlich ihrer Robustheit sein können, falls kleine Änderungen der Daten vorgenommen werden. Doch nicht nur diese ML-Methodik, sondern auch andere - insbesondere s.g. schwache (weak) - Lerner weisen gewisse Schwachstellen und damit Einbußen in ihrer Performance auf. So kann es durchaus sein, dass durch einfache bzw. schwache Lerner nur geringfügig bessere Prädiktionen erzielt werden als durch zufälliges Raten.
Derartige Problemstellungen führten letztlich zu der Idee mehrere Lernmethodiken miteinander zu kombinieren, um so deren jeweiligen Schwächen entgegenzutreten. Grob formuliert verlässt man sich nicht nur auf einen „Experten“, sondern beachtet mehrere Meinungen und konstruiert auf deren Basis eine „gemittelte“ Entscheidung. In diesem Sinne entwickelten sich unterschiedliche Ensemblemethoden - vom Französischen zu übersetzen als „zusammen, gemeinsam, das Ganze“ -, denen das Trainieren, sowie Kombinieren und damit die Gruppierung mehrerer Lernmethoden gemeinsam ist. Dieser Sachverhalt lässt sich schließlich wie in Abb. 4.3 darstellen. Hierbei unterscheidet man prinzipiell zwischen homogenen und heterogenen Ensemblemethoden. Die meisten Ensemblemethoden bringen mittels eines Basis-Lern-Algorithmus homogene Basis-Lerner hervor, wohingegen andere Prozeduren mehrere, verschiedene Lernmethoden nutzen und so heterogene Ensembles erzeugen. Zu ersteren zählen etwa Bagging und Boosting, letzterem lässt sich beispielsweise das (Modell-) Stacking zuordnen. Die Fähigkeit zur Generalisierung ist dabei gemäß Zhou (2012, S. 15) für Ensemblemethoden deutlich größer als bei der einfachen Nutzung der zugrundeliegenden Basis-Lerner. Ebenso wird diesen Methodiken nachgesagt, dass mittels derer selbst einfache Basis- bzw. schwache Lerner zu starken Lernern mit akkurater Performance gewandelt werden können.
Zu erwähnen ist wohl auch, dass der computationale Aufwand, der bei der Anwendung von Ensemblemethoden benötigt wird, Zhou (2012, S. 17) zufolge nur geringfügig größer ist als bei der Nutzung eines einzelnen Lerners. Die Begründung hierfür ist durch die Tatsache gegeben, dass selbst bei Gebrauch einer einzelnen Methode mehrere Versionen des Lerners für die Modellselektion oder das Hyperparameter-Tuning erzeugt werden. Dies kann generell mit der Generierung von Basis-Lernern im Ensemble verglichen werden, nachdem die computationalen Kosten für die Kombination der Basis-Lerner zumeist recht gering ist, da die meisten Kombinationsstrategien einfacher Natur sind.

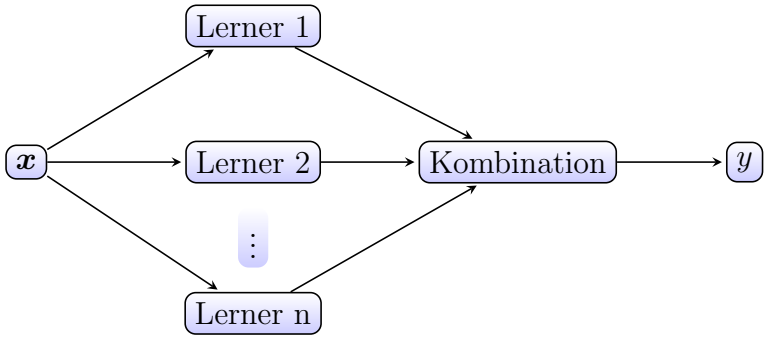


Abb. 4.3: Übliche Architektur von Ensemblemethoden

Neben all diesen Vorteilen, die durch die Nutzung von Ensemblemethoden zum Vorschein kommen, ist wohl der größte Nachteil darin auszumachen, dass diese Prozeduren teils sehr komplex sind und die Ergebnisse entsprechend schwierig zu interpretieren sein können. Nachdem die kombinierten Modelle viele einzelne, individuelle Modelle umfassen, ist ein Verständnis dessen - wie die jeweiligen Faktoren zu verbesserten Entscheidungen beitragen - v.a. im intuitiven Sinne nicht direkt greifbar. [vgl. Witten et al. (2011, S. 352)]
Nachfolgend sollen die bisher sehr allgemein gehaltenen Erklärungen zu den Ensemblemethoden vertieft werden, indem näher auf einen ausgewählten Satz derartiger Prozeduren eingegangen wird. Grob gesprochen gilt es hier nach Zhou (2012, Kap. 3.1) zwei Paradigmen zu unterscheiden, wobei zum einen die parallelen und zum anderen die sequentiellen Ensemblemethoden zu nennen sind. Die grundlegende Motivation bei parallelen Methoden besteht in der Ausnutzung von Unabhägigkeitsstrukturen zwischen den Basis-Lernern, nachdem der Fehler drastisch vermindert werden kann durch Kombination von unabhängigen Lernern. Dagegen zielen sequentielle Methodiken darauf ab Abhängigkeiten zwischen den Basis-Lernern zu erkennen, um so die globale Performance in eine Residuen-reduzierende Richtung zu verbessern. Als Repräsentant paralleler Ensemblemethoden ist das Bagging zu nennen, worauf über s.g. Random Forests (RF) nachfolgend näher eingegangen wird. Für den sequentiellen Typus sei auf Boosting-Methodiken verwiesen, welche in Untersektion 4.4.3 genauer behandelt werden.

4.4.2 Random Forests (RFs) als Erweiterung des Baggings

Heutzutage wohl eine der bekanntesten ML-Methoden sind zweifelsfrei s.g. Random Forests (RFs) - im Deutschen ab und an als Zufallswälder bezeichnet. Diese stellen nun den Hauptaspekt der folgenden Untersektion dar, wobei man hierbei nicht umhinkommt das Bagging, aus welchem sich RFs entwickelten, zu beschreiben. Als Begründer des Baggings lässt sich Breiman (1994) ausmachen, wobei sich der Name der genannten Methodik von „Bootstrap AGGregatING“ ableiten lässt. Diese Namensgebung lässt bereits die grobe Funktionsweise vermuten, sollte jedoch spätestens im Verlauf der nachfolgenden Untersektion deutlich werden.

Grundlagen zum Bagging

Die Ausführung der hier vorgenommenen Erklärungen zum Bagging lassen sich im Wesentlichen zurückführen auf Breiman (1994), Zhou (2012, Kap. 3), sowie James et al. (2013, Kap. 8.2.1). Wie bereits erwähnt führt die Kombination von unabhängigen Basis-Lernern zu einer drastischen Reduktion des Fehlers, sodass eben solche Basis-Lerner angestrebt werden. Eine erste Idee für ein solches Vorhaben wäre also einen gegebenen Trainingsdatensatz in mehrere nicht-überlappende Subsets zu unterteilen und auf jeden einzelnen eine Basis-Lerner anzuwenden. Man denke nun an einen Regressionsfall, bei dem ein Set von n unabhängigen Beobachtungen $A_1, \ldots, A_n$, jedes mit einer Varianz von σ^2 gegeben ist. Somit wäre die Varianz des Mittelwertes $\overline{A}$ gegeben durch σ^2/n und folglich würde das Mitteln über ein Set von Beobachtungen die Varianz reduzieren. Nachdem jedoch nur ein endlicher Trainingsset zur Verfügung steht, würde ein derartiger Prozess - bei dem der Trainingsdatensatz in nicht-überschneidende Subsets geteilt wird - nur kleine und unrepräsentative Stichproben hervorbringen und infolgedessen zu schlechter Performance der Basis-Lerner führen.
Um diese Problematik zu umgehen wird beim Bagging von Bootstrap-Stichproben Gebrauch gemacht. Genauer gesagt wird also aus einem Trainingsdatensatz B Stichproben der Größe m durch Ziehen mit Zurücklegen gezogen. Entsprechend können in den so entstehenden Subsets einige originale Beobachtungen mehrfach enthalten sein, wohingegen andere gar nicht vertreten sind (Bootstrap-Verteilung). Auf Basis jeder der $b = 1, \ldots, B$ Bootstrap-Stichproben kann nun ein Basis-Lernalgorithmus angewandt werden. Hierzu gilt es die durch Bootstrapping erlangten einfachen Lerner zu einem Ganzen zu kombinieren und damit den Aggregations-Schritt auszuführen. Dabei wird beim Bagging i.A. auf „Averaging“ bei Regressions- und auf „Voting“ bei Klassifikationsaufgaben (binären und multiplen Typus) zurückgegriffen. Ersteres kann folglich für die B Bootstrap-Trainingsdatensätze wie in Gl. 4.18 dargestellt werden.

$$\hat{f}_{avg}(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(\boldsymbol{x}), \tag{4.18}$$

mit $\hat{f}^b$ = *Prädiktion des b-ten Lerners basierend auf Bootstrap-Stichprobe b*

Im Falle eines quantitativen Outcomes Y gibt es verschiedene Ansätze zur Kombination der einzelnen Basis-Lerner. Das einfachstes und wohl intuitivste Vorgehen ist demnach das „Majority Voting“ - wie in Gl. 4.19 dargestellt -, bei dem diejenige Klasse k von der Ensemblemethode ausgewählt wird, die unter den B Prädiktionen am häufigsten gewählt wurde.

$$\hat{f}_{vot}(\boldsymbol{x}) = \arg\max_{k \in \mathcal{Y}} \sum_{b=1}^{B} \mathbb{I}(f^b(\boldsymbol{x}) = k) \tag{4.19}$$

Zugleich sei hier aber auch noch angemerkt, dass Bagging nicht immer notwendigerweise zu einer Verbesserung der Prädiktionen führt (s. Scheipl et al. (2018, Kap. 4)). Für die ausführ-

lichere bzw. mathematische Beantwortung der Frage, weshalb Bagging i.A. funktioniert bzw. Varianz-reduzierende Wirkung hat, sei auf Zhou (2012, Kap. 3.4) verwiesen.
Wie bereits in Sektion 4.3 erwähnt wurde, kann Bagging auch genutzt werden, um Entscheidungsbäume hinsichtlich ihrer fehlenden Robustheit zu verbessern. Entsprechend werden also B Regressions-/Klassifikationsbäume auf die B Bootstrap-Trainingsdatensätze angepasst. Dabei werden die Bäume keinem Pruning unterzogen, sondern weisen folglich eine Vielzahl an Splits auf. Somit ist jedem einzelnen Basis-Baum eine niedriger Bias zu eigen, dafür aber eine hohe Varianz. Letzterem wird durch den anschließenden Aggregations-Schritt Rechnung getragen. So können schließlich hunderte oder gar tausende Entscheidungsbäume in einer einzelnen Prozedur kombiniert werden und eine deutliche Verbesserung in der Performance gegenüber einem einzelnen Baum zu Tage bringen.
Diesem Gedankengang folgend lässt sich festhalten, dass Bagging die Varianz eines Schätzers reduziert, letztlich aber den Bias erhöht. Entsprechend eignet sich Bagging am besten für instabile/hohe Varianz aufweisende Basis-Methoden, also insbesondere Entscheidungsbäume, aber auch neuronale Netzwerke und Variablenselektion (vorwärts, rückwärts oder stufenweise) innerhalb von Regression. [vgl. Scheipl et al. (2018, Kap. 4)] Des Weiterenzeigt sich der Vorteil, dass auf recht einfache Weise der Testfehler für den durch Bagging erzeugten, was als Out-of-Bag (OOB) Fehlerschätzung und sich anhand der Untersektion 8.3.1 des Anhangs nachvollziehen lässt.

Modifikation/Erweiterung des Baggings: Random Forests (RFs)

Wie bereits in den vorhergehenden Untersektionen verdeutlicht wurde, eignet sich Bagging insbesondere zur Anwendung auf instabile, aber approximativ unverzerrte ML-Methoden, um eine Reduktion der Varianz zu erzielen. Man setze nun erneut das Augenmerk auf Entscheidungsbäume, die sich als ideale Kandidaten für Bagging herauskristalisieren, nachdem diese komplexe Interaktionsstrukturen in Daten aufdecken können und bei entsprechender Tiefe einen relativ niedrigen Bias aufweisen. Zudem wirkt die im Bagging inbegriffene Kombinationsmethodik der Basis-Lerner schließlich der fehlenden Robustheit der einzelnen Bäume entgegen und bewirkt so die erwähnte Varianzreduktion. Die folgende Erläuterung orientiert sich zunächst an Hastie et al. (2008, Kap. 15).
Man stelle nun die Überlegung an, dass - nachdem jeder Baum, der durch die Bagging-Prozedur erzeugt wird, identisch verteilt (i.d.) ist - man im Mittel von B derartigen Bäumen dasselbe Ergebnis erwartet wie von jedem einzelnen dieser Bootstrap-Bäume. Somit ist auch der Bias des durch Bagging erzeugten Baumes gleich dem der individuellen Bootstrap-Bäume und es wird mit dieser Ensemblemethode letztlich lediglich eine Verbesserung im Sinne der Varianzreduktion angestrebt. Dies steht im Kontrast zum Boosting, bei dem Bäume auf adaptive Weise zur Reduktion des Bias erzeugt werden und somit nicht identisch verteilt sind.
In jedem Fall ergibt sich allgemein, dass ein Durchschnitt von B unabhängig und identisch

verteilten (u.i.v.) Variablen - mit jeweiliger Varianz σ^2 - dann eine Varianz von σ^2/B aufweist. Wenn nun die Variablen einfach identisch verteilt (i.d.) - nicht notwendiger Weise identisch, also u.i.v. - mit positiver paarweiser Korrelation ρ, dann errechnet sich die Varianz für den Durchschnitt folgendermaßen:

$$\mathbb{V}(f^B(\boldsymbol{x}))= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2, \tag{4.20}$$

mit ρ = *Korrelation zwischen Basis-Bäumen* $\mathrm{Corr}(T^b(\boldsymbol{x}), T^{b'}(\boldsymbol{x}))$

σ^2 = *Varianz eines Baumes* $\mathbb{V}(T^b(\boldsymbol{x}))$

Anhand von Gl. 4.20 wird somit ersichtlich, dass durch Erhöhung von B und damit der Anzahl an Bootstrap-Subsets der zweite Summand immer kleiner wird bzw. für eine gegen Unendlich strebenden Wert von B nichtig wird. Jedoch bleibt der erste Summand erhalten und damit limitiert die Stärke der Korrelation zwischen den einzelnen Basis-Bäumen die durch Bagging auszuschöpfenden Vorteile.
Dies stellt nun den Punkt dar, an dem „Random Forests“ (RFs) ansetzen. Entsprechend streben RFs eine Erhöhung der Varianzreduktion von Bagging durch Reduktion der Korrelation zwischen den einzelnen Basis-Bäumen an, ohne dabei die Varianz zu sehr anzuheben. Aus diesem Grund spricht man bei RFs auch von auf Bootstrap basierenden, unkorrelierten Bäumen. Dieses Ziel wird erreicht, indem während der Erstellung des Baumes an jedem Knoten eine zufällige Selektion der Input-Variablen erfolgt.
Die genannte Vorgehensweise hat eine einfache Begründung, die anhand von James et al. (2013) dargelegt werden kann: Angenommen es gibt einen starken Prädiktor innerhalb des Datensatzes und einige weitere, aber weniger einflussreiche. Das wird letztlich dazu führen, dass im Ensemble die meisten Basis-Bäume diesen Prädiktor im Wurzelknoten als Splitvariable auswählen werden. Demgemäß wird der Großteil der Bäume einen recht ähnlichen Aufbau zeigen und damit die Bäume innerhalb des Baggings hoch korreliert sein. Dies wiederum bedingt, dass auch durch das Mitteln über derartig korrelierte Bäume eine nicht allzu große Reduktion der Varianz erreicht werden kann. Stellt man nun allerdings pro Knoten eines jeden Basis-Baumes nur eine geringe Anzahl zufällig ausgewählter Prädiktoren zur Verfügung, sollte diese Problematik umgangen und die Idee von RFs final geklärt sein.
Bei spezifischer Betrachtung der Erzeugung eines Basis-Baumes basierend auf einem Bootstrap-Datensatzes wird also vor jeder Aufteilung eines Knotens die Anzahl von $m \leq P$ an Input-Variablen zufällig als Kandidaten für einen Split ausgewählt, wobei P der Gesamtzahl an Variablen im Datensatz entspricht. Scheipl et al. (2018, Kap. 4, S. 12) folgend wird üblicherweise bei Klassifikationsaufgaben $m = \lfloor \sqrt{P} \rfloor$, im Falle von Regression $m = \lfloor \frac{P}{3} \rfloor$ angeraten. Allerdings wird in der Praxis auch dieser Parameter von der jeweiligen Problemstellung und den gegebenen Daten abhängen, weshalb es sinnvoll ist m innerhalb des Hyperparameter-Tunings (s. Untersektion 3.3.2) zu bestimmen.

Es lässt sich also folgern, dass der Parameter m die Einbindung der Zufälligkeit steuert. Man beachte allerdings, dass die Zufälligkeit nur während des Prozesses der Feature-Wahl, nicht aber bei der Bestimmung des Splitpunktes des ausgewählten Features inbegriffen ist und dieser Schritt entsprechend nach wie vor deterministisch ist. Insgesamt wird dadurch also eine Reduktion der Korrelation durch Randomisierung in jedem einzelnen Konten erreicht. Gleichzeitig werden die Bäume T^b frei ausgeweitet, ohne vorzeitiges Stoppen oder Pruning, sodass die Diversität innerhalb des Ensembles vergrößert wird.
In diesem Rahmen ist es nun auch denkbar einen RF basierend auf den in Untersektion 4.3.2 beschriebenen cTrees zu konstruieren, die entsprechend als Basis-Lerner herangezogen werden. Wenngleich sich die eingehende Beschreibung Software-gestützter Implemnetierungen hierzu in Kap. 5 und 6 findet, sei als Besonderheit hervorgehoben, dass innerhalb des R-Paketes *party* von Hornik et al. (2019) - hingegen zu den meisten bisherigen Implementierungen - das Aggregationsschema des „Conditional Inference Forest's" (CIF oder cForest) nicht mit dem direkten Mitteln der Prädiktionen einhergeht. Stattdessen werden die Beobachtungsgewichte, welche von jedem der im RF inbegriffenen Bäume extrahiert werden, gemittelt. Was sich letztlich jedoch für jegliche Art des RF's wohl herauskristallisiert ist die Tatsache, dass eine einfache Interpretation - wie etwa bei einem Entscheidungsbaum - bei RFs nicht gegeben ist. Das Kombinieren vieler Basis-Bäume, sowie die Tatsache, dass jeder Baum auf einem anderen Subdatensatz basiert und noch dazu bei jedem Knoten nur ein zufällig ausgewählter Satz möglicher Prädiktoren bereitgestellt wird, macht diese schier unmöglich. Aus diesem Grund ordnet man RFs auch den s.g. „Blackbox-Methoden" zu. Dennoch kann man die Wichtigkeit bzw. die Einflussstärke der inbegriffenen Variablen innerhalb der RFs beurteilen und daher sei für spezifische Erläuterung der s.g. „Variable Importance" innerhalb des RF's auf Anhang 8.3.2 verwiesen. Nachdem dies jedoch eine allgemein anwendbare Methodik für die Zugänglichkeit zur Interpretation von Blackbox-Methoden darstellt, sei an dieser Stelle spezifisch Sektion 5.3 nahe gelegt.

4.4.3 (Componentwise Gradient) Boosting

Der Begriff „Boosting" ist der Familien von Algorithmen zuzuweisen, die in der Lage sind schwache (weak) zu starken (strong) Lernern zu wandeln bzw. zu verbessern (engl. „(to) boost"). Folglich ist Boosting ebenso den Ensemblemethoden zuzuordnen, wobei zum Ausdruck gebracht werden soll, dass hierdurch eine Möglichkeit gegeben ist, einen Lerner, der nur geringfügig bessere Performance aufzeigt, als sich durch zufälliges Raten ergeben würde, hin zu einem Lerner zu „boosten", der nahezu perfekte Performance aufzeigen könnte.
Die Entwicklung des Boostings lässt sich dabei gemäß Zhou (2012, S. 23) auf die theoretische Frage zurückführen, ob die zwei Komplexitätsklassen der schwach lernbaren und stark lernbaren Probleme (vgl. Kearns und Valiant, 1989) äquivalent sind. Diese Frage ist von fundamentaler Relevanz, da eine positive Antwort rückschließend bedeutet, dass jeder schwache

Lerner zu einem starken Lerner verbessert werden kann. Insbesondere ist dies von großer Bedeutung, da es in der Praxis relativ einfach ist einen schwachen Lerner hervorzubringen, wohingegen starke nur schwer erreicht werden können. So konnte Schapire (1990) zeigen, dass die Antwort tatsächlich positiv ausfällt und der entsprechende Beweis eine Konstruktion wie beispielsweise das Boosting ist.

Generelle Boosting-Prozedur (hin zu AdaBoost)

Die grundlegende Idee des Boostings ist allgemein gesprochen recht simpel und soll nun mithilfe von Zhou (2012, Kap. 2.1) zugänglich gemacht werden. Hierzu gehe man zunächst von einem binärem Klassifikationsproblem und einem gegebenen, schwachen Lerner aus, der sich auf jegliche Verteilung der Daten anwenden lässt. Entsprechend besteht nun das Ziel aus einer Klassenzuordnung zu „positiv“ und „negativ“. Die Trainingsinstanzen $\mathcal{T}$ werden u.i.v. aus der Verteilung $\mathcal{D}$ gezogen und man nehme an, dass sich $\mathcal{T}$ in drei disjunkte Mengen $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ untergliedert, wobei jede einen Anteil von $1/3$ der Verteilung zukommt. Zusätzlich gehe man von einem Lerner aus, dessen Klassenzuteilung zufälligem Raten gleichkommt und damit eine Klassifikationsfehler von 50 % aufweist.
Es wird nun das Ziel verfolgt - trotz zugrundeliegendem schwachen Lerner - möglichst akkurate Prädiktionen treffen zu können. Beispielsweise trifft nun der Klassifikator h_{12} nur für $\mathcal{X}_1, \mathcal{X}_2$ korrekte und entsprechend für $\mathcal{X}_3$ falsche Zuweisungen, womit ein Klassifikationsfehler von $1/3$ vorliegt. Letzteres ist nicht gewünscht und man versucht daher durch Boosting den Fehlern, die durch h_{12} gemacht werden, entgegenzutreten. Hierzu leite man eine Verteilung $\mathcal{P}'$ von $\mathcal{P}$ ab, durch die die Fehler von h_{12} noch deutlicher hervortreten, beispielsweise indem man mehr auf die Features von $\mathcal{X}_3$ fokussiert. Erneut wird nun also ein Klassifikator h_{13} auf Basis von $\mathcal{P}'$ trainiert, um die genannte Problematik anzugreifen. Angenommen h_{13} ist wiederum ein schwacher Lerner und treffe zwar korrekte Zuteilungen für $\mathcal{X}_1$ und $\mathcal{X}_3$, jedoch nicht für $\mathcal{X}_2$. Wenn nun h_{12} und h_{13} auf angemessene Weise kombiniert werden, so zeigt der entsprechende neue Klassifikator korrekte Zuteilungen in $\mathcal{X}_1$ und nur einige Fehler in $\mathcal{X}_2, \mathcal{X}_3$. Wiederum such man also eine Verteilung $\mathcal{P}''$, welcher die Fehler des kombinierten Klassifikators deutlich hervorbringt und trainiere h_{23}, sodass dieser korrekte Klassifikationen in $\mathcal{X}_2, \mathcal{X}_3$ ermöglicht. Durch sinnvolle Kombination von h_{12}, h_{13}, h_{23} sollte man einen im engeren Sinne „perfekten“ Klassifikator erlangen, nachdem in jedem Raum $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ zumindest zwei Lerner korrekte Zuteilungen treffen.
Dieses Beispiel ist zwar sehr vereinfacht und idealisiert dargestellt, verdeutlicht jedoch gut die grundlegende Funktionsweise von Boosting-Methoden: ein Satz von Lernern wird sequentiell trainiert und für zukünftige Prädiktionen kombiniert, wobei sich die später entwickelten Lerner jeweils mehr auf die Fehler der vorhergehenden konzentrieren. Wie bereits erwähnt war damit der erste Boosting-Algorithmus kein Algorithmus im engeren Sinne, sondern diente zur theoretischen Beweisführung. Allerdings verhalf dieser Ansatz Freund und Schapire (1997)

direkt zur Generierung des ersten Algorithmus, der heute als adaptives Boosting oder kurz unter „AdaBoost“ bekannt ist.
Insbesondere da AdaBoost einen historischen Meilenstein in der Geschichte des ML's markiert und sich hieran orientiert viele weitere Boosting-Methoden entwickelten, kann diese Prozedur als Basis nachfolgender Erklärungen genutzt werden und bei mangelnder Kenntnis Untersektion 8.3.3 des Anhangs herangezogen werden. Darauf aufbauend soll im Folgenden nun auf eine Art Reinterpretation bzw. auf eine Generalisierung des AdaBoostings eingegangen werden und in diesem Zuge das s.g. „Gradient Boosting“ (GB) bzw. ausweitend das „Componentwise Gradient Boosting“ (CGB) Erklärung finden.
Hierzu sollte sich zunächst noch in Anlehnung an Mayr et al. (2014, S. 6) vor Augen geführt werden, dass AdaBoost sich als Blackbox-Methode deklariern lässt und damit konträr zu einem statischen Modell steht, welches das Ziel verfolgt einen quantifizierbaren Zusammenhang zwischen ein oder mehreren beobachteten Einflussgrößen $\boldsymbol{x}$ und dem erwarteten Response $\mathbb{E}(Y)$ herzustellen. Dies geschieht mithilfe einer interpretierbaren Funktion $\mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}) = f(\boldsymbol{x})$, wobei im Falle mehrerer Prädiktoren die einzelnen Effekte zumeist summiert und somit ein additives Modell der nachstehenden Form - mit β_0 als Intercept - bilden:

$$f(\boldsymbol{x}) = \beta_0 + h_1(\boldsymbol{x}_1) + \cdots + h_P(\boldsymbol{x}_P) \tag{4.21}$$

Die zugehörige Modellklasse lässt sich als den generalisierten additiven Modellen (GAM) zugehörend bestimmen. Innerhalb dieser Klasse wird das Ziel verfolgt den erwarteten Wert einer Responsevariable gegeben den in Form einer Link-Funktion $g(\cdot)$ bereitgestellten, beobachteten Prädiktoren zu modellieren:

$$g(\mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}))= \beta_0 + \sum_{p=1}^{P} h_p(\boldsymbol{x}_p) \tag{4.22}$$

Die Kernaussage zu der Friedman et al. (2000) damit kommen wollen ist die folgende: der AdaBoost Algorithmus in Kombination mit Regressions-artigen Basis-Lernern - beispielsweise lineare Modelle oder Splines - passen letztlich ein GAM für einen dichotomen Outcome über den exponentiellen Verlust auf schrittweise Art an.

(Componentwise) Gradient Boosting

Mit vorangehender Einordnung des AdaBoosting Algorithmus in den Kontext von GAMs sollte hier nun ein sanfter Einstieg in Gradient Boosting (GB) bzw. der Methodik Gradient Boosting Machines (GBM) gegeben sein. Die Erklärungen hierzu richten sich zunächst im Wesentlichen nach Friedman (2001), Scheipl et al. (2018, Kap. 7) sowie Mayr et al. (2014). In diesem Rahmen weitete Friedman (2001) die statistische Sichtweise von Boosting aus, indem er einen Boosting Algorithmus vorstellte, welcher das empirische Risiko über das Gradien-

tenverfahren (Steepest Gradient Descent) im Funktionenraum optimiert. Entsprechend kann das Optimierungsproblem für die Schätzung der Regressionsfunktion $f(\cdot)$ eines statistischen Modells über eine Verlustfunktion dann wie in Gl. 4.23 ausgedrückt werden.

$$\hat{f}(\cdot) = \underset{\hat{f(\cdot)}}{\arg\min}\Big\{\mathbb{E}_{\boldsymbol{Y},\boldsymbol{X}}\left[L(\boldsymbol{Y}, f(\boldsymbol{X}))\right]\Big\} \tag{4.23}$$

mit $L(\cdot,\cdot)$ = *Verlustfunktion*

Wiederum gilt es also für ein gegebenes Lernset von Beobachtungen $\{(y_1, \boldsymbol{x}_1), \ldots, (y_n, \boldsymbol{x}_n)\}$ das empirische Risiko zu minimieren und damit die nachstehende Gl. 4.24 zum Einsatz zu bringen.

$$\hat{f}(\cdot) = \underset{\hat{f(\cdot)}}{\arg\min}\{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i))\} \tag{4.24}$$

Die zugrundeliegende Idee beim GB ist nun nicht wie beim AdaBoosting die Basis-Lerner an die neu-gewichteten Beobachtungen anzupassen, sondern an den negativen Gradienten-Vektor $\boldsymbol{u}^{[m]}$ der Verlustfunktion $L(y, \hat{f}(x))$, welcher in der vorhergehenden Iteration $m-1$ evaluiert wurde. Demgemäß ergibt sich dann nachfolgende Gleichung zur Errechnung des negativen Gradienten-Vektors $\boldsymbol{u}$ während Iteration m.

$$\boldsymbol{u}^{[m]} = \left(u_i^{[m]}\right)_{i=1,\ldots,n} = \Big(-\frac{\partial}{\partial} L(y_i, f)\big|_{f=\hat{f}^{[m-1]}(\cdot)}\Big)_{i=1,\ldots,n} \tag{4.25}$$

Befindet man sich im Klassifikationskontext, so bietet es sich gemäß Natekin und Knoll (2013, Kap. 3) an auf die Bernoulli-Verlustfunktion oder den exponentiellen Verlust zurückzugreifen, wie letzterer im Algorithmus von AdaBoost Verwendung findet. Prinzipiell kann die Gradient Boosting Prozedur für die Optimierung jeder Verlust-Funktion herangezogen werden, solange diese konvex und differenzierbar ist. Demgemäß besteht insbesondere keine Beschränkung auf statistische Verteilungen, die der Exponentialfamilie angehören, wie dies für klassische GAMs der Fall sein muss. Unter genauerer Betrachtung führt so z.B. die Nutzung des L_2-Verlusts $L(y, f(\cdot)) = 0.5 \cdot (y - f(\cdot))^2$ zur einfachen Neuanpassung der Residuen $y - f(\cdot)$. Anders ausgedrückt wird in jeder der m Boosting-Iterationen der Basis-Lerner an die Fehler $y - f(\cdot)^{[m-1]}$ angepasst, die in der vorhergehenden Iteration gemacht wurden. Somit kann auch wieder Bezug zu AdaBoost genommen werden: beide Algorithmen zielen auf die Verbesserung der Performance durch Verschiebung des Fokus hin zu problematischen Beobachtungen ab. Lediglich die Vorgehensweise unterscheidet sich. AdaBoost konzentriert sich hierbei auf die Aufgewichtung fehlklassifizierter Beobachtungen, wohingegen GBMs derartige Probleme anhand der vorhergehend errechneten (Pseudo-)Residuen ausmachen.

Im Rahmen des CGB's bietet sich als Basis-Lerner jegliche Regressionstechnik - weshalb man dann auch von Modell-basiertem Boosting spricht - an, wobei die einfachste Form durch das klassische lineare Modell $h(\boldsymbol{x}) = \boldsymbol{x}^T\beta$ gegeben ist. Ebenso kann es auch sinnvoll sein nicht-

lineare Effekt auf den Response zu modellieren und in diesem Zuge Splines heranzuziehen. So brachten Bühlmann und Yu (2003) Smoothing Splines hervor die Komponenten-weise innerhalb der Iterationen des GB's als Basis-Lerner angepasst werden. Damit entwickelte sich die Idee des s.g. „Component-wise Gradient Boosting's" (CGB), wobei dies spezifisch dadurch gekennzeichnet ist, dass die einzelnen Prädiktoren separat als Basis-Lerner angepasst werden, also $h_j(\cdot)$, $j = 1, \ldots, P$. Normalerweise ist hierbei jeder Basis-Lerner $h_j(\cdot)$ einer Komponente $\boldsymbol{x}_j$ aus $\boldsymbol{X}$ zugehörig, wobei dann in jeder Iteration der Boosting-Prozedur nur ein kleiner Teil der angepassten Basis-Lerner mit bester/hoher Performance zu dem derzeit bestehenden additiven Prädiktor hinzugefügt werden. Entsprechend ist damit ein impliziter Prozess der Variablenselektion integriert. Derartiges CGB wird auch als "glmboost" bezeichnet, nachdem hierdurch die Schätzungen der Kovariableneffekte bereitgestellt sind. Jedoch sind hierbei keine gültigen Standardfehler zu den einzelnen Koeffizienten wie im GLM gegeben, sodass keine Konfidenzinterfalle errechnet oder Tests durchgeführt werden können. Dieser Problematik lässt sich (teilweise) durch Bootstrap-Inferenz beheben. Ausweitend stellen „gamboost" Modelle dann also GB auf Basis von Spline-Basis-Lernern bereit.
Hervorhebend sei für das CGB festgehalten, dass sich diese Methodik - im Gegensatz zu den meisten Standardmethoden - auch anwenden lässt, wenn hochdimensionale Daten mit einer höheren Anzahl an Prädiktoren als Beobachtungen ($P > N$) vorliegen. Neben der relativ hohen Robustheit gegenüber Multikollinearität vermerken Mayr et al. (2014, S.9) Folgendes: aufgrund der relativ kleinen Update-Schritte in Kombination mit s.g.„Early Stopping" wird eine Art von Shrinkage der geschätzten Effekte in den Schätzprozess eingebunden, was letztlich ähnlich zur penalisierten Regressionsansätzen wie beispielsweise dem LASSO ist. Somit wird also eine Reduktion der Varianz der Schätzungen bewirkt, was dann zur Erhöhung der Stabilität und Genauigkeit der Prädiktionen führt.
Nachdem eine große Bandbreite an Boosting-Prozeduren zur Auswahl steht, soll die Beschreibung bis hierher relativ allgemein gehalten bleiben und bei entsprechendem Wunsch nach vertieften Erklärungen insbesondere die zu Anfang dieser Untersektion genannten Quellen herangezogen werden oder etwa auf Hastie et al. (2008, Kap. 10) verwiesen sein. Namentliche Erwähnung soll lediglich noch das Extreme Gradient Boosting (XGBoost) finden, dessen Konzept auf dem GB basiert, jedoch auf einer stärker regulierten Modellformularisierung beruht, sodass gemäß den Autoren Overfitting besser entgegengesteuert werden kann und letztlich eine höhere Performance erzielt wird. Gleichzeitig hat der zugrundeliegende Algorithmus deutlich schnellere Computationszeiten zu vergleichbaren Methodiken inne.
Im Folgenden soll sich nun vom Boosting abgewandt werden und stattdessen auf die im vorhergehenden Absatz angesprochenen penalisierten Regressionsansätze im Kontext des ML's eingegangen werden, wobei dazu spezifisch das Logit-Modell in den Fokus gesetzt sei.

4.5 Logistische Regression (mit Penalisierung)

Im Feld des ML's kann nicht nur auf neuerartige Methoden zurückgegriffen werden, sondern ebenso auf alt bewährte Regressionsansätze die sich im Bereich der Statistik breit gefächert etabliert haben. So kann also neben den bisher nur non-parametrischen ML-Methoden, auch auf parametrische zurückgegriffen werden und damit im Klassifikationskontext insbesondere auf die logistische Regression. Die Benennung der Methodik mag zunächst verwirrend erscheinen, nachdem es sich um eine Technik zur Klassifikation und nicht der Regression handelt. Logistische „Regression" leitet sich vielmehr von der Tatsache ab, dass dem Feature-Raum ein lineares Modell zugrunde gelegt wird. Häufig spricht man allerdings auch vom s.g. Logit-Modell, nachdem die Modellierung auf der Logit-Funktion beruht.
Da prinzipiell durch die Nutzung von ML-Methodiken eine Art automatisierte Variablenselektion inbegriffen ist, gilt es im Rahmen der logistischen Regression LASSO, Ridge, sowie das Elastic Net als Erweiterungen vorzustellen und damit starke Techniken der Feature-Auswahl. Genauere Ausführungen zu den einzelnen genannten Bausteinen sollen nachfolgend insbesondere mithilfe von Hastie et al. (2008, Kap.4.4) erbracht werden.

4.5.1 Das Logit-Modell

Allgemein gesprochen lässt sich das genannte Logit-Modell der Familie der generalisierten linearen Modelle (GLM) zuordnen. Insbesondere liegt hierbei die Annahme zugrunde, dass innerhalb eines methodisch einheitlichen Rahmens der Effekt der Einflussvariablen in Form eines linearen Prädiktors realisiert werden kann. Dabei ist es für GLM's nicht erforderlich, dass die Zielgröße als metrische Variable vorliegt, sondern es können wie im Falle des Logit-Modells auch binäre Variablen modelliert werden. Die Einordnung des Logit-Modells in den Kontext von GLM's, sowie die Schätzung der Modellparameter stellt den Schwerpunkt dieser Untersektion dar und soll in Anlehnung an Fahrmeir et al. (2009, S. 189 ff.) dargelegt werden. Das Logit-Modell bietet nunmehr den Vorteil, dass die Effektstärke der einzelnen Einflussvariablen auf die Zielgröße identifiziert werden kann. Für genaue Ausführungen wird im weiteren Verlauf von binären Klassifikationsaufgaben ausgegangen, wobei für die Modellierung einer mehrkategoriellen Zielvariable auf die multinomiale logistische Regression verwiesen sei.

$$\pi_i = P(y_i = 1) = h(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_P x_{iP}) \tag{4.26}$$

Ausgehend von einer durch 0 und 1 kodierten Zielvariable bietet es sich im Rahmen der Regressionsanalyse an mit Wahrscheinlichkeitsaussagen zu arbeiten und damit den Wertebereich der binären Größe auf dem Intervall $[0, 1]$ einzuhalten. Die Verwendung dieses Modells begründet sich also v.a. in der Tatsache, dass es bei anderen Modellen - beispielsweise einem linearen Modell - möglich ist, dass die Wahrscheinlichkeitsfunktion $\pi_i = \mathbb{P}(y_i = 1)$ Werte kleiner 0 oder größer 1 annehmen kann. Stattdessen wird also ein derartiges Modell gesucht, sodass

der Wertebereich der Funktion h in vorhergehender Gleichung 4.26 im Intervall [0,1] liegt und gleichzeitig eine streng monoton wachsende Funktion darstellt. Eine Funktion, welche die gewünschten Kriterien für die Zielvariable $y_i \in \{0, 1\}$ erfüllt, ist die logistische Verteilungsfunktion. Das sich ergebende Modell wird als Logit-Modell bezeichnet und lässt sich wie in Gl. 4.27 darstellen. Dabei gilt zu beachten, dass die Schreibweise derart gewählt ist, dass der Vektor $\boldsymbol{\beta}$ den Intercept einschließt - entsprechend $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_P)$ -, weshalb jede Beobachtung $\boldsymbol{x}_i$ hier um den konstanten Term 1 zu erweitern ist, um dem Offset Rechnung zu tragen. Es wird nun also von einem Vektor der Form $\boldsymbol{x}_i = (1, x_{i1}, \ldots, x_{iP})^T$ ausgegangen und q als Erweiterung des Index p um 1 aufgefasst, sodass $x_{i0} = 1$ für $i = 1, \ldots, N$.

$$\begin{aligned} P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}) = \pi_i = h(\boldsymbol{x}_i^T\boldsymbol{\beta}) &= \frac{\exp(\eta_i)}{1 + \exp(\eta_i)} \\ &= [1 + \exp(-\eta_i)]^{-1} \\ &= [1 + \exp(-\sum_{q=0}^{P} x_{iq}\beta_q)]^{-1} \end{aligned} \tag{4.27}$$

Verteilungsannahme:	$y_i\|\pi_i \sim \text{Bin}(\pi_i)$, wobei die Dichte der Binomialverteilung für ein Ereignis $k = 0, \ldots, n$ gegeben ist durch: $f(k) = \binom{n}{k} \cdot \pi^k \cdot (1 - \pi)^{n-k}$
Strukturannahme:	$\eta_i = \boldsymbol{x}_i^T\boldsymbol{\beta}$
Link-Funktion:	$\pi_i = h(\eta_i)$ bzw. $\eta_i = g(\pi_i)$, wobei $h(\eta_i)$ - wie in Gl. 4.27 dargestellt - definiert ist: $h(\eta_i) = \frac{\exp(\eta_i)}{1+\exp(\eta_i)} \Leftrightarrow g(\pi_i) = \log(\frac{\pi_i}{1-\pi_i})$

Tab. 4.2: Annahmen im GLM mit Logit-Link

Zusammenfassend liegen dem GLM mit Logit-Link also die in Tab. 4.2 gelisteten Annahmen zugrunde. Für genauere Ausführungen hierzu, ebenso wie für das ausführlich dargestellte Vorgehen zur Schätzung des Modells sei auf Untersektion 8.3.4 des Anhangs verwiesen. Letzteres beschreibt das konkrete Vorgehen hinsichtlich der Parameterschätzung, um schließlich die interessierenden Ergebnisse aus dem derartigen Regressionsmodell ziehen zu können. Hierbei wird nicht wie im LM auf die gewöhnliche Methode der kleinsten Quadrate („Ordinary Least Squares“ = OLS) zurückgegriffen, sondern die Maximum-Likelihood Schätzung (MLE) genutzt. Zur Berechnung der Effektschätzer $\hat{\boldsymbol{\beta}}$ wird zunächst die Likelihood als Produkt der einzelnen Dichten bzw. - aufgrund einfacherer Handhabung - die logarithmierte Likelihood gebildet. Letztere kann schließlich wie in Gl. 4.28 dargestellt werden.

$$\begin{aligned} l(\boldsymbol{\beta}) = \sum_{i=1}^{N} l_i(\boldsymbol{\beta}) &= \sum_{i=1}^{N} y_i \cdot \log\Big(\frac{\pi_i}{1 - \pi_i}\Big) + \log(1 - \pi_i), \\ \textit{wobei } \pi_i = \frac{\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_i^T\boldsymbol{\beta})} &\Longleftrightarrow \log\Big(\frac{\pi_i}{1 - \pi_i}\Big) = \boldsymbol{x}_i^T\boldsymbol{\beta} = \eta_i \end{aligned} \tag{4.28}$$

Um diese log-Likelihood zu maximieren, wird deren Ableitung - bezeichnet als s.g. Score-Funktion - gleich 0 gesetzt. Schließlich lassen sich die Parameterschätzer in $\hat{\boldsymbol{\beta}}$ über iterative Algorithmen wie den Newton-Raphson-Algorithmus ermitteln, wobei diese aufgrund der konkaven log-Likelihood typischer Weise konvergieren. Jedoch stößt man insbesondere in Anbetracht dessen, dass im Kontext des ML's zumeist eine Vielzahl an potentiellen Einflussgrößen mit ggf. verhältnismäßig geringer Anzahl an Beobachtungen bzw. Trainingsdaten einer Lernmethodik übergeben werden, unter Zuhilfenahme des Logit-Modells an seine Grenzen. Gemäß Schimek (2003) ist nämlich in Form dieses Settings eine der grundlegenden Annahmen verletzt, dass die Zahl der Prädiktoren deutlich kleiner zu sein hat als die Zahl der Beobachtungen. Ein Verstoß führt zu der Konsequenz, dass es mehr Unbekannte als Gleichungssysteme gibt und folglich unendlich viele Lösungen existieren, was ein schlecht konditioniertes Problem darstellt. Darüber hinaus lässt sich einer weitere Problematik im Falle der Klassifikation durch die niedrige Power der Diskriminierung aufgrund der hohen Varianz bei der Anpassung des Modells an die Daten erkennen. Den aufgeführten Problemstellungen kann durch Penalisierung entgegengetreten werden und stellt aus diesem Grund den Fokus der nachfolgenden Untersektion dar.

4.5.2 Regularisierung mittels Shrinkage-Methoden

Shrinkage-Methoden entwickelten sich im Kontext der Regressionsanalyse auf Basis der Idee Overfitting zu vermeiden, indem starke Schwankungen der geschätzten Parameter in Hinsicht auf die angepasste Funktion bestraft werden und damit die Funktion an sich geglättet wird. Gleichzeitig erfolgt so auch eine Variablenselektion, was entsprechend eine Dimensionenreduktion bedingt. Die nachfolgenden Beschreibungen zu den Methoden des Shrinkage's - was sich als „Schrumpfung, Rückgang, Schwund“ (der Regressionskoeffizienten) übersetzt lässt - fußen auf Schimek (2003, Kap. 4), Pereira et al. (2016, Kap. 2), sowie Pekhimenko (2006, Kap. 3) und sind als eine Art der Regularisierung zu verstehen. Ein kurzer Überblick zu Regularisierungsmechanismen findet sich im Anhang bei Untersektion 8.3.4.
Als wichtigste Mechanismen der Regularisierung mittels Shrinkage sind wohl LASSO und Ridge aufzuführen. Der Name ersterer Technik leitet sich von englischsprachigen Bezeichnung „Least Absolute Shrinkage and Selection Operator“ ab. Noch einmal zusammenfasst stellt das Hauptziel der Penalisierung nun die verbesserte prädiktive Performance bei Anwendung auf einen neuen Datensatz durch balancierte Modellanpassung an die Trainingsdaten und stabile Schätzung der Parameter dar. Basis dessen bildet die log-Likelihood aus Gl. 4.28.

$$l^{\star}(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - \lambda \cdot J(\boldsymbol{\beta}^{(p)}), \quad \text{mit} \;\; J(\boldsymbol{\beta}^{(p)}) = \sum_{p=1}^{P} \gamma_p \psi(\boldsymbol{\beta}_p), \; \gamma_p > 0 \tag{4.29}$$

Allgemein gesprochen kann diese log-Likelihood nun um den Bestrafungsterm $\lambda \cdot J(\cdot)$ erweitert werden, sodass die regularisierte Likelihood $l^{\star}(\boldsymbol{\beta})$ die in Gl. 4.29 dargestellte Form annimmt.

Dabei gilt zu beachten, dass sich die Penalisierung nur auf die Regressionskoeffizienten, nicht aber den Intercept auswirkt. Gleichzeitig wird die s.g. Penality $\psi(\cdot)$ i.A. derart gewählt, dass diese symmetrisch und steigend auf dem Intervall $[0; \infty[$ ist. Darüber hinaus kann diese sowohl in Form einer konvexen oder nicht-konvexen, sowie glatten oder nicht-glatten Funktion zutage treten. Der Komplexitätsparameter λ steuert hierbei die Größe der Koeffizient bzw. den relativen Einfluss des Bestrafungsterms auf die Regressionskoeffizienten, womit entsprechend ein großer Wert von λ mit einer Verringerung der β-Parameter einhergeht.
Um die penalisierte log-Likelihood aus Gl. 4.29 zu maximieren, werden die einzelnen Derivate gleich Null gesetzt, sodass sich die nachstehende Gl. 4.30 ergibt.

$$\frac{\partial l^{\star}(\boldsymbol{\beta})}{\partial \beta_0} = 0 \quad \text{und} \quad \frac{\partial l^{\star}(\boldsymbol{\beta})}{\partial \beta_p} = 0, \quad \text{für } p = 1, \ldots, P \tag{4.30}$$

So ergeben sich die penalisierten Likelihood-Gleichungen wie in Gl. 4.31, wobei $\mathbb{1}_N$ einen N-dimensionalen Vektor bestehend aus Einsen darstellt.

$$\mathbb{1}_N^T(\boldsymbol{y} - \boldsymbol{\pi}) = 0 \quad \text{und} \quad \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{\pi}) = 0 \tag{4.31}$$

Wie üblich für logistische Regression sind die Gleichungssysteme in Gl. 4.31 non-linear und so kann π_i durch eine Taylorentwicklung erster Ordnung angenähert werden, um darauffolgende Rechenschritte einleiten zu können. Diese lassen sich anhand der Untersektion 8.3.4 des Anhangs nachvollziehen. Wiederum können final dann iterative Algorithmen genutzt werden, um die interessierenden Koeffizienten des penalisierten logistischen Modells zu ermitteln.
Bisher ungeklärt geblieben ist die Frage nach der Wahl des λ-Parameters, nachdem dieser maßgeblich entscheidend ist hinsichtlich der Stärke der Regularisierung. Schimek (2003) zufolge befindet man sich hierbei in derselben Situation wie im Falle der Nutzung von Smoothing-Verfahren innerhalb der Regressionsanalyse, wobei jedoch aufgrund mangelnder empirischer Untermauerung die Erfahrungen bezüglich der Glättungstechniken nicht eins zu eins übertragen werden sollten. Entsprechend ist λ aber auch hier als Glättungsparameter zu verstehen, dessen Festsetzung innerhalb der logistischen Regression Empfehlungen zufolge auf Basis von CV oder dem AIC getroffen werden sollten. Eine halb-automatische Methodik für die Bestimmung von λ innerhalb der Ridge-Regression findet sich bei Cule und de Iorio (2012). Für weitere Ausführungen kann bei allen genannten Quellen auf einen kurzer Anklang bzgl. der Schätzung von λ verwiesen werden, was an dieser Stelle nun aber nicht weiter vertieft wird. Stattdessen soll nachfolgend im Wesentlichen auf die zwei Penalisierungsmethoden Ridge und LASSO, sowie auch auf das Elastic Net eingegangen werden, die sich jeweils durch Spezifizierung des Bestrafungsterms $J(\boldsymbol{\beta}^{(p)})$ aus Gl. 4.29 ergeben.

Ridge (L_2-Regularisierung)

Greift man nun zunächst auf quadratische Regularisierung/Verlustfunktion (L_2) zurück und bringt in diesem Kontext beispielsweise die Ridge-Methode zum Einsatz, so gilt es die log-Likelihood um den in Gl. 4.32 dargestellten Bestrafungsterm zu erweitern.

$$J_R(\boldsymbol{\beta}^{(p)}) = ||\boldsymbol{\beta}||_2 = \sum_{p=1}^{P} \beta_p^2 \tag{4.32}$$

Die Nutzung einer Penalisierung dieser Form führt zu einem Klassifikator mit kleineren Werten der Gewichte - womit sich der alternativ verwendete Begriff „Weight Decay“ Methode begründen lässt - und zumeist besserer Fähigkeit zur Generalisierung. Offensichtlich schrumpft der inkludierte Ridge-Bestrafungsterm die Koeffizienten gegen 0, jedoch wird aufgrund der quadratischen Form keiner der β-Parameter gänzlich auf 0 gesetzt, sodass kein Ausschluss irrelevanter Prädiktoren und damit keine Variablenselektion erfolgt. Zwar kann die Vorgehensweise so gewählt werden, dass Gewichte, deren Größe geringer ist als ein festgesetzter Threshold, als redundant erachtet werden und das Entfernen zugehöriger unabhängiger Größen aus dem Modell mit sich führen. Allerdings empfiehlt es sich zur Zusammenfassung der Schritte der Penalisierung und gleichzeitiger Exklusion irrelevanter Einflussgrößen auf eine L_1-Penalisierung zurückzugreifen, wobei eine derartige anhand von LASSO vorgestellt wird.

LASSO (L_1-Regularisierung)

Unter Nutzung der L_1-Penalisierung bzw. dem Shrinkage auf Basis der L_1-Norm, wie dies u.a. für LASSO der Fall ist, kann die Problematik der Ridge-Regression umgangen werden und entsprechend die Anzahl der Prädiktoren im finalen Modell herabgesetzt werden. Somit beinhaltet LASSO - insofern λ ausreichend groß ist - Variablenselektion und Shrinkage in Einem. Der zugehörige Bestrafungsterm gestaltet sich entsprechend Gl. 4.33. Pereira et al. (2016) zufolge kann davon ausgegangen werden, dass - falls nur eine kleine Zahl der Prädiktoren substanzielle Koeffizienten aufweist - mittels LASSO eine höhere Prädiktionsgenauigkeit erzielt werden kann. Dagegen erwartete man unter dem Sachverhalt nahezu identischer Koeffizienten aller Prädiktoren eine bessere Performance durch Nutzung der Ridge-Regression. In diesem Sinne wird CV zur Bestimmung der besseren Performance-Leistung bei Gegenüberstellung der Ridge- und LASSO-Penalisierung - angewandt auf die fokussierten Datensätze - empfohlen.

$$J_L(\boldsymbol{\beta}^{(p)}) = ||\boldsymbol{\beta}||_1 = \sum_{p=1}^{P} |\beta_p| \tag{4.33}$$

Offensichtlich bietet LASSO gegenüber Ridge also insbesondere den Vorteil einfachere und damit auch leichter zu interpretierende Regressionsmodelle hervorzubringen. Jedoch ist als Schwäche der LASSO-Regression v.a. das Unvermögen hinsichtlich gruppierter Auswahl zu

nennen. Genauer gesagt scheitert LASSO meist beim Vorliegen einer Gruppe von Variablen mit paarweiser hoher Korrelation, da zumeist nur eine Gruppe der Variable ausgewählt wird, wohingegen die anderen ignoriert werden. Zur Vertiefung dieser Thematik sei insbesondere auf Zou und Hastie (2005, Kap. 2.3) hingewiesen. Einen sinnvollen Ausweg aus dieser Problematik ist durch das s.g. Elastic Net bereitgestellt, welches nachfolgend kurze Erläuterung findet.

Elastic Net

Letztlich versucht das Elastic Net von Zou und Hastie (2005) die Vorteile von Ridge und LASSO miteinander zu vereinen. Dabei setzt sich der Bestrafungsterm aus einem Kompromiss zwischen eben diesen beiden Shrinkage-Methoden zusammen, was anhand von Gl. 4.34 verdeutlicht werden soll. Demgemäß befähigt das Elastic Net zu automatischer Variablenselektion und kontinuierlichem Shrinkage, kann jedoch auch die Gruppen korrelierter Variablen wählen. Namensgebend ist hierbei die Beschreibung der Begründer des Elastic Net's, die von einem „strechable fishing net that retains 'all the big fish'" (Zou und Hastie, 2005, S. 302) sprechen und somit eine verbessertes Shrinkage - verglichen mit Ridge und LASSO - bereitzustellen versuchen.

$$\begin{aligned} \ell_E^{\star}(\boldsymbol{\beta}) &= \ell(\boldsymbol{\beta}) - \Big[\lambda_1 \cdot J_L(\boldsymbol{\beta}_p) + \lambda_2 \cdot J_R(\boldsymbol{\beta}_p)\Big] \\ &= \ell(\boldsymbol{\beta}) - \lambda \cdot \Big[\alpha \cdot J_L(\boldsymbol{\beta}_p) + (1-\alpha) \cdot J_R(\boldsymbol{\beta}_p)\Big] \\ &= \ell(\boldsymbol{\beta}) - \lambda \cdot J_E(\boldsymbol{\beta}_p), \end{aligned} \tag{4.34}$$

mit $\lambda = \lambda_1 + \lambda_2$ *und* $\alpha = \lambda_2/(\lambda_1 + \lambda_2)$.

Hierbei stellt $J_E(\boldsymbol{\beta}_p)$ die Bestrafungsfunktion des Elastic Net's dar, welche sich als konvexe Kombination aus LASSO- und Ridge-Penality ergibt. Für den Fall, dass $\alpha = 0$ gewählt wird, vereinfacht sich das Elastic Net zur Ridge-Regression, sowie sich andererseits für $\alpha = 1$ die LASSO-Regression offenbart. Dabei stellt die L_1-Regularisierung ein sparsames Modell sicher, wohingegen durch den quadratischen Teil der Bestrafungsfunktion der Limitation der Anzahl von ausgewählten Variablen gegengesteuert wird, der Gruppierungs-Effekt unterstützt wird, sowie eine Stabilisation des L_1-Regularisierungspfades erreicht wird. Nachdem α den Anteil der L_1- bzw. L_2-Regularisierung innerhalb der Bestrafungsterms des Elastic Net's steuert, spricht man bei diesem Parameter auch vom „Mixing Parameter".
Dieser Ansatz bietet sich insbesondere dann an, wenn die Anzahl der Prädiktoren deutlich größer ist, als die Zahl der Beobachtungen. Für alles Weitere und insbesondere die Schätzung der β-Parameter - wobei dieses Optimierungsproblem letztlich äquivalent ist zu dem der LASSO- bzw. Ridge-Regression - sei auf Zou und Hastie (2005) verwiesen. Dazu sei angemerkt, dass sich die inbegriffenen Beschreibungen auf die gewöhnliche lineare Regression beziehen und nicht auf den hier im weitesten verwendete GLM mit Logit-Link.

5 Software-Nutzung und Entwicklung einer Shiny App

Im Rahmen der Auswertungen als Teil dieser Arbeit offenbarte sich die Komplexität von Benchmark-Studien, sowie die dadurch gegebene Notwendigkeit ausweitender Software-Nutzung. Hierzu kann nun insbesondere das *mlr* Paket genannt werden, welches eine Vielzahl notwendiger Funktionalitäten im Bereich des ML's bereitstellt. Auf derartige Nutzungsmöglichkeiten soll in nachgehender Sektion 5.1 im Hinblick auf ausgewählte Implementierungen eingegangen werden und dadurch zugleich ein Eindruck über das *mlr* Paket selbst und einiger dessen Funktion vermittelt werden. Mit Blick auf Benchmark-Studien - unter Nutzung des genannten *mlr* Paketes - kam im Verlauf dieser Arbeit aufgrund der komplexen Strukturen der Gedanke zur Entwicklung von Visualisierungsmöglichkeiten auf. Damit war die Basis für die Ausarbeitung einer s.g. Shiny App gelegt. Diese baut auf den im Rahmen dieser Arbeit getätigten Analysen auf und soll durch Verallgemeinerung derart implementiert sein, dass diese im breiten Umfang genutzt werden kann. Genauer wird hierauf in Sektion 5.2 eingegangen, doch sollen durch Sektion 5.1 bereits erste Ansatzpunkte zugänglich gemacht werden. Während der Entwicklung der Shiny App etablierte sich auch die Implementierung derartiger Methoden, die generell einen Zugang zur Interpretierbarkeit von ML-Methodiken bereitstellen sollen. In diesem Zusammenhang lässt sich insbesondere das *iml* Paket nennen, welches in Sektion 5.3 Ansprach findet und insbesondere auch einen Teil der hervorgebrachten Shiny App darstellt.

5.1 Machine Learning in R (mlr)

In der Open Source Software R (Version 3.5.1) - bereitgestellt durch R Core Team (2018) - können statistische Methoden mithilfe von bereits implementierten Funktionen bewerkstelligt werden, welche innerhalb von s.g. Paketen (Packages) abrufbar sind. Insbesondere im Hinblick auf maschinelle Lernalgorithmen steht hierbei eine breite Vielzahl an Paketen zur Verfügung. Als Problematik führt Bischl et al. (2016, Kap. *Machine Learning in R - Introduction*) hierzu die Tatsache auf, dass R keine standardisierte Schnittstelle für all diese Algorithmen bietet. Infolgedessen ist es erforderlich bei nicht-trivialen Experimenten lange, komplexe und zumeist fehlerbehaftete Wrapper zu erzeugen, um die unterschiedlichen Algorithmen aufzurufen, sowie die verschiedenen Outputs zu vereinheitlichen. Nachdem Wrapper ein wichtiger Bestandteil des *mlr* Paketes darstellen, soll hierauf in der nachfolgenden Untersektion 5.1.1 kurz eingegangen werden.
Zusätzlich bedarf es der Implementierung einer derartigen Infrastruktur, die das Resampling von Modellen, Optimierung von Hyperparametern, Auswählen bestimmter Eigenschaften, Umgang mit Vor- und Nachbereitung der Daten und Vergleich von Modelle auf eine statistisch gesehen sinnvolle Weise ermöglicht. All dies kann zudem computational hoch aufwendig werden und es wird somit ggf. die Parallelisierung der Experimente erforderlich.

Diese Umstände veranlassten schließlich Bischl et al. (2016) zur Ausarbeitung des *mlr* (Version 2.13) Paketes, welches die gewünschte Infrastruktur enthält und die Verwendung maschineller Lernmethoden wie etwa Klassifikation, Regression, Lebensdaueranalyse und auch Clustering, sowie deren Evaluierung und Optimierung ermöglicht. Weiterhin ist dieses Paket derart konzipiert, dass die verwendeten Methoden auf relative einfache Art und Weise an die Benutzeransprüche angepasst bzw. erweitert werden können. Einen wesentlichen Bestandteil bilden daher s.g. Wrapper, die nachfolgend Erklärung finden.

5.1.1 Wrapper

Der Begriff „Wrapper“ leitet sich vom englischen Wort „(to) wrap“ - zu deutsch „umhüllen, einpacken“ - ab und soll im Software-Kontext zumeist dazu dienen Systemstrukturen miteinander zu verbinden bzw. dadurch zu erweitern. Allerdings erfährt der Begriff des Wrappers in der Informatik und Programmierung eine breite Anwendung, weshalb dieser nicht auf allgemeine Weise definiert werden kann. Aus diesem Grund soll die Begrifflichkeit hier nun im Rahmen des *mlr* Paketes beschrieben und damit auf Erklärungen bei Bischl et al. (2016, Kap. *Wrapper*) zurückgegriffen werden.
In diesem Zusammenhang stellt ein Wrapper Adaptionsmöglichkeiten dar, um bereits integrierte Lerner bzw. Lernalgorithmen um neue Funktionalitäten zu erweitern. Hierbei unterstreicht die Bandbreite von Anwendungsoptionen, sowie Methoden, die als Wrapper implementiert sind, die Flexibilität dieses Wrapper-Ansatzes. Insbesondere gilt es in diesem Sinne die Datenaufbereitung, Imputation, Bagging, Tuning, Feature Selection, Kosten-sensitive Klassifikation, sowie die Behebung des Imbalance-Problems durch Under- und Oversampling, u.v.m. aufzuführen. Wenngleich diese Anwendungen recht unterschiedlich erscheinen mögen, so haben diese über den Wrapper-Ansatz alle gemeinsam, dass ein Lerner auf irgendeine - hier nicht genauer definierte - Weise „umhüllt“ wird und so einen neuen Lerner ausgibt, wobei Lerner auch mehrfach umhüllt werde können. Die hier angesprochene Problematik der Imbalance in Anbetracht von Klassifikationsaufgaben soll in der folgenden Untersektion nachgegangen werden, wobei Lösungsansätze diesbezüglich in *mlr* bereitgestellt sind.

5.1.2 Imbalance-Korrektur

Insbesondere im Hinblick auf binäre Klassifikation kann eine starke Unausgewogenheit bzw. Ungleichgewicht (Imbalance) zwischen den Klassen zu sehr unbefriedigenden Ergebnissen bei neuen Prädiktionen führen. Diese Imbalance und/oder stark variierende Kosten der unterschiedlichen Fehler können entsprechend auch nachteilige Auswirkungen im Hinblick auf genutzte ML-Methoden haben. Dazu genügt es sich vorzustellen, es läge innerhalb der Daten, die zur Modellanpassung genutzt werden, eine Ausprägung der Zielgröße mit einer relativen Häufigkeit von 95% vor, wohingegen die entsprechende andere Klasse nur in 5% der Fälle auftritt. Bereits ohne Nutzung von ML-Methoden oder Einbeziehung jeglicher möglicher Ein-

flussvariablen würden man bei der willkürlichen Prädiktion der dominierenden Klasse also mit hoher Wahrscheinlichkeit richtig liegen und eine Accuracy von von 95% erreichen.
Ganz allgemein gesprochen lässt sich hierzu auch die Behauptung geltend machen, dass ML-Methoden zumeist die beste Prognose-Leistung zeigen, wenn ein ausgewogenes Verhältnis zwischen den Klassen der Zielvariable besteht. Denn je nach angewandter ML-Methoden können diese bei starker Imbalance zwischen den Klassen dazu neigen die minder vertretene (Minority-) Klasse als Rauschen einzustufen und entsprechend die überrepräsentierte (Majority-) Klasse deutlich öfter zu prognostizieren. Um dieser Problematik entgegenzutreten kann auf unterschiedliche Ansätze zurückgegriffen werden, um eine stärkere Gewichtung der Minority-Klasse herbeizuführen. So entwickelte sich innerhalb des ML's einerseits der Sample-basierte und andererseits der Kosten-basierte Ansatz. Ersterer untergliedert sich im Wesentlichen in Undersampling-, Oversampling- und Hybrid-Methoden. Nachfolgend wird zu den einzelnen Aspekten eine kurze Erklärung in Anlehnung an Bischl et al. (2016, Kap. *Imbalance Classification Problems*) dargeboten.

Sample-basierte Ansätze

Die relativ einfache Idee hinter den Sample-basierten Ansätzen besteht darin, das Verhältnis der Klassen auszugleichen, indem das Gewicht der Minority-Klasse vergrößert wird. Dabei greifen alle nachfolgend gelisteten Methoden direkt auf die zugrundeliegenden Daten zurück und versuchen der Imbalance entgegenzutreten, indem die Daten „neu arrangiert" werden.

1. Undersampling: Verwerfen von Beobachtung der Majority-Klasse, um deren Effekt auf den Klassifikator zu mindern. Gleichzeitig werden alle Beobachtungen der Minority-Klasse beibehalten.

2. Oversampling: Greift im Gegensatz zum Undersampling an der Minority-Klasse an und verstärkt deren Effekt durch das Anfertigen von Kopien hier inbegriffener Beobachtungen (künstliche Beobachtungen/Daten). Gleichzeitig werden alle Beobachtungen der Majority-Klasse beibehalten.

3. Hybrid-Methoden: Strategien, die sich aus der Mischung von Under- und Oversampling ergeben.

Wie bereits in Sektion 5.1.1 dargestellt wurde, kann sowohl das Over-, als auch Under-Sampling auf die Nutzung des Wrapper-Ansatzes ausgeweitet werden. Hierbei wird dann nicht mehr die Aufgabe an sich modifiziert - durch direkte Anwendung des Over-/Under-Samplings auf die der Aufgabe zugrundeliegenden Daten -, sondern das Lerner-Objekt an sich nutzt intern Over- oder Undersampling vor jeder Modellanpassung.
Des Weiteren sollen als mögliche Methoden zur Ausweitung des Oversamplings insbesondere die „Synthetic Minority Oversampling" Technik (SMOTE), sowie das Overbagging aufgeführt werden. Nachdem das simple Duplizieren von Beobachtungen - wie es innerhalb des

Oversamplings geschieht - zu Überanpassung führen kann, wird die Konstruktion „neuer“ Beobachtungen der Minority-Klasse durch SMOTE auf andere Weise bewerkstelligt. Dabei wird jede „neue“ Beobachtung durch Interpolation von einer zufällig gewählten Beobachtung der unterrepräsentierten Klasse und einem seiner zufällig gewählten nächsten Nachbarn erzeugt. Innerhalb des Software-Paktes *mlr* kann diese Technik sowohl auf kategoriale, als auch numerische Features angewandt werden, nachdem sich die zugehörige Funktion auf die s.g. „Gower Distanz“ zur Kalkulation der nächsten Nachbarn stützt. Das faktorielle Level der neuen, künstlichen Beobachtung wird durch zufälliges Ziehen aus den zwei möglichen Ausprägungen der interpolierten Beobachtungen bestimmt. Ausweitende Beschreibungen zur Funktionsweise von SMOTE finden sich beispielsweise bei Chawla et al. (2002).
Einen weiteren Ausbau des Oversamplings stellt die Kombination von Beobachtungen durch den Ansatz des Baggings dar, dessen Grundzüge sich bereits in Untersektion 4.4.2 finden. Hierbei werden bei jeder Iteration des Bagging-Prozesses Beobachtungen der Minority-Klasse mehrfach - mit einer festgelegten Rate - zufällig gezogen. Gleichzeitig können innerhalb der Iterationen die Fälle der Majority-Gruppe entweder alle berücksichtigt werden oder aber durch Bootstrapping mit Zurücklegen gezogen werden, sodass die Variabilität zwischen den Trainingsdatensätzen gesteigert wird.

Kosten-basierte Ansätze

Im Gegensatz zum Sample- bzw- Stichproben-basierten Ansatz erfordert ein Ansatz über die Kosten spezifische Lerner, die mit unterschiedlichen, von der Klasse abhängenden Kosten (Kosten-sensitive Klassifikation) umgehen können. In diesem Kontext soll hier beispielhaft kurz der „Weighted Class“ Wrapper Erklärung finden. Unabhängig vom verwendeten Klassifikator werden dabei die Kosten als Klassengewichte zugewiesen, sodass jede Beobachtung eine Gewichtung in Abhängigkeit von der ihr zugehörigen Klasse bekommt. Somit erfolgt - ähnlich dem Sample-basierten Ansatz - eine Vergrößerung des Effekts derjenigen Beobachtung, die der Minority-Klasse angehören, durch höhere Gewichtung dieser Instanzen. Entsprechend wird der Effekt der Beobachtungen aus der Majority-Klasse reduziert.
Demzufolge kann also jeder Klassifikator, der fähig ist mit Klassengewichtung umzugehen, durch die entsprechende Anwendung des Wrapper-Ansatzes erweitert werden. Doch selbst wenn der Lerner keinen direkten Parameter für die Gewichtung der Klassen aufweist, jedoch die Gewichtung von Beobachtungen unterstützt, so können die Gewichte in Abhängigkeit von der jeweiligen Klasse intern innerhalb des Wrappers gesetzt und damit auf eine Kosten-basierten Ansatz zum Ausgleich des Imbalance-Problems zurückgegriffen werden.
Im Falle binärer Klassifikation wird lediglich die Majority-Klasse mit einem Gewicht kleiner 1 versehen, während die Minority-Klasse keine Gewichtung bzw. eine Gewichtung von 1 erhält. So gesehen werden in diesem Fall also keine wirklichen Kosten verwendet, sondern nur das Kostenverhältnis berücksichtigt.

5.1.3 Zusatzpaket mlrHyperopt

Im Falle der Hyperparameter-Optimierung sieht man sich im Angesicht mit unterschiedlichsten, von der Lern-Methode abhängigen, potentiellen Einstellungen der Hyperparameter, wobei zusätzlich deren festzulegender Suchraum eine Herausforderung darstellen kann. Doch nicht nur Neulinge im Bereich des ML's haben damit eine komplexe Voreinstellung vorzunehmen, auch unter Experten herrschen Unstimmigkeiten über die in das Hyperparameter-Tuning zu integrierende Parameter, sowie deren Spannbreite. Insbesondere da die Wahl der Hyperparameter einen (starken) Einfluss auf die Performance der einzelnen Lerner haben kann.
Um dieser Problematik entgegenzutreten bietet sich bei Analysen mittels *mlr* das Zusatzpaket *mlrHyperopt* (Version 0.1.1) an. Durch *mlrHyperopt* möchte Richter (2017) insbesondere empfohlene Parameterraum-Konfigurationen der - am häufigsten genutzten - Lern-Methodiken bereitstellen und hiermit ein vollautomatisches, einfach abzurufendes Tuning ermöglichen. Zugleich wird auch eine Schnittstelle offenbart, die das Nutzen der vollen Bandbreite der im *mlr* Paket dargebotenen Lerner, sowie Tuning-Optionen erlaubt. Zu Letzterem sind insbesondere das Grid und Random Search, sowie MBO (s. Sektion 3.2.2) zu nennen.
Nach wie vor gilt es auch hier zu beachten, dass man sich in Abhängigkeit von der Datenmenge im Trade-Off zwischen Genauigkeit und Rechenzeit befindet. Je mehr Hyperparameter betrachtet werden und je größer der Suchraum gewählt wird, desto größer wird auch die erforderliche Laufzeit des Tunings. Gemäß Koehrson (2018) ist es i.A. unmöglich die „besten" Hyperparameter in einem begrenzten Zeitraum zu bestimmen. Zugleich zeigt sich so, dass durch das vorgenommene Tuning der Punkt erreicht ist, an dem ML von einer Wissenschaft zu einer Trail-and-Error Technik übergeht. Um der Problematik hoher Dimensionalität bei der Optimierung vieler Hyperparameter entgegenzutreten werden beim automatisierten Prozess des Tunings im Paket *mlrHyperopt* daher Parameter betrachtet, die hinsichtlich eines Performance-Kriteriums zu einem Minimum hin optimiert werden können. Dies bedeutet, dass beispielsweise nicht die Anzahl der Bäume innerhalb eines RF's oder die Anzahl der Splits bei den einzelnen Bäumen Beachtung finden, weil hier prinzipiell gilt, dass mit steigender Anzahl auch die Genauigkeit zunimmt. Vielmehr interessiert man sich z.B. für die Optimierung der Anzahl zufällig auszuwählender Variablen als Kandidaten für einen Splitpunkt. Die im Hyperparameter-Tuning spezifisch zu optimierenden Parameter lassen sich größtenteils aus den Beschreibungen der einzelnen Methodiken in Kap. 4 ableiten. Zudem ist der Suchraum konkret angewandter ML-Methodiken Tab. 8.4 des Anhangs zu entnehmen.
Nachdem das aktuell zur Verfügung stehende Paket *mlrHyperopt* in seiner ersten Version vorliegt, womit potentielle Änderungen und Erweiterungen durchaus denkbar sind, und bisher auch die Parameter-Konfigurationen nur über den Online-Dienst GitHub - und nicht über CRAN - in R implementierbar sind, ist es wohl für reproduzierbare Ergebnisse zu empfehlen das genutzte Setting abzuspeichern und so jeder Zeit wieder zugänglich zu machen. Insbesondere warnt auch Richter (2017) selbst, dass sich aufgrund bestehender Konstruktion des

Paketes innere Funktionsweisen ggf. ändern können, ohne Kenntlichmachung anhand eines Versions-Updates.

Im Verlauf der in Kap. 6 offengelegten Auswertungen zeigte sich die Notwendigkeit der Visualisierungsmöglichkeiten von Benchmark-Studien. Um diese allgemein zugänglich zu machen wurde schlussendlich eine App entwickelt, die auf vorausgehend Nutzung von *mlr* aufbaut.

5.2 Entwicklung der Shiny App shinyBMR

Im Rahmen dieser Arbeit kristallisiert sich die Notwendigkeit geeigneter Visualisierungsmöglichkeiten derart getätigter Analysen heraus. Diesem Gesichtspunkt wurde mit der Entwicklung einer Shiny App mit dem Namen *shinyBMR* entgegengetreten, deren Aufbau sich wie in Abb. 5.1 gestaltet. Hierzu soll zunächst die Motivation in Untersektion 5.2.1 dargelegt werden und anschließend in Untersektion 5.2.2 ein wenig auf das konkrete Vorgehen und genutzte Methodiken, sowie damit die App selber eingegangen werden.

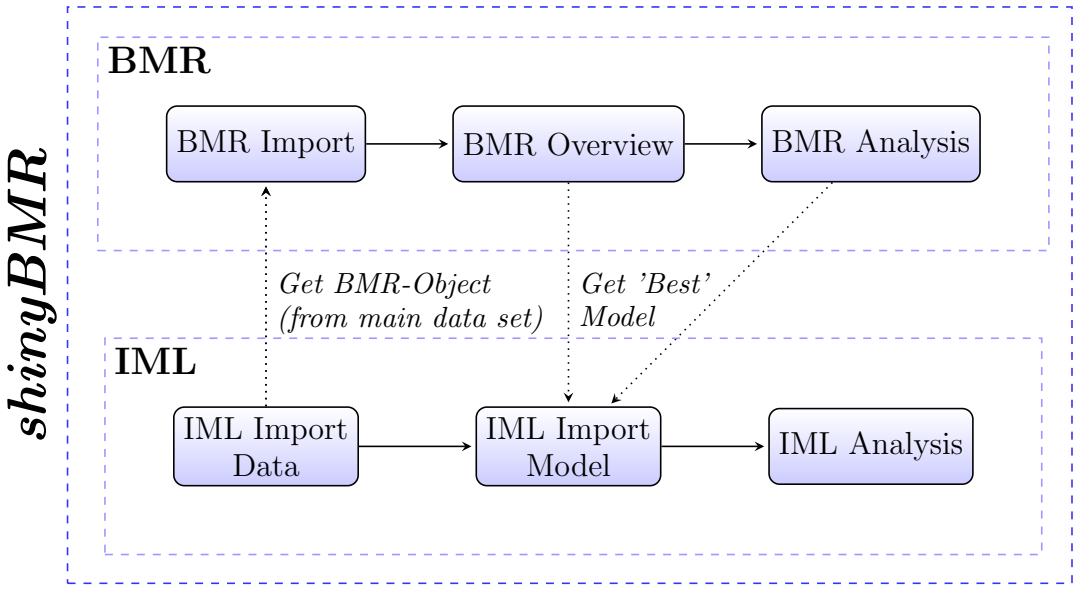


Abb. 5.1: Aufbau der App *shinyBMR*; gepunktete Pfeile als indirekte Schritte

5.2.1 Motivation

Im Zuge der umfassenden Auswertung der Daten zu akuter Cholangitis basierend auf einer Benchmark-Studie, deren Ergebnisse in Kap. 6 präsentiert werden, offenbarte sich die Komplexität dieser Art von Analysen. Nachdem zur konkreten Anwendung der Benchmark-Studie auf Funktionalitäten aus *mlr* zurückgegriffen wurde, was mit der Ausgabe eines s.g. Benchmark-Objektes mit zugehörigen BMRs - Abkürzung zu „Benchmark Results“ - verbunden ist, stellte

sich die Frage nach der Zugänglichkeit zu den inbegriffenen Ergebnissen. Das Benchmarking ist ein umfangreicher und rechenaufwendiger Vorgang, der mit der Speicherung vieler Einstellungen, Parameter, sowie Ergebnisse verbunden ist, die in Form eines Listen-Objekts gesammelt werden. So kann es insbesondere Neulingen schwer fallen den Speicherort konkret interessierender Ergebnisse zu erfassen und abzurufen. Zwar sind hierfür in *mlr* spezifische Funktionen bereitgestellt, doch erfordert auch dies das explizite Auseinandersetzen mit den umfangreichen Funktionalitäten des Paketes. So entwickelten sich erste Anreize für eine Erleichterung der Zugänglichkeit zu den BMRs.
Konkret interessiert man sich innerhalb dieser Arbeit für die Durchführung einer Benchmark-Analyse anhand von Daten zu akuter Cholangitis zur Ausarbeitung der hier spezifischen Vorrangstellung einer ML-Methodik. Dabei steht ein einzelner Datensatz zur Verfügung mit der gleichzeitigen Vorgabe der Performance-Beurteilung anhand einer oder zweier Gütemaße. Wie sich während der Analyse herausstellte, sind für derartige Fälle wenige bis keine Möglichkeiten innerhalb des *mlr* Paketes gegeben, um die Performance sinnvoll zu visualisieren. Vielmehr fokussieren Implementierungen im *mlr* Paket die Beurteilung der Lernmethodiken an sich, was auf Basis mehrerer Datensätze erfolgt. Jedoch kann verallgemeinert davon ausgegangen werden, dass insbesondere im klinischen Kontext vorrangig die Gegenüberstellung unterschiedlicher ML-Methoden in Anwendung auf einen spezifischen Datensatz angestrebt werden und in diesem Sinne geeignete Möglichkeiten der Analyse derartiger BMRs mangelnde Implementierung erfahren.
Die genannten Anreize führten schließlich zur Entwicklung der App *shinyBMR*. Hierbei ermöglichen *shiny* (Version 1.3.2), sowie dessen Erweiterung *shinydashboard* (Version 0.7.1) - bereitgestellt durch Chang et al. (2019), sowie Chang und Borges Ribeiro (2018) - Analysen aus R in interaktive Web-Applikationen zu integrieren und entsprechende Benutzeroberflächen bereitzustellen. Demgemäß kann *shiny* als ein auf R bauendes Java Script bzw. CSS Framework deklariert und dessen Nutzungsmöglichkeiten anhand von *shinyBMR* umrissen werden.

5.2.2 Erarbeitung der App shinyBMR und Nutzungsmöglichkeiten

Der Aufbau von *shinyBMR* kann anhand von Abb. 5.1 nachvollzogen werden. Insbesondere sind hierbei die Analysen von BMRs als entkoppelt von denjenigen mittels *iml* anzusehen. Die App selber kann über `https://compstat-lmu.shinyapps.io/shinyBMR/` genutzt werden bzw. ist auch in GitHub unter `https://github.com/tess-st/shinyBMR` bereitgestellt und ist damit direkt in R zugänglich. Für das Verständnis deren Nutzungsweise, sowie der Funktionalitäten ist ein Handbuch bereitgestellt, welches sich im Anschluss an den Anhang findet und dem zur Shiny App bereitgestellten Readme entspricht.

Import und Zusammenfassung des BMR-Objektes

Um BMR Objekte überhaupt mit *shinyBMR* visualisieren bzw. analysieren zu können, müssen diese eingelesen werden. Unterstützt werden hierbei *.RDS* oder *.Rdata* Formate, wobei für die anschließenden Analysen auch jeweils ein Regressions-, sowie Klassifikationsbeispiel ausgewählt werden können, die mit unterschiedlichen angewandten Lernern, sowie Gütemaßen einhergehen. Entsprechend bedarf es jedoch vorab einer vom Benutzer durchgeführten Benchmark-Studie, die mittels der zugehörige *benchmark()*-Funktion des *mlr* Paketes durchgeführt wurde, sowie einer anschließenden Abspeicherung des Objektes im korrekten Format. Alternativ sei an dieser Stelle auf die von Coors und Fendt (2019) bereitgestellte App *shinyMlr* (verfügbar unter `https://github.com/mlr-org/shinyMlr`) hingewiesen, die letztlich eine grafische Benutzeroberfläche zur Anwendung des *mlr* Paktes verkörpert und demgemäß dem Anwender Schritt für Schritt zur Anpassung eines BMR Objektes verhelfen kann.
Der Upload des Objektes innerhalb von *shinyBMR* endet in der Ausgabe einer anschaulichen Tabelle mit Informationen zu den angewandten Lernern mitsamt deren potentiell vorhergehender Bearbeitung durch SMOTE oder Hyperparameter-Tuning. Dazu werden die Werte der herangezogenen Gütemaße bereitgestellt, wobei diese sich auf aggregierter - also anhand der Mittelwerte über die einzelnen Resampling-Methoden pro Lerner - oder unaggrgierter Weise darbieten lassen. Anschließend kann sich ein erster Überblick verschaffen werden, indem wesentliche Informationen wie Name und Anzahl der Datensätze, angewandte ML-Methoden, sowie deren Bearbeitung bereitgestellt sind. Hierbei erkennt die Shiny App, ob ggf. Lerner inbegriffen sind, die mit Hyperparameter-Tuning einhergehen. In diesem Falle besteht die Zugriffsmöglichkeit auf die Performance eines einzelnen Lerners, die sich für unterschiedliche Werte der Hyperparameter ergeben, was letztlich auch grafische Darstellung findet. Dieses Analysewerkzeug ist derart interaktiv gestaltet, dass durch Anklicken des Lerners innerhalb einer bereitgestellten Tabelle, die alle ML-Methoden mit Hyperparameter-Tuning und zugehörigen „optimalen“ Hyperparametern enthält, ein Grafikfenster erscheint, innerhalb dessen der Optimierungspfad des interessierenden Parameters gegen ein beinhaltetes Gütemaß abgetragen werden kann. Sollten die bisher dargestellten Informationen nicht ausreichen, so kann ausweitend auf die Generierung von Kreuztabellen zurückgegriffen werden.

Bestimmung der „optimalen“ ML-Methodik

Insbesondere wird durch eine Benchmark-Studie natürlich das Ziel verfolgt den möglichst „optimalen“ Lerner in Anwendung auf die vorliegenden Daten zu ermitteln. An diesem Punkt stellt sich nun die Frage, wie „optimal“ in diesem Kontext zu definieren ist. Für den Fall, dass die Benchmark-Analyse auf Basis von lediglich einem Performance-Maß statt findet, kann wohl derjenige Lerner mit der im Sinne der Maximierung bzw. Minimierung besten Wert als „optimal“ erachtet werden. Hierzu erfolgt die Ausgabe eines Scatterplots, innerhalb dessen die erreichten Werte der einzelnen Lerner gegen den Wert des Gütemaßes abgetragen sind.

Der beste Wert wird durch entsprechende farbliche Markierung kenntlich gemacht. Allerdings sei dem Anwender eine subjektive Beurteilung überlassen, nachdem beispielsweise ein ähnlich guter Lerner - im Vergleich zu dem besten - eine geringere Computationszeit oder einfacherer Interpretationsmöglichkeit aufweisen könnte, sodass potentiell der Titel der „optimalen" ML-Methodik anders vergeben werden sollte. Die gleichzeitige Berücksichtigung derartiger Eigenschaften - wie etwa die Rechendauer - kann innerhalb von *shinyBMR* nicht bereitgestellt werden, da derartige Informationen nicht im BMR Objekt inbegriffen sind.
Sollte allerdings die Beurteilung der Performance innerhalb der Benchmark-Analyse anhand von zwei oder mehr Gütemaßen erfolgen, so kann eine ausweitende Betrachtung des im engeren Sinne „optimalen" Lerners vorgenommen werden. Hierzu lässt sich Bezug nehmen auf die bereits in Untersektion 3.3.2 kurz angesprochene multi-kriterielle Optimierung und damit auf die s.g. Pareto-Menge, wobei sich die vorgenommenen Implementierungen auf Roocks (2016) und damit zugleich auf das von diesem bereitgestellte R-Paket *rPref* (Version 1.3) zurückgreift. Hierbei ist eine Pareto-Menge derart definiert, dass nur diejenigen Tupels des zugrunde gelegten Datensatzes enthalten sind, die nicht Pareto-dominiert sind durch irgendein anderes inbegriffenes Tupel. Man spricht von einem Tupel t als Pareto-dominant gegenüber einem anderen Tupel t', falls diese besser oder gleich gestellt ist in Hinsicht auf alle relevanten Attribute und wenn gleichzeitig ein Attribut existiert, für das t strikt besser bzw. dominant ist verglichen mit t'. Dieses Konzept entwickelt sich unter dem Namen „Skyline Operator" durch Börzsönyi et al. (2001). Die hier getätigten Beschreibungen sollen möglichst einfach und knapp gehalten werden, sodass insbesondere hinsichtlich mathematischer Formulierungen und Algorithmen auf Roocks (2016) verwiesen sei.
Letztlich lässt sich also auch mittels der vorhergehenden Definition die Menge an Lernern bestimmen, die Pareto-dominant in Anbetracht ausgewählter Gütemaße sind. Allerdings ist hierbei nur die Beurteilung über die Pareto-Dominanz anhand zweier Maße bereitgestellt, sodass diese insbesondere auch grafisch noch visualisierbar ist. Derartige Grafiken werden als Skyline-Plots bezeichnet und markieren die s.g. Pareto-Front, die sich aus der stufenweisen Verbindung der Punkte aus der Pareto-Menge ergeben. Entsprechend kennzeichnen alle Punkte, die nicht auf der Pareto-Front liegen, Lerner, die durch eine andere ML-Methode dominiert werden. Ein Lerner wird dabei genau dann dominiert, wenn es eine weitere Methodik gibt, die mindestens gleichermaßen gut in Anbetracht aller Attribute und besser hinsichtlich mindestens eines dieser Attribute ist - hier also bzgl. zweier Gütemaße.
Weitergehend kann man ggf. auch an denjenigen Tupeln interessiert sein, die nun geringfügig schlechter sind wie die als „optimal" deklarierten. Dabei kann man sich der s.g. „top-k Selektion" bedienen, sodass letztlich iterativ für die jeweils verbleibenden Daten bei Ausschluss der zur entsprechenden Pareto-Front angehörigen Tupel erneut die resultierende Pareto-Dominanz ermittelt wird. So kann schließlich für jeden Punkt dasjenige Level bestimmt werden, zu welchem dieser als der Pareto-Front angehörig bestimmt werden würde. Auch dies lässt sich

Visualisieren und stellt letztlich die Ergänzung des bisherigen Skyline-Plots um die Pareto-Fronten der übrigen Level dar, weshalb diese im Folgenden als Skyline-Level-Plots bezeichnet werden. Rückschließend ist in der Begrifflichkeit der top-k Selektion also k gleichzusetzen mit der Anzahl an Pareto-Leveln. Zumeist zeigen sich jedoch innerhalb der beschriebenen Skyline-Level-Plots sich überlappende Fronten, was dadurch zu begründen ist, dass die Definition der Pareto-Dominanz lediglich die Präferenz-Stellung in einer Dimension erfordert und in anderen Dimensionen eine Gleich- oder Besser-Stellung erwartet. Dies kann dahingegen abgeändert werden, dass - in diesem Fall - in beiden Dimensionen strikte Dominanz gefordert wird, sodass keine Intersektionen auftreten, wobei sich entsprechend zumeist die Zuordnung der Pareto-Level ändern bzw. sich allgemein die Anzahl derer erhöht.

Explizite Analyse des BMR Objektes

Bereits angesprochen wurde die Tatsache, dass sich innerhalb klinischer Studien zumeist die Notwendigkeit der Analysen mittels einer Benchmark-Analyse in Anbetracht eines spezifischen Datensatzes ergibt. Hierzu finden sich bezüglich des *mlr* Paketes wenige Visualisierungsmöglichkeiten, nachdem dabei weitestgehend von der Bereitstellung mehrerer Datensätze für die explizite Beurteilung eines Lerners und der damit verbundenen Vergabe von Rängen ausgegangen wird. Entsprechend wird ein Lerner also mehr im Sinne seiner Methodik und Leistungsstärke beurteilt, statt dass davon ausgegangen wird für eine bestimmte Situation einen Lerner möglichst hoher Performance ermitteln zu können. Bezug nehmend auf Letzteres soll innerhalb dieser App deshalb auch eine Grundlage gegeben werden die Notwendigkeit von rechenaufwendigen Wrapper-Funktionalitäten wie das Hyperparameter-Tuning zu beurteilen. In diesem Rahmen sind Boxplots und Heatmaps bereitgestellt, die letztlich auf unterschiedliche Weise die gleichen Informationen vermitteln sollen. Fokussiert wird hierbei die Performance der im BMR-Objekt inbegriffenen ML-Methoden durch Bewertung anhand eines Gütemaßes. Eine erste Unterteilung findet jeweils durch Gruppierung der einzelnen inbegriffenen ML-Methoden statt. Innerhalb derer werden zudem die einzelnen Bearbeitungsmaßnahmen bzw. Wrapper-Funktionalitäten unterschieden. Interaktiv kann schließlich auch die Performance auf Basis des aggregierten oder unaggregierten BMR-Objektes veranschaulicht werden. Das Gefälle der Leistungsstärke zwischen den einzelnen Lernern soll entsprechend durch die Spannweite der Boxplots bzw. die Distanz der Performance-Werte an sich verdeutlicht werden, während dieses innerhalb der Heatmap über den visuellen Eindruck durch farbliche Abstufungen kenntlich gemacht wird.
Die bisherigen Werkzeuge für die grafische Analyse von Benchmark-Studien stützt sich auf die Benutzer-gesteuerte Auswahl eines Gütemaßes. Auch die Beurteilung anhand der Pareto-Front geht nicht über zwei Dimensionen hinaus. Um dem Rechnung zu tragen, sollte auch eine Funktionalität für die gleichzeitige Bewertung anhand mehrerer im BMR-Objekt enthaltener Performancemaße bereitgestellt werden. Dies lässt sich durch den s.g. „Parallel Coordinates

Plot“ (PCP) verwirklichen. Im Gegensatz zu einem Koordinatensystem werden die Achsen hier nicht im rechten Winkel zueinander angeordnet, sondern parallel zueinander, sodass die Möglichkeit besteht hochdimensionale Strukturen abzubilden. Dabei erhält also jedes Maß eine eigene (y-)Achse und auf dieser wird der entsprechende Performance-Wert der einzelnen Lerner abgetragen. Reiht man diese Achsen aneinander und verbindet die jeweiligen Werte bei zugehörigen Gütemaßen gemäß der unterliegenden Lerner verbildlicht sich die Leistungsstärke der einzelnen ML-Methodiken durch den steigenden/fallenden Verlauf der zugehörigen Linien. Neben alle dem kann auch auf eine grafische Darstellung aus *mlr* zurückgegriffen werden, bei der die Verteilung der Performance-Werte über die einzelnen Resampling-Iterationen hinweg anhand von Boxplots dargestellt wird. Demgemäß werden hier die unaggregierten Ergebnisse der jeweiligen Benchmark-Studie betrachtet.

Einbettung von iml

Schließlich soll mittels *shinyBMR* auch die Möglichkeit gegeben werden - insbesondere s.g. Blackbox-Methoden bzw. generell - ML-Methoden „interpretierbar“ zu machen. Für derartige Analysen empfiehlt es sich das *iml* Paket heranzuziehen, wobei dessen Methoden den Fokus der nachfolgenden Sektion 5.3 darstellen sollen. Hierzu finden sich im Besonderen auch grafische Illustrationen, die derart mittels *shinyBMR* interaktiv erzeugt werden können. Dazu bedarf es allerdings der Bereitstellung des fokussierten Modells, sowie des der Anpassung unterliegenden Datensatzes durch den Anwender. Während der Import des Modells in *shinyBMR* mit der Bereitstellung einer Liste inbegriffener Informationen erfolgt, wird zu dem importierten Datensatz eine Tabelle mit der Zusammenfassung bzgl. jeder Variable bereitgestellt. Ausweitend wird durch Auswahl beliebiger Zeilen dieser Zusammenfassung - entsprechend des Types der jeweiligen Variable - eine angemessene Grafik ausgegeben, ebenso wie der jeweilige Zusammenhang zwischen allen ausgewählten Größen visualisiert wird.
Es sei vermerkt, dass die Analysen nunmehr losgelöst sind von BMRs und sich tendenziell auf jeden Lerner, der mittels *mlr* angepasst wurde, anwenden lassen. In diesem Sinne seien an dieser Stelle die Beschreibungen zu *shinyBMR* abgeschlossen und für die Funktionsweise hinter der integrierten Analyse mittels *iml* auf die nachfolgende Sektion 5.3 verwiesen. Bevor jedoch genauer auf *iml* eingegangen wird und damit letztlich die Sektion zur Beschreibung von *shinyBMR* ausgeweitet wird, sollen hier noch kurz Anreize im Hinblick auf Erweiterungsmöglichkeiten offengelegt werden, nachdem wohl ein „Mehr“, „Besser“, „Schöner“ immer möglich zu sein scheint.

5.2.3 Herausforderungen und Ausblick

Generell geht wohl keine statistische Analyse ohne gewisse Herausforderungen einher. Doch sollen die hieraus gezogenen Feststellungen und Lösungen die Grundlage von Anreizen für neue Ansätze, Ausbaufähigkeiten und Erweiterungsmöglichkeiten darstellen. In diesem Sinne wird innerhalb nachfolgender Untersektion kurz auf Problemstellungen und denkbare Ausweitungen im Bezug auf *shinyBMR* eingegangen.

Herausforderungen bei der Generierung von shinyBMR

Zunächst sieht man sich wohl generell mit der Problematik fehlender Kenntnisse bei Nutzung neuer Methodiken konfrontiert. So zeigte sich zu Beginn der Entwicklung von *shinyBMR* die größte Herausforderung in Form des Umgangs mit der bis dahin relativ unbekannten Skriptsprache Java. Erschwerend kommt wohl auch der nunmehr recht komplexe bzw. geschachtelte Aufbau der Shiny App hinzu, der sich durch - größtenteils mehrfach logische Abfragen beinhaltenden - Bedingungen, sowie der Generierung und dem Zugriff auf reaktionsfähige Eingabewerte bzw. Funktionen ergibt.
Beispielhaft sei hierzu die folgende Situation beschrieben: zunächst wurden die in *shinyBMR* inbegriffenen Grafiken in Form von Boxplots und Heatmap auf Basis der Analysen zu akuter Cholangitis entwickelt. Diese galt es nunmehr derart verallgemeinert zu implementieren, dass sich eine Vielzahl von BMR-Objekten hierauf anwenden lassen, wobei auf möglichst interaktive Weise dem Benutzer eine Vielzahl möglicher Einstellungen überlassen werden sollte. In diesem Zuge sollte ein Slider bereitgestellt werden, der die Skalengrenzen des Gütemaßes und damit letztlich die Zoom-Einstellung steuern soll. Fraglich ist dabei die Initialisierung der Startwerte, wobei das jeweilige Minimum bzw. Maximum der gemessenen Werte nicht unbedingt die Möglichkeit bieten, die absoluten Intervallgrenzen - also für das AUC beispielsweise $[0; 1]$ - abzubilden. Folglich sollte die Initalsierung über die in *mlr* hinterlegten Informationen zu den Gütemaßen angesteuert und eingebunden werden, sodass die Intervallgrenzen hierüber fixiert bzw. im Falle von nach oben oder unten offenen Grenzen mit dem minimalen oder maximalen Wert aus dem BMR-Objekt versehen werden können. Allerdings zeigten sich durch Einlesen des BMR-Objektes zu akuter Cholangitis erste Schwachstellen: die Performance-Beurteilung beruht auf dem in *mlr* selbstständig konstruierten Gütemaß pAUC. Nachdem also das pAUC als solches (noch) nicht in *mlr* als implementiertes Gütemaß hinterlegt ist, liefern die bis dahin in *shinyBMR* inbegriffenen Abfragen zu den Skalengrenzen eine Fehlermeldung. Entsprechend bedarf es der Berücksichtigung weiterer konditionaler Schleifen, die eine reibungslose Funktionsweise der App ermöglichen, was allerdings mit zunehmender Komplexität und Schachtelung einhergeht.

Ausblick und Erweiterungsmöglichkeiten

Nachdem die Generierung dieser App von einer spezifischen Auswertung eines Klassifikationsproblems ausgehend aufgebaut wurde, finden sich bezüglich der im BMR-Objekt potentiell inbegriffenen Wrapper bisher nur Hyperparameter-Tuning, sowie SMOTE berücksichtigt. In Anbetracht dessen könnten mögliche Erweiterungen der App dahingehend erfolgen, dass auch andere Wrapper bzw. die daraus resultierenden Lerner extrahiert und analysiert werden könnten. Bezüglich zusätzlich zu berücksichtigender Wrapper sei auf 5.1.1 verwiesen.
Insofern die einzelnen Rechenzeiten im BMR Objekt, welches mithilfe von *mlr* erzeugt wird, abgespeichert werden könnten - unter dem gleichzeitigen Trade-Off zusätzlichen Speicherbedarfs -, wäre es im Zuge der Bestimmung des „besten“ Modells mittels der Pareto-Dominanz möglich eine weitere nicht zu verachtende Dimension zu berücksichtigen. Ebenso könnte eine über die zwei Dimensionen hinausgehende Bewertung der Lerner mittels der Pareto-Dominanz diskutiert werden.
Eine weitere Änderungsmöglichkeit sich im Hinblick auf den PCP ergeben. Bisher ist diese Grafik derart - und ohne Zugriff auf eine bereits existierende Funktion von PCP's - implementiert, dass die jeweiligen (y-)Achsen der Gütemaße gemäß ihres kleinsten hin zu deren größten (von unten nach oben) Wert verlaufen. Insofern nun der Einfachheit halber zwei Maße betrachtet werden, bei denen eines zu einem Maximum und das andere zu einem Minimum hin optimiert werden sollen, ergeben sich innerhalb des PCP's zwei gegenläufige (y-)Achsen. Entsprechend kann sich auf den ersten Blick ein (falscher) Eindruck bzgl. der Performance-Leistung - abfallend vs. steigend - einstellen und sich somit potentiell eine irrtümliche Präferenz hinsichtlich der Lerner ergeben. Alternativ wäre es vielleicht sinnvoll die Achsen nicht gemäß ihrer natürlichen Ordnung von kleinen Werten hin zu großen Werten anzuordnen, sondern vom schlechtesten hin zum besten Wert der jeweiligen Skala bzw. des Gütemaßes.
Stark ausweitend betrachtet kann es sinnvoll sein dem Anwender die Möglichkeit bereit zu stellen die Benchmark-Studie an sich bereits in der App durchzuführen und hierauf die in *shinyBMR* bereitgestellten Analyse-Methoden einzusetzen. Gegebenenfalls wäre dies durch sinnvolle Verknüpfung bzw. Einbettung in *shinyMlr* möglich, sodass auch nach Wahl des Gewinner-Modells eine interaktive Anpassung und Auswertung der entsprechenden ML-Methodik ermöglicht wird. So kann es an dieser Stelle schließlich zweckmäßig sein die Implementierung der *iml* Methoden einfließen zu lassen. Hinsichtlich *mlr* wäre es wohl auch denkbar eine direkte Verknüpfung zu dem Zusatzpaket *mlrHyperopt* herzustellen, sodass bei durchzuführendem Hyperparameter-Tuning eine Voreinstellung des Suchraumes dargeboten würde.
Mit diesen Ausführungen gelangt man wohl zu der genannten Problematik, dass ein „Mehr“ an Möglichkeiten bzw. Erweiterungen oder Änderungen wohl immer denkbar sind und es daher innerhalb von Ausarbeitungen zwingend erforderlich ist, klare Zielpunkte zu bestimmen. Mit der zusätzlichen Implementierung von *iml* in *shinyBMR* ist die Zielvorstellung dieser Arbeit erreicht und in diesem Sinne sei nachfolgend genauer auf die Methodiken von *iml* eingegangen.

5.3 Interpretable Machine Learning (Zusatzpaket iml)

Komplexe, nicht-parametrische Modelle - wie diese insbesondere Einzug im ML gefunden haben - verzeichnen v.a. im Falle von Prädiktionen hohe Leistungen bzw. Erfolge. Hierbei kommt die genannte Komplexität durch die s.g. „Black Box“ zu Tage. Derartige Methoden sind - im Gegensatz zu parametrischen Methoden - durch fehlende Interpretierbarkeit gekennzeichnet, sodass im heutigen Zeitalter diese ML-Methoden zwar einfach anzuwenden sind, der Computer jedoch keine Rückschlüsse auf die Interpretation der Ergebnisse gibt. Dieser Problematik wird auf unterschiedliche Art und Weise versucht entgegenzutreten und in dem Kontext der bisherigen Analysen soll daher auf das *iml* Paket - abgekürzt zu verstehen als „Interpretable Machine Learning“ - von Molnar (2018) zurückgegriffen werden.
Wie kann nun Interpretierbarkeit im Rahmen von ML-Methodiken erreicht werden? Zum einen sei darauf aufmerksam gemacht, dass es nicht alle Methoden an Interpretationsmöglichkeit mangelt. Insbesondere sind parametrische Modelle - wie etwa lineare oder logistische Regression - interpretierbar und auch einige non-parametrische, zu denen beispielsweise Entscheidungsbäume zu nennen sind. Sollte man also insbesondere die Interpretierbarkeit von Blackbox-Methoden anstreben, so kommen im Wesentlichen Modell-agnostische Hilfsmittel zum Einsatz, die sich generell auf jegliche Art von USL-Modellen anwenden lassen. Kennzeichnend für diese Werkzeuge der Interpretation ist grob gesprochen das Abändern des Inputs, welcher den ML-Methoden übergeben wird, und anschließendes Messen der Änderungen im Prädiktions-Output. Gleichzeitig können Modell-agnostische Methoden weiter differenziert werden durch deren Ansatzpunkt, der entweder auf globaler Ebene das Modellverhalten über alle Dateninstanzen hinweg ansetzt oder der Analyse individueller Prädiktionen fußt.
Molnar (2018, Kap. 1) trifft hierbei folgende Einteilungen möglicher Werkzeuge für die Zugänglichkeit zur Interpretation, die insbesondere in Form des *iml* Paketes bereitgestellt sind:

- Globales Verhalten des Modells betreffend: Partial Dependence Plots, Accumulated Local Effects, Feature Interaction und Importance, Global Surrogate Models, sowie Prototypes - Criticisms.
- Erklärung individueller Prädiktionen: Local Surrogate Models, Shapley Value Explanations, Counterfactual Explanations und Adversial Examples.
- Methodiken, welche sich hinsichtlich beider Aspekte eignen: Individual Conditional Expectation und Influential Instances.

Wenngleich hier nur die englischen Fachtermini aufgeführt sind und damit noch keine Erklärung der Funktionsweise dieser Methoden getan ist, soll die vorhergehende Einordnung im Wesentlichen der Schaffung eines ersten groben Überblicks dienen. Insbesondere werden im Rahmen dieser Arbeit nicht alle genannten Aspekte Erläuterung finden, sondern es wird

weitestgehend ein Grundverständnis zu den einzelnen Methoden vermittelt und es sei für genauere Ausführungen auf Molnar (2018) verwiesen, sowie insbesondere auch der Querverweis zu Hastie et al. (2008) hergestellt sei. Weitere Quellen bzw. hilfreiche Literatur werden bei den einzelnen Sektionen genannt.
Zunächst gilt es nun in diesem Zusammenhang der Begriff der „Interpretierbarkeit“ zu umreißen. Hierbei sei vorab vermerkt, dass keine mathematische Definition in diesem Sinne aufgestellt werden kann. Zum groben Verständnis ziehen man das nachstehende Zitat zu Rate. Damit soll zum Ausdruck gebracht werden, dass mit steigender Interpretierbarkeit eines ML-Modells auch das Maß steigt, mit dem jemand Entscheidungen bzw. Prädiktionen eines Algorithmus nachvollziehen kann.

„Interpretability is the degree to which a human can understand the cause of decision.“
Miller (2017, S. 14)

Insgesamt sollte deutlich werden, dass bisher der Fokus auf die Güte/Performance einer ML-Methodik bzw. deren prädiktive Leistungsfähigkeit gelegt wurde. Mittels der Interpretierbarkeit soll nun die Thematik vertieft werden und im Wesentlichen der Frage nach dem „Warum?“ - eine Prädiktion derartig ausfällt - nachgegangen werden. Ziel dessen ist es, mehr über das zugrundeliegende Problem an sich und die Daten in Erfahrung zu bringen, sowie die potentiellen Ursachen für das Scheitern bestimmter ML-Methoden, bei deren konkreter Anwendung, zu ermitteln. Damit entspringt die Frage nach dem „Warum?“ der Unvollständigkeit in der Formalisierung eines Problems. Es gilt also die Werkzeuge der Interpretierbarkeit insbesondere dann heranzuziehen, wenn korrekte Prädiktionen zwar einen Teil der Problematik lösen, aber potentielle Fehler mit drastischen Konsequenzen/Kosten einhergehen oder aber auch eine Methodik in ihrer Anwendung bisher nicht ausreichend erprobt bzw. evaluiert ist.
Indem somit bestimmte Entscheidungen von ML-Modellen erklärt werden können, fällt zudem das Prüfen nachfolgender Punkte deutlich leichter bzw. finden indirekte Ansprache:

- Fairness: Sicherstellen unverzerrter Prädiktionen und keine implizite oder explizite Diskriminierung bestimmter Gruppen,
- Geheimhaltung: Unzugänglichkeit sensibler Informationen in den Daten,
- Robustheit: Kleine Änderungen des Inputs führen nicht zu starken Veränderungen der Prädiktionen,
- Kausalität: Einzig kausale Beziehungen werden aufgegriffen,
- Zuverlässigkeit: Mehr Vertrauen in ein erklärbares System als in Blackbox-Methodiken.

Ein erster, verhältnismäßig trivialer Ansatz für den Zugang zur Interpretierbarkeit soll mittels der nachfolgenden Untersektion 5.3.1 Erklärung finden. Für nachfolgender Beschreibungen soll zudem ein direkter Zugang zu der Funktionalität von *shinyBMR* dargeboten werden, indem teilweise Grafiken aus der App durch Anwendung der iml-Methodiken dargeboten werden.

Hierzu sei erwähnt, dass es sich bei dem zugrunde gelegten Modell um einen cForest - ohne Hyperparameter-Tuning oder SMOTE - handelt, der zwar nicht als Gewinnermodell aus der in Kap. 6 durchgeführten Benchmark-Studie zu Daten akuter Cholangitis hervorgeht, allerdings im engeren Sinne keine einfache Zugänglichkeit zur Interpretierbarkeit der Ergebnisse bietet.

5.3.1 Partial Dependence (PD) Plot

Der s.g. „Partial Dependence Plot“ (PDP) kann herangezogen werden, um den marginalen Effekt von ein oder zwei Features auf den von einer ML-Methode prognostizierten Output darzustellen. Insbesondere kommt so die Struktur zwischen der Einfluss- und der Zielvariable zum Vorschein, der sich etwa linear, monoton oder in komplexerer Form darstellen kann. Insgesamt kann der PDP als globale Methode erachtet werden, nachdem alle Instanzen berücksichtigt werden und die Ausgabe als Statement über den globalen Zusammenhang der Features mit dem prognostizierten Outcome angesehen werden kann. Die nachfolgenden Beschreibungen hierzu orientieren sich an Molnar (2018, Kap.5.1) und für ausführliche Erklärungen sei insbesondere auf Zhao und Hastie (2018) verwiesen.
Im Regressionskontext gestaltet sich die partielle Abhängigkeitsfunktion wie in Gl. 5.1. Hierbei entspricht $\boldsymbol{x}_S$ derjenigen Einflussgröße(n), für welche die partielle Abhängigkeitsfunktion dargestellt werden soll, während $\boldsymbol{x}_C$ den verbleibenden Variablen entspricht, welche zur Berechnung des Modells $\hat{f}$ herangezogen wurden. Zumeist umfasst das Set S nur ein oder zwei Features und die Kombination von dem Subset S und C ergibt den gesamten Raum der unabhängigen Variablen $\mathcal{X}$. Durch die Marginalisierung über alle Features in C, ergibt sich die Prädiktion des Outcomes einzig in Abhängigkeit vom Feature S, wobei jedoch Interaktionen dessen mit anderen Einflussgrößen inkludiert sind.

$$\hat{f}_{\boldsymbol{x}_S}^{PDP}(\boldsymbol{x}_S) = \mathbb{E}_{\boldsymbol{x}_C}[\hat{f}(\boldsymbol{x}_S, \boldsymbol{x}_C)] = \int \hat{f}(\boldsymbol{x}_S, \boldsymbol{x}_C) d\mathbb{P}(\boldsymbol{x}_C) \tag{5.1}$$

Die partielle Funktion $\hat{f}_{\boldsymbol{x}_S}$ wird schließlich durch das Mitteln über die Trainingsdaten geschätzt, was auch bekannt ist als Monte-Carlo-Methode:

$$\hat{f}_{\boldsymbol{x}_S}^{PDP}(\boldsymbol{x}_S) = \frac{1}{N} \sum_{i=1}^{N} \hat{f}(\boldsymbol{x}_S, \boldsymbol{x}_C^{(i)}) \tag{5.2}$$

Hierbei gilt zu beachten, dass dem PDP die Annahme zugrunde liegt, dass die Features in C nicht mit dem/denen aus S korreliert sind. Sollte diese Annahme verletzt sein, zeigt sich dies i.A. in Form von Schätzungen/Datenpunkten im PDP, die sehr unwahrscheinlich, wenn nicht gar unmöglich sind.
Im Falle von Klassifikationsaufgaben, wobei die ML-Methode den Output durch Klassenwahrscheinlichkeiten ausgibt, kann der PDP zur Anwendung kommen, indem die die Wahrscheinlichkeiten der jeweilige Klasse für die unterschiedlichen Werte von S abgetragen werden.

Entsprechend stellt sich der PDP in diesem Fall durch mehrere Linien - in Abhängigkeit von der Anzahl an Klassen in der Zielvariable - oder durch separate PDPs pro Klasse dar. Eine Darstellung mit letzterer Aufteilung findet sich in Abb. 5.2, wobei die sich im Kontext des PDP's ergebenden Linien gelb gekennzeichnet sind und letztlich den Mittelwert der individuellen Kurven - siehe hierzu ICEs - widerspiegeln. Genauere Ausführungen finden sich in der nachfolgenden Untersektion 5.3.2.
Zusammenfassend ist die Interpretation des PDP's im Falle unkorrelierter Features also offensichtlich: die Graphen spiegeln den gemittelten, prognostizierten Outcome in Anbetracht der vorliegenden Daten wider, wenn eine Veränderung der j-ten unabhängigen Größe erfolgt. In diesem Sinne dienen PDPs also der kausalen Interpretation bzw. der Analyse der kausalen Wirkungsbeziehung zwischen Feature und Prädiktion.
Offensichtlich kritisch zu beachten, gilt es den Umstand der darstellbaren Dimensionen. Da es nicht möglich ist mehr als drei Dimensionen bildlich darzustellen, können mittels der beschriebenen Grafik nicht mehr als zwei Einflussvariablen in ihrer Beziehung zum Outcome untersucht werden. Zudem können heterogen Effekt potentiell unerkannt bleiben, nachdem einzig die gemittelten marginalen Effekte grafisch dargeboten werden. Insbesondere kritisch zu beäugen gilt es aber auch die bereits zuvor genannte Annahme der Unabhängigkeit zwischen den Features. Diese bezeichnet eine sehr starke Annahme, der zumeist im ML-Kontext nicht Rechnung getragen werden kann. Man bedenke hier beispielsweise die gemeinsame Aufnahme von Körpergröße und Gewicht für die Prognose eines interessierenden Outcomes. Letztlich werden damit neue Datenpunkte in Bereichen der Feature-Verteilung erzeugt, die sehr unwahrscheinlich bzw. theoretisch nicht möglich sind. Einen Ausweg aus dieser Problematik stellen beispielsweise s.g. „Accumulated Local Effect" (ALE) Plots, die auf die konditionale an Stelle der marginalen Verteilung zurückgreifen und in Untersektion 5.3.3 Anklang finden.

5.3.2 Individual Conditional Expectation (ICE) Plot

Wie mittels der vorangehenden Untersektion bereits deutlich wurde, zeigt sich die Schwäche der PDPs bei dem Vorliegen substantieller Interaktionseffekte, sodass die partielle Beziehung zum Response heterogener Natur sein kann und entsprechend eine gemittelte Kurve - wie durch PDPs bereitgestellt - die eigentliche Komplexität der modellierten Beziehung verschleiert. Gemäß A.Goldstein et al. (2013) verfeinern bzw. verbessern der nun Anklang findende „Individual Conditional Expectation Plot" (ICEP) den PDP durch Darstellung der funktionalen Beziehung zwischen dem prognostizierten Response und einem Feature für die individuellen Beobachtungen. Dadurch wird also die Variation in den angepassten Werten über die gesamte Spannweite der Kovariablen fokussiert, sodass insbesondere deren Lokation, sowie das entsprechende Ausmaß dieser Variation verdeutlicht wird. Damit lässt sich folgern, dass sich ein ICEP eine Linie pro Instanz umfasst, um die Beziehung zwischen dem geschätzten Response und einem Feature darzustellen. Im Umkehrschluss ergibt sich der PDP also durch

mitteln dieser individuellen Kurven.
Möchte man den ICEP formaler ausdrücken, so kann man sich auf Molnar (2018, Kap. 5.2) beziehen und folgende Beschreibung vornehmen: In ICEPs wird für jede Instanz in $\{\boldsymbol{x}_S^{(i)}, \boldsymbol{x}_C^{(i)}\}_{i=1}^N$ die Kurve $\hat{f}_S^{(i)}$ gegen $\boldsymbol{x}_S^{(i)}$ abgetragen, während $\boldsymbol{x}_C^{(i)}$ fixiert/unverändert bleibt.
Es kann durchaus möglich sein, dass ein Unterschied in den ICE-Kurven der Individuen zu Tage tritt, weil diese bei unterschiedlichen Prädiktionen beginnen. Um diese Problematik des scheinbaren Unterschiedes auszuschließen, bietet es sich an die Kurven um einen spezifischen Punkt des Features zu zentrieren und lediglich die Differenz in den Prädiktionen zu diesem Punkt abzutragen. Die resultierende Grafik wird als zentrierter ICEP bezeichnet und leitet sich vom englischsprachigem „centered ICE (c-ICE) Plot“ ab. Mathematisch können diese Kurven also folgendermaßen ausgedrückt werden:

$$\hat{f}_{cent}^{(i)} = \hat{f}^{(i)} - \mathbb{1}\hat{f}(x^a, \boldsymbol{x}_C^{(i)}), \tag{5.3}$$

mit $\mathbb{1}$ = *Einservektor mit entsprechender Dimensionalität (normalerweise 1 oder 2 Dim.)*

x^a = *Ankerpunkt (Punkt des Features, um den zentriert wird)*

Eine weitere Möglichkeit Heterogenität im Rahmen der ICEPs zugänglich zu machen, besteht durch die Nutzung der Derivate einer Funktion bzw. Kurve, wobei für genauere Ausführungen wiederum auf Molnar (2018, Kap. 5.2) verwiesen sei. An dieser Stelle sei nun allerdings darauf verzichtet und stattdessen eine Darstellung der ICEs mitsamt der Abtragung des punktweisen Mittelwerts dieser individuellen Kurven und folglich der durch den PCP hervorgebrachten gelb markierten Linie dargeboten. Hierbei ist ein cForest - nicht als Gewinnermodell resultierend aus einer konkreten Benchmark-Analyse - zur beispielhaften Modellierung der Daten akuter Cholangitis (s. Kap. 6) herangezogen. Dabei wird eine getrennte Betrachtung der beiden Klassifikationsgruppen bereitgestellt, sodass sich entsprechend ein gegenläufiges Bild für das Todesereignisse in der Klasse „ ja“ und „nein“ ergibt. Fokussiert man nun die Sterbewahrscheinlichkeit - im Weiteren bezeichnet als In-Hospitale Mortalität (IHM) - in partieller Abhängigkeit von der absoluten Anzahl an Leukozyten so lässt sich tendenziell - ohne Heranziehen genauer Maßzahlen - ein leicht erhöhtes Risiko für das Eintreten eines Todesereignisses mit zunehmender Zahl an Leukozyten erkennen. Diese Zunahme scheint erstmalig ab einem Wert von etwa 15 Leukozyten pro G/l einzusetzen. Es sei insbesondere bedacht, dass die Beobachtungen in etwa ab mehr als 25 Leukozyten pro G/l deutlich abnehmen und damit auch Wahrscheinlichkeitsaussagen in diesem Bereich mit deutlicher Ungenauigkeit einhergehen. Der PDP kann also als nahezu glatt angesehen werden, wobei sich wenn dann eine leichte Zunahme der prognostizierten, mittleren (punktweisen) Sterbewahrscheinlichkeit mit Anstieg der Leukozyten zeigt. Die ICEPs dagegen verbildlichen diejenigen Beobachtungen, für die die Erhöhung der Leukozyten tatsächlich mit höherem prognostizierten Sterberisiko einhergehen und beschreiben somit, wie sich das individuelle von dem gemittelten Verhalten unterscheidet.

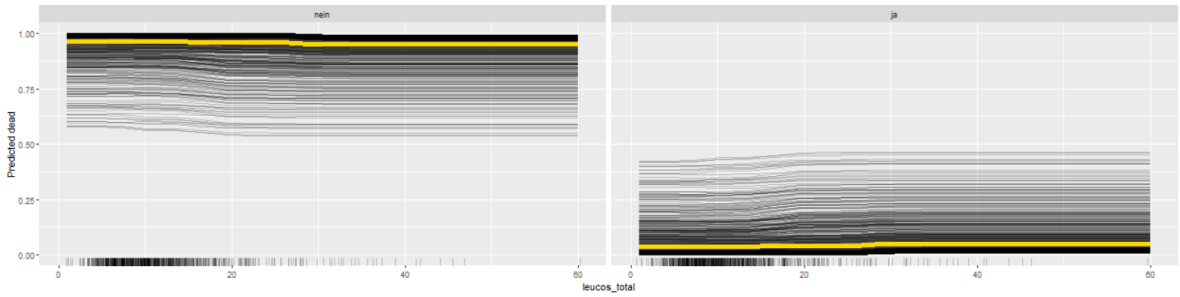


Abb. 5.2: Beispielhafte Veranschaulichung zu PD (gelb) und ICE (schwarz) Plots, cForest: Funktionaler Zusammenhang zwischen prognostizierter IHM (nein/ja) in partieller Abhängigkeit von der Anzahl an Leukozyten.

5.3.3 Accumulated Local Effects (ALE) Plot

In dieser Untersektion soll nun kurz auf die bereits genannten „Accumulated Local Effects" Plots (ALEP) eingegangen werden, die im Gegensatz zu den PDPs nicht auf die marginale, sondern die konditionale Verteilung der Features zugreifen. Hierbei wird ALEPs gegenüber PDPs der Vorteil schnellerer Berechenbarkeit, sowie geringerer Verzerrung zugesprochen. Um dieser Verzerrung, die sich aufgrund von Korrelationen der Features untereinander ergibt, nun also Rechnung zu tragen, wird im Rahmen der ALEs letztlich über die konditionale Verteilung eines interessierenden Features gemittelt. Grob formuliert wird hierbei für die die einzelnen (Grid-) Werte der Einflussgröße der Mittelwert über Prädiktionen von Instanzen mit gleichem bzw. ähnlichem Wert gebildet. Trägt man die sich daraus ergebenden Effekte in einer Grafik ab, wobei in diesem Falle die bedingte Verteilung der Ziel- gegeben dem (Grid-) Wert einer interessierenden Einflussvariable dargestellt wird, ergibt sich der s.g. Marginal- bzw. kurz M-Plot.

Durch die Nutzung derartiger M-Plots wird zwar das Mitteln über Prädiktionen unwahrscheinlicher Dateninstanzen verhindert, jedoch Vermischen sich hier der Effekt des Features mit denen korrelierter Einflussgrößen, sodass hier nun ALE Plots ansetzen. ALEPs fußen ebenfalls auf der konditionalen Verteilung der Features, allerdings werden die Differenzen in den Prädiktionen anstelle der Mittelwerte gebildet und stellen damit den reinen Effekt ohne Einfluss des Effekts korrelierter Features bereit. Um noch einmal den Unterschied zwischen den drei grafischen Darstellungen anhand von PDPs, M-Plots und ALEPs zu verdeutlichen und diese in einen mathematischen Kontext einzubetten, können die nachstehenden Gl. 5.4, 5.5 und 5.6 herangezogen werden.

$$\hat{f}_{\boldsymbol{x}_S}^{PDP}(\boldsymbol{x}_S) = \mathbb{E}_{\boldsymbol{X}_C}[\hat{f}(\boldsymbol{x}_S, X_C)] = \int_{\boldsymbol{x}_C} \hat{f}(\boldsymbol{x}_S, \boldsymbol{x}_C)\mathbb{P}(\boldsymbol{x}_C)d\boldsymbol{x}_C \tag{5.4}$$

$$\hat{f}_{\boldsymbol{x}_S}^{M}(\boldsymbol{x}_S) = \mathbb{E}_{\boldsymbol{X}_C|X_S}[\hat{f}(\boldsymbol{X}_S, \boldsymbol{X}_C)|\boldsymbol{X}_S = \boldsymbol{x}_s] = \int_{\boldsymbol{x}_C} \hat{f}(\boldsymbol{x}_S, \boldsymbol{x}_C)\mathbb{P}(\boldsymbol{x}_C|\boldsymbol{x}_S)d\boldsymbol{x}_C \tag{5.5}$$

ALEPs mitteln im Gegensatz dazu nun allerdings die Änderungen in den Prädiktionen und akkumulieren diese über das jeweilige Grid. Entsprechend zeigt sich der Unterschied zur vorangehenden Gl. 5.5 durch Mittelung der Änderungen in den Prädiktionen und durch das Mitteln der Prädiktionen selbst. Hierbei definiert sich diese Änderung als Gradient $\hat{f}^S(\boldsymbol{x}_S, \boldsymbol{x}_C)$. Außerdem gilt es ein zusätzliches Integral über $\boldsymbol{z}$ zu berücksichtigen und schließlich eine Konstante *const* von dem sich ergebenden Term zu subtrahieren, um den ALEP zu zentrieren, sodass der mittlere Effekt über die Daten Null ist.

$$\begin{aligned} \hat{f}^{ALE}_{\boldsymbol{x}_S}(\boldsymbol{x}_S) &= \int_{\boldsymbol{z}_{0,1}}^{\boldsymbol{x}_S} \mathbb{E}_{\boldsymbol{X}_C|\boldsymbol{X}_S}[\hat{f}^S(\boldsymbol{X}_S, \boldsymbol{X}_C)|X_S = \boldsymbol{z}_s] d\boldsymbol{z}_S - const \\ &= \int_{\boldsymbol{z}_{0,1}}^{\boldsymbol{x}_S} \int_{\boldsymbol{x}_C} \hat{f}^S(\boldsymbol{z}_S, \boldsymbol{x}_C) \mathbb{P}(\boldsymbol{x}_C|\boldsymbol{z}_S) d\boldsymbol{x}_C \; d\boldsymbol{z}_S - const, \end{aligned} \tag{5.6}$$

$$mit\ \hat{f}^S(\boldsymbol{x}_S, \boldsymbol{x}_C) = \frac{\delta \hat{f}(\boldsymbol{x}_S, \boldsymbol{x}_C)}{\delta \boldsymbol{x}_S}.$$

Auf einer genauere Vertiefung im Hinblick auf ALEPs und insbesondere deren Schätzung, sowie den Umgang mit kategorialen Features sei an dieser Stelle verzichtet. Für genauere Ausführungen sei auf Molnar (2018, Kap. 5.3) und insbesondere Apley (2016) verwiesen. Zusätzlich sei angemerkt, dass dennoch - wenngleich ALEs unverzerrt sind im Falle korrelierter Features - weiterhin die Problematik der schweren Interpretierbarkeit stark korrelierter Features besteht, da es einzig Sinn macht für die Analyse eines Effektes beide Features zu ändern und nicht eines in Isolation zu betrachten.
Erneut sei an dieser Stelle nun der Effekt von der Anzahl an Leukozyten auf die IHM betrachtete, wobei dieser Effekt nunmehr auf der konditionalen anstelle der marginalen Verteilung beruht, sodass insbesondere im Falle stark korrelierter Features stabiler Schätzungen hervorgebracht werden können. Auch hier kann wie mittels des PDPs mit der Steigerung der Leukozyten über etwa 15 G/l eine steigendes Risiko im Hinblick auf die IHM ausgemacht werden. Wenngleich die zugehörigen Wahrscheinlichkeitsangaben recht gering ausfallen, sollten diese doch nicht verachtet werden. Insbesondere sollte hier auch kurz erwähnt werden, dass in der Medizin von erhöhten Infektionsparametern ab einem Wert größer 12 G/l und/oder einer Körpertemperatur größer 38°C gesprochen wird, was in 6 noch genauer zur Sprache kommt. V.a. ist hier nun aber Bezug nehmend auf die Einflussvariable Fieber die Frage nach potentiell existierenden Interaktionseffekten und stochastischer Abhängigkeit zwischen den jeweiligen Prädiktoren zu stellen. Gemäß Apley (2016) ist unter derartigen Bedingungen die Aussagekraft von ALEs fraglich.

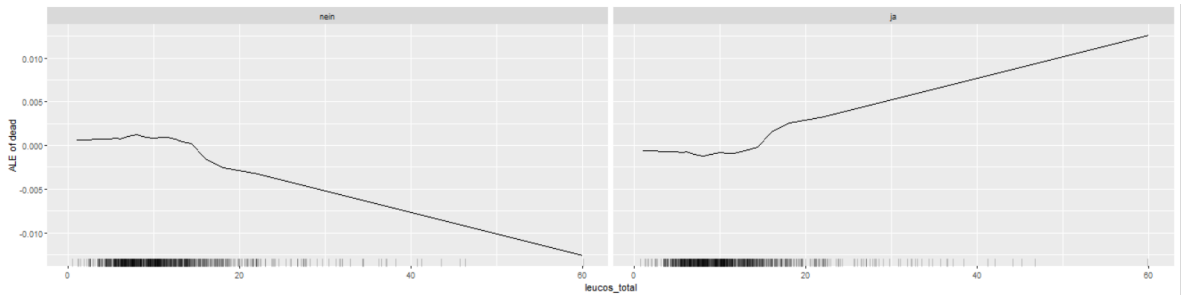


Abb. 5.3: Beispielhafte Veranschaulichung zu ALE Plots, cForest: Visualisierung der ALEs von der Anzahl an Leukozyten auf die prognostizierter IHM (nein/ja).

5.3.4 Feature Importance

Wie der Name dieser Interpretationsmethodik bereits vermuten lässt, soll über die „Feature Importance“ - nachfolgend lediglich mit Importance bezeichnet - Aufschluss über die Wichtigkeit einer Variable in Anbetracht ihres Einflusses auf die Zielgröße geben werden. Hierbei gestaltet sich das Vorgehen zur Berechnung der Importance recht intuitiv: man misst die sich durch Permutation der Werte eines Features ergebende Steigerung des Prädiktionsfehlers durch das Modell. Mittels der Permutation der Werte der betrachteten Einflussgröße wird entsprechend der Zusammenhang zwischen dieser und dem tatsächlichen Outcome aufgehoben. Demgemäß kann geschlussfolgert werden, dass ein Feature genau dann von Relevanz ist, wenn die Permutation eine Erhöhung des Modellfehlers hervorruft, da dann das Modell für neue Prädiktionen auf diesem Feature aufbaut. Schließlich lässt sich in einfacher Weise der zugrundeliegende Algorithmus zur Ermittlung der Importance folgendermaßen ausdrücken:

Algorithmus 5.1 Feature Importance in Anlehnung an Fisher et al. (2018)

1: Input: Trainiertes Modell $\hat{f}$, Vektor mit Werten der Zielgröße y, Feature-Matrix $\boldsymbol{X}$, Fehlermaß $L(\boldsymbol{y}, \hat{f})$
2: Schätzung des originalen Modellfehlers (z.B. MSE): $\epsilon_{orig} = L(\boldsymbol{y}, \hat{f}(\boldsymbol{X}))$
3: **for** j $= 1 \rightarrow$ p **do**
4: Generierung der permutierten Feature-Matrix $\boldsymbol{X}_{perm}$ durch Permutation des Features j in der originalen Datenmartix $\boldsymbol{X}$ zur Auflösung des Zusammenhangs zwischen dem j-ten Feature und der Zielgröße $\boldsymbol{y}$
5: Schätzung des Modellfehlers nach Permutation: $\epsilon_{perm} = L(\boldsymbol{y}, \hat{f}(\boldsymbol{X}_{perm}))$
6: Berechnung der Feature Importance über den Quotienten $FI_j = \epsilon_{perm}/\epsilon_{orig}$ oder alternativ über die Differenz $FI_{j,Diff} = \epsilon_{perm} - \epsilon_{orig}$
7: **end for**
8: Output: FI's der einzelnen Features (absteigend) sortiert

Diese Idee lässt sich auf Breiman (2001) zurückführen, welcher RFs als Blackbox-Methode dem Nutzer zugänglicher machen wollte. Darauf aufbauend schafften Fisher et al. (2018) eine Modell-agnostische Version dieser Methode, wobei in diesem Rahmen stets von „Model Re-

liance“ gesprochen wird und somit der Grad der Zuverlässigkeit eines Modell's im Mittelpunkt des Interesses steht. Es gilt zu bedenken, dass mittels dieser globalen Interpretationsmethode automatisch Interaktionen der Features untereinander Berücksichtigung finden. Durch die Permutation werden entsprechend neben den Haupteffekten der Features zusätzlich die Interaktionseffekte auf die Modellperformance bedacht. Dies kann auch als nachteilig angesehen werden, nachdem die Importance eines Features entsprechend größer ausfällt, insofern es in Interaktion mit einem anderen Feature steht. Damit spiegelt sich nicht der „reine“ Verlust hinsichtlich der Modellperformance bei Permutation der betrachteten Einflussgröße wider.
Beispielhaft kann also Abb. 5.4 herangezogen um den grafischen Output bezüglich der Importance zugänglich zu machen. Dabei wird jeweils für das permutierte Feature der Verlust anhand des „Classification Errors“ gemessen, sodass sich im Mittel der höchste Modellfehler für die binären Variablen Kreatinin und vorhergehende Cholangitis-Episode einstellen würden, weshalb diesen innerhalb des cForest's mitunter eine hohe Importance zugesprochen wird. Dabei geht die Permutation der beiden Einflussgrößen im Mittel mit einer Steigerung des Fehlers in den Prädiktionen um den Faktor 1.25 einher. Hingegen scheint beispielsweise die kontinuierliche Variable des Alters nicht von größerer Relevanz für die Vorhersagen der IHM innerhalb des Modells zu sein.

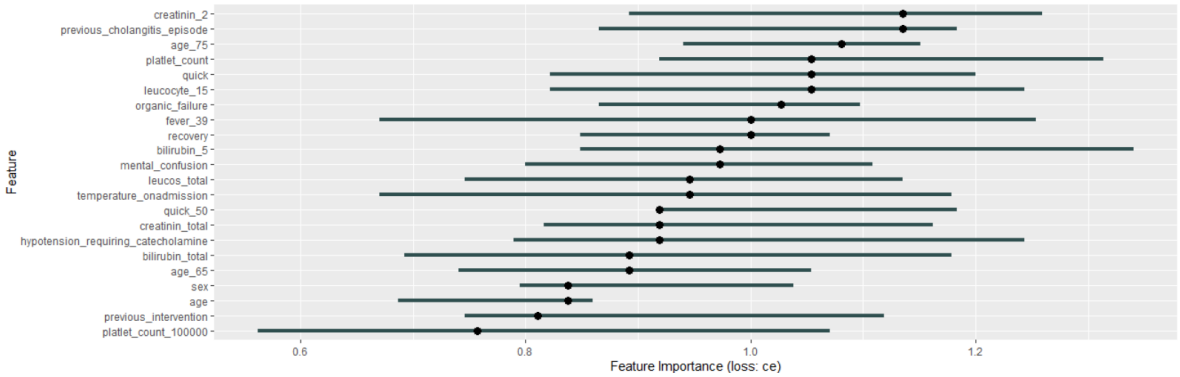


Abb. 5.4: Beispielhafte Veranschaulichung zu Feature Importance, cForest: Assoziation der Features gemessen anhand der Höhe des Modellfehlers bei der Prädiktion von IHM (nein/ja) durch Permutation des jeweiligen Features.

Letztlich können auch für die Nutzung der Importance klare Vorteile, sowie aber auch Nachteile und Unklarheiten verzeichnet werden. So sei an dieser Stelle noch erwähnt, dass einige Fragen hinsichtlich der Verwendung von Test- oder Trainingsset offen sind, sowie außerdem durch die Permutation eine Komponente der Zufälligkeit eingeführt wird, wodurch sich bei mehrfacher Wiederholung stark variierende Ergebnisse zeigen können. In Anbetracht letzteren genannten Punktes kann es sich also anbieten eine Stabilisation der Ergebnisse durch mehrfache Wiederholung und anschließendes Mitteln der Importance zu erreichen, womit jedoch ein hoher computationaler Aufwand einhergeht.

5.3.5 Feature Interaction

Bereits in vorheriger Untersektion fand die Begrifflichkeit der Interaktion zwischen Features Anklang, wobei der Einfachheit halber im Folgenden stets nur der Begriff der Interaktion gebraucht wird. Somit kann es durchaus auch von Interesse sein die Stärke des Zusammenhangs zwischen einzelnen Prädiktoren zugänglich zu machen. Treten Interaktionen auf, so kann der Effekt eines Features auf die Zielgröße also nicht als „rein" angesehen werden, da zusätzlich die Stärke des Zusammenhangs mit einem oder mehreren anderen Features bedacht werden muss. Allein bei zwei potentiellen Einflussgrößen gilt es somit den Effekt auf den Output in vier Summanden zu zerlegen: einen konstanten Term, jeweils einen Term für die beiden Features, sowie einen für deren Interaktion. Molnar (2018, Kap. 5.4) beschreibt dabei die Interaktion als Veränderung in der Prädiktion einer ML-Methode, die in Erscheinung tritt durch Variation der Features - nachdem die individuellen Effekte der Features ermittelt wurden.
Eine Möglichkeit die Stärke der Interaktion zu messen besteht nun darin zu ermitteln, wie sehr die Variation innerhalb der Prädiktion von der Interaktion der Features abhängt. Ein derartiges Maß ist die s.g. H-Statistik, welche von Friedman und Popescu (2008) eingeführt wurde. Theoretisch gesehen kann man sich so mit der Interaktion beliebig vieler potentieller Einflussgrößen auseinandersetzen. Zunächst soll nun allerdings ein zweiseitiges Maß zur Ermittlung, ob und wenn mit welcher Stärke eine Interaktion zwischen zwei Features vorliegt, dargeboten werden.
Insofern zwei Features nicht miteinander interagieren, kann deren zweiseitige partielle Abhängigkeitsfunktion („Partial Dependence" = PD) $PD_{jk}(\boldsymbol{x}_j, \boldsymbol{x}_k)$ - und damit wiederum Bezug nehmend auf 5.3.1 - in die PD-Funktionen der beiden einzelnen Features zerlegt werden:

$$PD_{jk}(\boldsymbol{x}_j, \boldsymbol{x}_k) = PD_j(\boldsymbol{x}_j) + PD_k(\boldsymbol{x}_k) \tag{5.7}$$

Ausweitend gilt dann, dass - falls ein Feature keinerlei Interaktion mit einer der anderen Einflussgrößen aufweist - die Prädiktionsfunktion $\hat{f}(\boldsymbol{x})$ als Summe aller PD-Funktionen dargestellt werden kann, wobei der erste Summand einzig von dem betrachteten j-ten Feature abhängt und der zweite von allen Features, ausgeschlossen dem j-ten. Letzteres wird durch PD_{-j} zum Ausdruck gebracht und Gl. 5.8 umfasst die zugehörige PD-Funktion die sich bei fehlender Interaktion des j-ten mit den übrigen vorliegenden Features egribt.

$$\hat{f}(\boldsymbol{x}) = PD_j(\boldsymbol{x}_j) + PD_{-j}(\boldsymbol{x}_{-j}) \tag{5.8}$$

Auf Basis dessen lässt sich nun eine Statistik als Maß der Interaktionsstärke entwickeln, welche sich folgendermaßen erklären lässt: man messe den Unterschied bzw. die Differenz zwischen der beobachteten PD-Funktion und den einzelnen zerlegten, also diejenigen, die sich bei fehlender Interaktion ergeben. Je nach Fokus berechne man dann die Varianz des Outputs der PD- (für die Interaktion zwischen zwei Features) oder der gesamten Funktion (für die Interaktion

zwischen einem Feature und den verbleibenden). Nun lässt sich die Stärke der Varianz, welche sich durch die Interaktion erklären lässt und damit die Differenz zwischen beobachteter und PD ohne Interaktion, als Statistik zur Kalkulation der Interaktionsstärke verwenden.
Diese Statistik soll genau dann 0 sein, wenn keine Interaktion vorliegt und 1, falls die gesamte Varianz von PD_{jk} bzw. $\hat{f}$ durch die Summe der einzelnen PD-Funktionen beschrieben werden kann. Damit ergeben sich in Anlehnung an Friedman und Popescu (2008) nachstehende Gleichungen für die Interaktion des j-ten Features mit dem k-ten bzw. mit allen verbleibenden Einflussgrößen:

$$\begin{aligned} H_{jk}^2 &= \sum_{i=1}^{N} \Big[PD_{jk}(\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_k^{(i)}) - PD_j(\boldsymbol{x}_j^{(i)}) - PD_k(\boldsymbol{x}_k^{(i)}) \Big]^2 / \sum_{i=1}^{N} PD_{jk}^2(\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_k^{(i)}), \\ H_j^2 &= \sum_{i=1}^{N} \Big[\hat{f}(\boldsymbol{x}^{(i)}) - PD_j(\boldsymbol{x}_j^{(i)}) - PD_{-j}(\boldsymbol{x}_{-j}^{(i)}) \Big]^2 / \sum_{i=1}^{N} \hat{f}^2(\boldsymbol{x}^{(i)}) \end{aligned} \tag{5.9}$$

Diese ML-Methode kann beispielhaft wie in Abb. 5.5 visualisiert werden, wobei hier die Interaktion jedes Features mit alle übrigen Prädiktoren in Betracht gezogen ist. In diesem Falle von binärer Klassifikation wäre es natürlich ausreichend eine Grafik zu präsentieren, dennoch lassen sich einige Feststellungen machen. So kommen generell einige stärkere Interaktionen zutage, wobei die meisten weniger als 10 % zur Erklärung der Varianz beitragen können. Jedoch sind insbesondere der Zustand mentaler Verwirrtheit, welcher über 35 % der Varianz des cForests erklärt, sowie das Vorliegen von Organversagen auffallend.

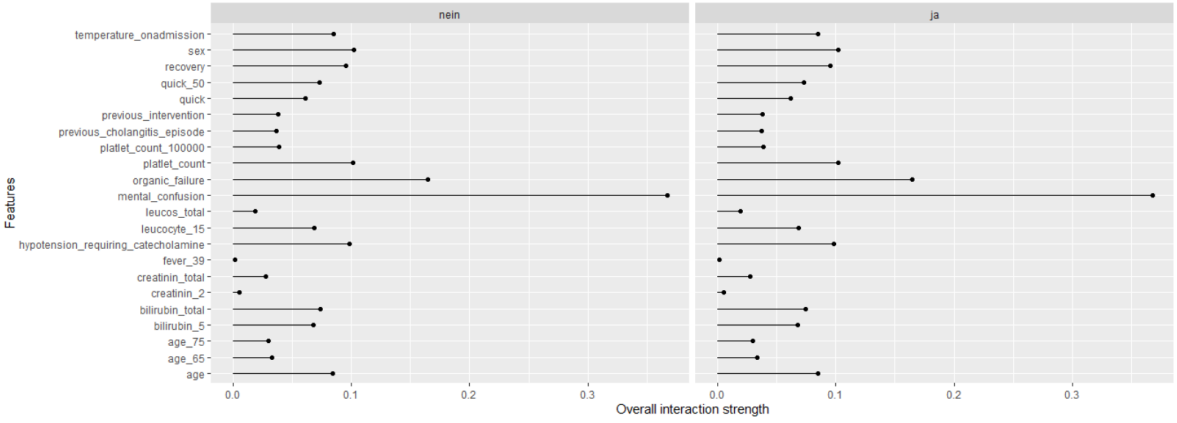


Abb. 5.5: Beispielhafte Veranschaulichung zu Feature Interaktion, cForest: Interaktionsstärke (H-Statistik) für jedes Feature mit allen übrigen zur Prognose der IHM (nein/ja).

Ein deutlicher Vorteil zeigt sich also durch die Dimensionslosigkeit der Statistik - definiert auf einem Intervall zwischen $[0, 1]$ -, sodass diese über Features hinweg und sogar verschiedene Modelle verglichen werden kann. Wie jedoch anhand von 5.9 bereits ersichtlich werden kann, ist die s.g. H-Statistik zeitlich gesehen teuer in ihrer Berechnung, nachdem über alle Datenpunkte iteriert wird und an jedem dieser Punkte mittels aller N Punkte die PD berechnet

wird. Um dies zu beschleunigen, kann beispielsweise nur ein Sample für die Kalkulation herangezogen werden, was dann wiederum aber den Nachteil mit sich führt, dass sich die Varianz der PD-Schätzung vergrößert.
Friedman und Popescu (2008) entwickelten zudem eine Teststatistik, um zu ermitteln, ob die H-Statistik signifikant verschieden von Null ist, wobei die Nullhypothese das Fehlen der Interaktion(en) annimmt. Jedoch muss man für die Kalkulation der Interaktionsstatistik unter der Nullhypothese das zugrundeliegende Modell derartig anpassen können, dass Interaktionen zwischen dem j-ten und einem weiteren Feature bzw. allen übrigen entfernt werden können. Dies kann nicht für jeden Typen der ML-Methoden vorgenommen werden, womit der Test wenn dann Modell-spezifisch angewandt werden kann und man sich damit außerhalb des Rahmens Modell-agnostischer Mittel bewegt. Zudem sei anzumerken, dass mittels der H-Statistik zwar die Stärke von Interaktionen erkannt werden kann, jedoch nicht welche Form diese Interaktionen zwischen den Features aufweist.

5.3.6 Surrogate Interpretationsmethoden - global und lokal (LIME)

Eine weitläufige Verwendung zur Interpretierbarkeit von Blackbox-Methoden weisen s.g. Surrogate Techniken auf, wobei hierzu auf Molnar (2018, Kap. 5.6 und 5.7) zu verweisen ist. Die zugrundeliegende Idee soll hier nur umrissen werden und nicht im genaueren Sinne vertieft werden. Die Nutzung surrogater Modelle fand bereits mehrfach Anklang innerhalb dieser Arbeit und wird beispielsweise auch im Rahmen der (S)MBO, welche innerhalb der Untersektion 3.3.2 Erklärung findet, eingesetzt. Im Fokus dieser surrogaten Modelle steht im Falle von IML eine möglichst gute Annäherung an die Prädiktionen des zugrundeliegenden Modells. Gleichzeitig soll dieses surrogate Modell einfache Interpretierbarkeit bereitstellen, weshalb beispielsweise auf Modelle der linearen oder logistischen Regression zurückgegriffen wird, ebenso wie auf GLMs und Entscheidungsbäume.
Letztendlich wird also in Form einer Art Metamodells ein derartig interpretierbares Modell auf die Prädiktionen der Blackbox-Methode angewandt. Zunächst zieht man also einen Datensatz $\boldsymbol{X}$ heran, bei welchem es sich sowohl um die Trainingsdaten des Blackbox-Modells handeln kann, als auch um neue Daten mit gleicher Verteilung. Für den gewählten Datensatz werden die Prädiktionen, welche sich durch Anwendung der zu interpretierenden Methode ergeben, geschätzt. Anschließend wird das Metamodell auf den Datensatz $\boldsymbol{X}$ und die Prädiktionen der Blackbox-Methode $\hat{\boldsymbol{y}}$ angewandt und so beispielsweise die β-Koeffizienten eines linearen Modells ermittelt, die ihrerseits als Steigungsparameter interpretiert werden können.
So kann nun beispielsweise einen Entscheidungsbaum als surrogates Modell für die Modellierung der Prädiktionen des cForest bezüglich der IHM genutzt werden. Aufgrund der vereinfachten Darstellung wurden hier 2 Splits festgesetzt. Damit zeigt sich nun anhand von Abb. 5.6, dass der erste Split durch Gruppierung der Patienten in diejenigen mit oder ohne mentaler Verwirrtheit vorgenommen wurde. Falls diese vorliegt rät der Entscheidungsbaum zu einer

weiteren Unterscheidung gemäß eines Bilirubin-Wertes größer oder kleiner 6.7 mg/l an. Anderenfalls sollte das Vorliegen von Organversagen als Indiz bedacht werden, um Klassifikationen bezüglich der IHM vornehmen zu können.

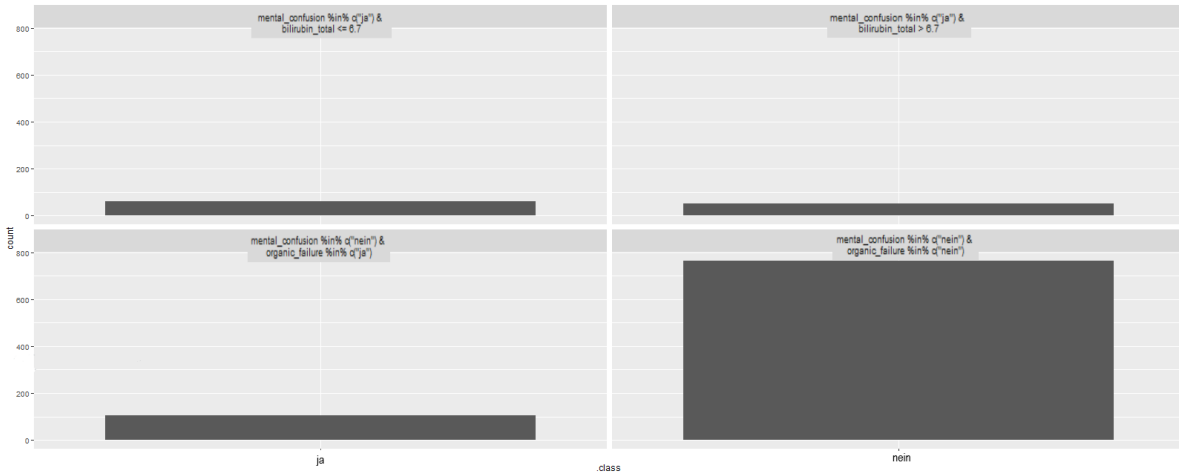


Abb. 5.6: Beispielhafte Veranschaulichung zu Surrogatem Modell (global), cForest: Terminale Knoten eines surrogaten Entscheidungsbaumes, der die Prädiktionen des cForest's modelliert, mit Darstellung der Häufigkeiten von Klassifikationen der Blackbox-Methode innerhalb der einzelnen Knoten.

Gleichzeitig sollte aber auch nicht außer Acht gelassen werden, die Güte des interpretierbaren Modells zu ermitteln, um den Grad der Interpretierbarkeit bzw. dessen Verlässlichkeit ausmachen zu können. In diesem Kontext bietet sich dann beispielsweise das Bestimmtheitsmaß R^2 an, welches bereits in Untersektion 3.4.4 beschrieben wurde. Entsprechend kann durch ein R^2, welches nahe 1 liegt, auf eine gute Interpretierbarkeit mittels des surrogaten Modells geschlossen werden, wobei sich die Frage stellt, ob die komplexe ML-Methode nicht durch das einfachere Metamodell ersetzt werden kann. Andersherum gilt für ein R^2 nahe 0, dass das surrogate Modell einen hohen SSE aufweist und die Blackbox-Methode nicht oder nur sehr spärlich beschreiben kann.
Die bisherige Beschreibung surrogater interpretierbarer Modelle greift auf globaler Ebene an. Genauso können diese jedoch auch auf lokale Ebene überführt werden. Dazu betrachte man ein Subset des originalen Datensatzes oder eine neue Gewichtung der Instanzen. Beides führt zu einer Änderung der Verteilung, womit sich auch der Fokus der Interpretation verlagert. Wenn man etwa eine lokale Gewichtung der Daten durch eine spezifische Instanz - je näher die Nachbarn an der betrachteten Instanz, desto größer deren Gewicht - vornimmt, so erhält man ein lokales surrogates Modell. Damit werden die individuellen Prädiktionen interessierender Instanzen zugänglich und man spricht in diesem Zusammenhang kurz von s.g. LIMEs - abgeleitet vom Englischen „Local Interpretable Model-agnostic Explanations“ -, welche sich auf Riberio et al. (2016) zurückführen lassen.
Die Grundidee bei LIME ist die Zugänglichkeit und das Verständnis der Änderung in Prädik-

tionen, wenn kleine Variationen an den Daten vorgenommen werden. Hierzu wird ein neuer Datensatz generiert, welche aus permutierten Samples und den zugehörigen Prädiktionen der Blackbox-Methode bestehen. Anschließend wird darauf ein interpretierbares Modell angepasst, welches gemäß der Nähe der gezogenen Instanzen zu der im Fokus stehenden Instanz gewichtet wird. Hierbei ist genau dann das angepasste Modell von hoher Güte, wenn dieses lokal eine gute Approximation an die Prädiktionen der ML-Methode liefert, jedoch muss dieser Aspekt nicht auf globaler Ebene erfüllt sein. Bei einer derartigen Beurteilung der Genauigkeit spricht man von lokaler „Fidelity".
Mathematisch kann dies nun folgendermaßen zum Ausdruck gebracht werden: Ausgangspunkt stellt eine Erklärung (explanation) bzw. ein interpretierbares Modell $g \in G$ dar, wobei G eine Klasse interpretierbarer Modelle entspricht. Nachdem jedoch auch die Interpretierbarkeit von g sich nicht immer gleichermaßen einfach darstellt, führe man zusätzlich $\Omega(g)$ ein, womit die Komplexität des surrogaten Modells zum Ausdruck kommen soll. Diese kann sich beispielsweise in der Tiefe eines Entscheidungsbaumes darbieten. Das eigentlich im Fokus des Interesse stehende Blackbox-Modell wird wiederum mit f bezeichnet, sowie nun auch ein Maß der Nähe bzw. des Abstandes der fokussierten Instanz x und einer weiteren Instanz z durch $\pi_x(z)$ eingeführt wird. Gleichzeitig stellt aber auch die Festsetzung der Distanz π_x, welche als Maß für die in nächster Nachbarschaft zu berücksichtigenden Instanzen darstellt, eine der größten Herausforderungen dar und legt eine sehr vorsichtige Verwendung von LIME nahe. Das erklärende Modell g für Instanz x ergibt sich dann durch Minimierung der Verlustfunktion $\mathcal{L}$, welche die Nähe der Erklärung zu der Prädiktion der Blackbox-Methode f misst, unter gleichzeitig möglichst geringer Komplexität des surrogaten Modells $\Omega(g)$. Während die gewählte Verlustfunktion gewählt werden kann, ist es Aufgabe des Anwenders die Komplexität des erklärenden Modells vorab festzusetzen und verlangt damit einen Kompromiss zwischen Sparsamkeit und Fidelity.

$$explanation(x) = argmin_{g \in G}\, \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{5.10}$$

Zusätzlich zeigen Riberio et al. (2016) bei aller Anschaulichkeit dieser lokalen Interpretationsmethodik einen weiteren kritischen Aspekt auf: die Instabilität der Erklärungen. So zeigten sich innerhalb von Simulationssettings große Variationen der Erklärungen für zwei sehr nahe beieinander liegende Punkte. Zusammenfassend eignet sich LIME insbesondere für das Verständnis des lokalen Verhaltens von Blackbox-Methoden, doch gilt es Rückschlüsse und Interpretationen mit Vorsicht anzustellen.

Bisher fanden einzig Modell-agnostische Methoden Beschreibung, da diese das Kernstück des *iml* Paketes und damit auch von *shinyBMR* darstellen. Kurze Beschreibungen zu den hier nicht weiter fokussierten Beispiel-basierten Erklärungsmethoden finden sich im Anhang bei Sektion 8.4.1 oder können anhand von Molnar (2018, Kap. 6) nachvollzogen werden.

6 Durchführung einer Benchmark-Studie basierend auf einem Datensatz zu akuter Cholangitis

Dieses Kapitel soll im Wesentlichen die Ergebnisse einer konkret durchgeführten Benchmark-Studie unter Nutzung der innerhalb dieser Arbeit präsentierten Methoden bereitstellen. Die zugehörigen Analysen finden ihre Anwendung auf einen Datensatz zu akuter Cholangitis, wobei der interessierende Outcome eine Klassifikation/Prädiktion der durch dieses Krankheitsbild bedingte Todesfälle bzw. In-Hospitalen Mortalität (IHM) darstellt. Zunächst sollen daher in Sektion 6.1 das Krankheitsbild als solches kurz beschrieben werden und anschließend ein erster Überblick über die vorliegenden Daten in Untersektion 6.1.2 vermittelt werden, womit zu deskriptiven, sowie univariaten Analysen in Untersektion 8.5.4 übergeleitet werden kann. Daran anschließend kommt in Sektion 6.2 der eigentliche Vergleich der ausgewählten ML-Methoden mittels einer Benchmark-Studie zur Sprache, indem zunächst Vorbereitunsgmaßnahmen in Untersektion 6.2.1 und schließlich die Auswertung des Benchmarkings in 6.2.2 Anklang finden. Abschließend wird in Untersektion 6.2.3 das hieraus resultierende Gewinner-Modell angepasst und die so bestimmten Effekte der einzelnen Prädiktoren präsentiert.

6.1 Daten zu Todesfällen bei akuter Cholangitis

6.1.1 Krankheitsbild der akuten Cholangitis

Cholangitis - oder zu deutsch Gallengangentzündung - beschreibt eine Entzündung der Gallenwege. Diese stellen die Verbindung zwischen der Gallenblase und dem Dünndarm bzw. genauer - einem Abschnitt dessen - dem Zwölffingerdarm dar. Um das Krankheitsbild als solches noch eingängiger verständlich zu machen, kann Sektion 8.5.1 des Anhangs herangezogen werden, worin der anatomischen Aufbau der Gallenwege, sowie der Transport der Galle Beschreibung findet. Zunächst soll hier nun aber der Frage nachgegangen werden, wie das Krankheitsbild der akuten Cholangitis ausfällt, wobei dies in Anlehnung an Vastmans (2019) und Mosler (2011) erfolgt. Generell gilt es eine chronische von einer akuten Krankheit abzugrenzen: erstere wird durch eine lang andauernde (griech. „chronos“) körperliche, psychische, soziale, o.ä. Überbelastung hervorgerufen, die sich zumeist manifestiert und potentiell lebenslang bestehen kann. Demgegenüber lässt sich eine akute Krankheit als vorübergehende Infektionserkrankung beschreiben, welche durch mikrobiologische Erreger - beispielsweise Viren oder Bakterien - hervorgerufen wird. Prinzipiell kann sich die akute Cholangitis zu einer lebensbedrohlichen Erkrankungen entwickeln, die durch die Kombination einer Infektion und der Behinderung der Gallengänge verursacht wird, wobei teilweise oder gänzlich der Fluss der Gallenflüssigkeit in den Zwölffingerdarm verhindert und damit einen Rückstau in die Leber („Gallenstase“) hervorgerufen wird. Dies geht mit unterschiedlichen Symptomen wie etwa

hohem Fieber, Gelbfärbung der Haut (Ikterus) oder Schmerzen im rechten Bereich des Bauches einher. Die klinischen Fälle reichen dabei von leichten Formen, welche durch die Gabe von Antibiotika behandelt werden können, hin zu lebensgefährlichen Erscheinungsbildern, die intensive Pflege und Gallendrainage verlangen.
In den letzten Jahrzehnten konnte die Todesrate durch akute Cholangitis zwar deutlich gesenkt werden, jedoch ist dieses gemäß Mosler (2011, S. 166) nach wie vor erheblich im Falle schwerer Formen und zudem unzureichender Behandlung. Frühzeitige Diagnosen auf Basis klinischer Symptome, Labor-, sowie bildgebender Befunde sind hierbei maßgeblich für die zeitnahe Intervention durch Behandlung. Das breite Feld möglicher Auslöser, bedingt gleichzeitig den Einfluss vieler potentieller Faktoren auf das Überleben der betroffenen Patienten.

6.1.2 Beschreibung der Studie und der Daten (-aufbereitung)

Die zuvor beschriebenen Umstände veranlassten schließlich die im Rahmen dieser Arbeit Einzug nehmende Studie zu akuter Cholangitis, welche bei Schneider et al. (2016) umfangreiche Analyse fand. Fokussiert wird dabei die Anpassung eines Risikomodells als Orientierungshilfe für Ärzte, um die Dringlichkeit der Intervention hinsichtlich der Druckentlastung der Galle, sowie eine Richtungsweisung bezüglich der Stärke medikamentöser Behandlung offenzulegen. Alle in dieser Studie berücksichtigten Patienten wurden an der II. Medizinischen Klinik und Poliklinik, Klinikum rechts der Isar, Technische Universität München aufgenommen. Jedem Studienteilnehmer konnte eindeutig - gemäß der Tokyo Guidelines 2013 (TG13, siehe Yokoe et al. 2013) - Cholangitis nachgewiesen werden. Auch die Definition von Organversagen wurde im Wesentlichen von TG13 abgeleitet, für dessen Vorliegen einer der folgenden Kriterien erfüllt sein musste: Hypotonie, die die Gabe von Katecholamine erfordert, Kreatinin im Blutserum größer 2 mg/dl, Quick-Wert - Parameter zur Beurteilung der Funktionsleistung der Blutgerinnung - kleiner 50%, Anzahl der Blutplättchen unter 100 000 pro mm^3 oder auftretende Bewusstseinsstörungen. Zudem mussten weitere Faktoren erfüllt sein, die einen Einschluss in die Studie garantierten, wie etwa erhöhte Temperatur (über 38 °C) und/oder erhöhte Infektionsparameter, worunter beispielsweise ein erhöhter Anteil an Leukozyten (über 12 G/l), sowie erhöhte Cholestasis-Parameter fallen. Gleichzeitig blieben Patienten, für die Daten nicht vollständig vorlagen - im Sinne von fehlenden Laborwerten oder medizinischen Untersuchungen -, unberücksichtigt, ebenso wie Patienten mit Leberzirrhose im Endstadium. Es verstarben 13 Studienteilnehmer aufgrund anderer Ursachen als Cholangiosepsis, weshalb diese nicht in die Analysen einbezogen sind. Die vollständige Beschreibung der Studienpopulation findet sich bei Schneider et al. (2016) und soll hier nun nicht weiter fokussiert werden.
Nicht alle erfassten Daten wurden für die Entwicklung eines Risikomodells zur Prognose der IHM bei akuter Cholangitis herangezogen. Vielmehr wurden Prädiktoren fokussiert, die unmittelbar bei Aufnahme ins Krankenhaus leicht zu erfassen sind, und so wurden in Absprache mit dem Betreuer dieser Arbeit A. Hapfelmeier eine Auswahl zu fokussierender Einflussgrößen

festgesetzt, die bereits bei Schneider et al. (2016) herangezogen sind. Für die konkrete Analyse der Daten wurden weitere Bearbeitungen vorgenommen, wobei diese im Wesentlichen von Schneider et al. (2016) übernommen sind. Alle als Prädiktoren genutzten Größen sind Tab. 6.2 zu entnehmen. Hierzu sei vermerkt, dass alle numerischen Variablen auch in dichotomisierter Form aufgenommen sind. Im Zuge der Dichotomisierung wurde u.a. die Variable des Alters auf zweierlei Arten zusätzlich berücksichtigt: So wurde eine Unterteilung in Patienten, die älter oder gleich 65 Jahre sind, sowie in Patienten, die älter oder gleich 75 Jahre sind, getroffen. Erstere Gruppierung stellt gemäß Schonberg et al. (2011) eine gängige Methode in der Medizin zur Unterscheidung zwischen älteren und alten Personen dar. Zweitere kann von TG13 abgeleitet werden, ebenso wie auch die Dichotomisierung der Variable Fieber durch Temperaturen größer oder gleich 39 °C und der Variable Hyperbilirubinämie - Erhöhung der Bilirubinkonzentration im Blut -, welche ab Werten größer oder gleich 5 mg/dl vermerkt wird.

Dichotome Prädiktoren (*Schwellenwert*)	**Numerische Prädiktoren**
Alter ($\geq$ 65 Jahre) Alter ($\geq$ 75 Jahre)	Alter in Jahren
Temperatur/Fieber ($\geq$ 39 °C)	Temperatur/Fieber in °C
Anzahl der Leukozyten ($>$ 15 G/l)	Anzahl der Leukozyten in G/l
Bilirubin-Gehalt ($\geq$ 5 mg/dl)	Bilirubin-Gehalt in mg/dl
Quick's Wert ($<$ 50%)	Quick's Wert in %
Serumkreatinin ($\geq$ 2 mg/dl)	Serumkreatinin in mg/dl
Anzahl an Blutplättchen ($<$ 100 G/l)	Anzahl an Blutplättchen in G/l
Geschlecht (männlich, weiblich)	
Mentale Verwirrtheit (ja, nein)	
Vorhergehende Cholangitis Episode (ja, nein)	
Vorhergehende Intervention (ja, nein)	
Katechoalamine erfordernde Hypotension (ja, nein)	
Organversagen (ja, nein)	
Ursache der Cholangitis (maligne, nicht-maligne)	

Tab. 6.1: Als Prädiktoren für die Anpassung des Modells zur Prognose der Mortalität bei akuter Cholangitis herangezogenen (dichotome und numerische) Variablen

Für genauere Veranschaulichungen können die Histogramme in Abb. 8.9 des Anhangs herangezogen werden. Zusätzlich sei vermerkt, dass die Prädiktoren Fieber und Quick's Wert - jeweils 24 und 14 - fehlende Werte innerhalb des ursprünglichen Datensatzes aufwiesen. Nachdem einige Methodiken des ML's nicht in der Lage sind mit diesen fehlenden Werten umzugehen, wurde auf das Hilfsmittel der Datenimputation über das Paket *missForest* (Version 1.4) zurückgegriffen. Genauere Ausführungen hierzu finden sich in Untersektion 8.5.2.

6.1.3 Deskriptive und univariate Analysen

Insgesamt umfasst der in dieser Arbeit fokussierte Datensatz 981 Episoden/Beobachtungen zu akuter Cholangitis, die bei den 810 inkludierten Studienteilnehmern im Alter von 24 bis 97 (Median 86) Jahren vermerkt werden konnten. Hierbei sind mehr männliche (456/810≙56.3%), als weibliche (354/810≙43.7%) Patienten inbegriffen. Jeder der Patienten wurde entweder einer Endoskopisch Retrograde Cholangiographie (ETC, 671/981≙68.40%) oder einer Perkutane Transhepatische Cholangiographie (PTC, 310/981≙31.60%) unterzogen.
Zurückzuführen sind die Episoden akuter Cholangitis dabei auf drei Ursachen: maligne (bösartige) in 51.9% der Fälle und benigne (gutartige) Genese in 39.9% der Fälle, sowie idiopathische Gallenstriktur, die sich bei 8.3% der Patienten als auslösender Faktor erkennen ließ. Letzterer Gesichtspunkt beschreibt eine Gallenverengung, deren (gutartige) Ursache idiopathisch/unbekannt ist. Bei nachfolgenden Analysen wird eine ausschließliche Betrachtung maligner und nicht-maligner Ursachen unterschieden, sodass die Gruppen der benignen Genese und idiopathischen Gallenstriktur zusammengefasst werden. Insgesamt lässt sich auf Basis der vorliegenden Daten eine Mortalität von 4.5% verzeichnen, wobei der Tod der Patienten direkt auf Cholangiosepsis zurückzuführen ist.
Ausweitend wurde eine univariate Analyse in Form des Fisher χ^2-Tests - bzw. Exaktem Test nach Fisher - durchgeführt, um einen ersten Eindruck bzgl. der Assoziationen binärer und dichotomisierter Einflussgrößen mit der IHM zu gewinnen. Diese lässt sich wiederum auch von Schneider et al. (2016) ableiten und die Ergebnisse können in Form von Odds Ratios (OR), sowie dem zugehörigen 95%-Konfidenzintervall (KI) der Tab. 8.3 im Anhang entnommen werden. Angemerkt sei, dass statistische Tests auf einem explorativen, zweiseitigen 5% Signifikanzlevel durchgeführt wurden. Zudem ist ein direkter Vergleich der Risikofaktoren anhand des OR's möglich, da diesen eine gemeinsame Skala zugrunde liegt, in dem Sinne, dass diese binär bzw. dichotomisiert gemessen sind. Hierbei wurden bei der Durchführung der χ^2-Tests die Assoziation aller binären Größen mit der Zielvariable des Todesereignisses in Betracht gezogen, also nicht nur die in Tab. 6.2 gelisteten Prädiktoren berücksichtigt.
Im Besonderen sollte hervorgehoben werden, dass Organversagen den weitaus größte Zusammenhang ($p < 0.001$; OR: 47.1; KI: $[14.3 - 155.1]$) mit der IHM aufweist. Unter denjenigen Einflussgrößen, die u.a. auf ein Organversagen hindeuten, lassen in absteigender Reihenfolge mentale Verwirrtheit ($p < 0.001$; OR: 37.6 $[16.7 - 84.9]$), ein Quick's Wert kleiner 50% ($p < 0.001$; OR: 6.4 $[2.9 - 14.5]$), Serumkreatinin größer 2 mg/dl^3 ($p = 0.012$; OR: 4.3 $[1.5 - 11.8]$), sowie die Anzahl der Blutplättchen kleiner 100 000/mm^3 ($p = 0.038$; OR. 3.0 $[1.1 - 8.2]$) eine hohe Assoziation mit der IHM vermerken. Neben alledem stehen die dichotomisierten Größen Bilirubin $\geq$ 5 mg/l, Aspartat-Aminotransferase (AST) $>$ 70, PTC als therapeutische Maßnahme und das Vorliegen von Bakteriämie oder einer unzureichenden Drainagierung signifikant in Zusammenhang mit erhöhter IHM, wobei sich für die genauen Maßzahlen bzw. ORs mitsamt KI auf Tab. 8.3 berufen werden kann. Die univariate Ana-

lyse abschließend kann noch festgehalten werden, dass Patienten mit maligner Ursache der Cholangitis eine höhere Inzidenz hinsichtlich der IHM aufweisen im Vergleich zu denjenigen Patienten mit benigner oder idiopathischer Genese ($p < 0.004$; OR: 3.0 $[1.3 - 6.]$).

6.2 Vergleich von ML-Methoden mittels einer Benchmark-Studie

Nun folgend gilt es eine ML-Methode zu ermitteln, die dazu befähigt möglichst genau Prädiktionen in Anwendung auf den vorliegenden Datensatz zu akuter Cholangitis hervorzubringen. Eben hier setzt die zuvor bereits ausführliche beschriebene Benchmark-Studie an, mittels solcher verschiedene Methoden des ML's durch konkrete Probeläufe verglichen und anhand deren Performance beurteilt werden können. In diesem Zuge wurde nun auch auf einen Großteil der bereits - insbesondere in Kap. 3 - beschriebenen Hilfsmittel bzw. Werkzeuge zurückgegriffen.

6.2.1 Vorbereitungen und Einstellungen in Anbetracht der Benchmark-Analyse

So erfolgt die Begutachtung der Leistung auf Basis von AUC und pAUC, welche in Sektion 3.4 als Gütemaße dargeboten sind, und ist damit als multi-kriterielles Optimierungsproblem zu erachten. Die Wahl des AUC's als Beurteilungsmaß lässt sich insbesondere darin begründen, dass das AUC als Skalen-invariant, sowie Threshold-invariant ist. Ersteres stellt also eine Beurteilung anhand von Rängen anstatt von absoluten Werten sicher und Letzteres garantiert die Unabhängigkeit des Maßes von einem spezifisch gewählten Threshold. Augenmerk wird nachfolgend jedoch insbesondere auf das von diesem Maß abgeleitete partielle AUC gesetzt. Bereits am Rande erwähnt wurde die Tatsache, dass eine starke Imbalance hinsichtlich der Verteilung der Zielgröße IHM vorliegt. So liegt die Mortalitätsrate bei 4.5 % und damit zeigt sich das Klassifikationsproblem als stark unausgeglichen hinsichtlich der positiven Klasse des Todesereignisses. Letztlich strebt man also an ML-Methoden zu bevorzugen, die eine möglichst hohe TPR bzw. Sensitivität in Anwendung auf vorliegende Daten erreichen. In Anbetracht dessen kann das pAUC als Gütemaß herangezogen werden und konzentriert sich bei der gegebenen Problemstellung auf einen Ausschnitt hoher TPR, sodass diese hier durch das Intervall $[c_1; c_2] = [0.8; 1.0]$ begrenzt ist. Daraus resultiert die Beschränkung des Definitionsbereich auf $pAUC \in [0; c_2 - c_1] = [0; 0.2]$. Angemerkt sei an dieser Stelle, dass auf eine Korrektur des pAUC's nach McClish (1989) verzichtet wird, nachdem Werte des pAUC's potentiell unterhalb der Diagonalen liegen können und damit die Korrektur ungültig bzw. nicht definiert ist. Unter Heranziehung von Gl. 3.7 würde die beschriebene, ungültige Korrektur von Kurven unterhalb der Winkelhalbierenden sich in $pAUC < min$ widerspiegeln. Nachdem das pAUC (bisher) nicht im *mlr* Paket bereitgestellt ist, wurde dies mittels bereitgestellter Funktionen des *mlr* Paketes unter gleichzeitiger Nutzung der entsprechenden Implementierung des pAUS's im *pROC* (Version 1.15.0) Paket von Robin et al. (2011) selbstständig als Gütemaß für weitere Analysen integriert.

Um nun letztlich eine möglichst große Bandbreite verschiedener ML-Methodiken, die sich im Klassifikationskontext anbieten, auszuschöpfen, wurden die nachfolgend gelisteten Lernalgorithmen herangezogen. Gleichzeitig sind die R-Pakete angegeben, auf denen der jeweilige in *mlr* implementierte Algorithmus fußt.

- k-Nearest-Neighbors **kNN** (s. Sektion 4.1); R-Paket: *kknn* (Version 1.3.1) von Schliep und Hechenbichler (2016).
- Support Vector Machines **svm** (s. Sektion 4.2); R-Paket: *e1071* (Version 1.7-2) von Meyer et al. (2019a).
- RF basierend auf Bäumen (rekursive Partitionierung) konditionaler Inferenz als Basis-Lerner **cTree** und **cForest** (s. Sektion 4.3 und Untersektion 4.4.2); R-Paket: *party* (Version 1.3-3) von Hothorn, Hornik und Zeileis (2006), Hothorn, Buehlmann, Dudoit, Molinaro und Van Der Laan (2006), Strobl et al. (2007), sowie Strobl et al. (2008). Die Implementierung im Rahmen der konditionalen Inferenz ist hier derart gewählt, dass das Stopp-Kriterium auf Basis der dem multiplem Testen adjustierten p-Werte (Test-Typ: Bonferroni) bestimmt wird. Es gilt dieses Kriterium zu maximieren, also $1 - p$.
- Boosting für GLM's **glmboost** (s. Sektion 4.4.3); R-Paket: *mboost* (Version 2.9-1) von Hothorn et al. (2018), Hofner et al. (2014), Hothorn et al. (2010), Buehlmann und Hothorn (2007), sowie Hofner et al. (2015). Entspricht der Anwendung von Gradient Boosting für die Optimierung einer wählbaren Verlustfunktion - hier: Binomialverteilung mit Logit-Link, sodass sich die negativ-binomiale log-Likelihood als Verlustfunktion ergibt -, wobei komponentenweise lineare Modelle als Basis-Lerner genutzt werden.
- GLM mit Regularisierung über Lasso oder Elastic Net **glmnet** (s. Sektion 4.5); R-Paket: *glmnet* (Version 2.0-18) von Friedman et al. (2010) und Simon et al. (2011). Entspricht der Anpassung eines GLM's - hier: Binomialverteilung mit Logit-Link - über die penalisierte Maximum Likelihood. Der Regularisierungspfad wird über LASSO (Default) bzw. ausweitend über das Elastic Net gesteuert.

Damit ergibt sich grundlegend eine Wettbewerbsanalyse zwischen sechs verschiedenen ML-Methodiken, die zunächst mit ihren Default-Einstellungen angepasst werden. Hinzu kommt nun allerdings die bereits angesprochene Tatsache, dass sich hinsichtlich der Zielgröße IHM eine starke Imbalance der Klassen sichtbar macht. Aus diesem Grund wurden alle Lerner zusätzlich mit vorausgehender Korrektur der Imbalance mittels SMOTE bedacht, dessen Erklärung sich in Untersektion 5.1.2 vertieft findet. Des Weiteren wurde durch Sektion 3.3 die potentielle Erfordernis des Tunings der Hyperparameter verdeutlicht. Um auch dem Rechnung zu tragen werden die ausgewählten ML-Methodiken dem Hyperparameter-Tuning unterzogen, wobei hierzu insbesondere das in Untersektion 5.1.3 beschriebene Paket *mlrHyperopt* zum Einsatz kommt, um den Suchraum der Hyperparameter auf sinnvolle Weise festzusetzen.

Hiervon ausgenommen ist der cForest, da bei Probst et al. (2018), sowie Probst et al. (2019) offen gelegt ist, dass sich durch Hyperparameter-Tuning keine nennenswerte Verbesserung in der Performance von RFs einstellt und diese bereits mit Default-Einstellungen äußerste gute Leistungen zu verzeichnen haben, sodass an dieser Stelle innerhalb des Benchmarkings v.a. Rechenzeit eingespart werden kann.
Kombiniert man nun jegliche der genannten Settings der sechs zugrunde gelegten ML-Methoden, so gilt es vier verschiedene Einstellungen pro Lerner - wobei der cForest nur mit zwei unterschiedlichen Settings bedacht ist - zu unterscheiden. Demgemäß wird jeder Lerner auf Basis der Default-Einstellungen trainiert, sowie dies um - das im engeren Sinne dem Tuning unterzogene - SMOTE erweitert wird. Außerdem wird ausgenommen von cForest jeder Lerner mit gleichzeitigem Hyperparameter-Tuning beurteilt und des Weiteren im vierten Setting die Performance der Lerner mit Berücksichtigung von SMOTE und Tuning ermittelt. Letztlich ergeben sich so 22 Lerner, welche innerhalb der Benchmark-Studie gegeneinander antreten. Die zugehörigen Default-Einstellung bzw. der Suchraum des Hyperparameter-Tunings und von SMOTE können Tab. 8.4 des Anhangs entnommen werden. Hierin spiegelt sich der durch *mlrHyperopt* (s. Sektion 5.1.3) bereitgestellte „optimale“ Suchraum für das Hyperparameter-Tuning wider. Auf dieses lässt sich allerdings nur für die Methoden SVM, glmboost, sowie glmnet zurückgreifen, für kNN und cTree dagegen findet sich (noch) keine Implementierung. Deren Suchraum wurde selbständig bzw. im Falle von cTree durch Übertragung der implementierten Einstellung von rpart auf cTree festgesetzt. Diese Einstellungen sind in Rücksprache mit dem Betreuer dieser Arbeit A. Hapfelmeier vorgenommenen und können ebenso Tab. 8.4 entnommen werden.

Lerner	**Default-Einstellung**	**SMOTE**	**Tuning**	**SMOTE und Tuning**
kNN	✔	✔	✔	✔
SVM	✔	✔	✔	✔
cTree	✔	✔	✔	✔
cForest	✔	✔		
glmboost	✔	✔	✔	✔
glmnet	✔	✔	✔	✔

Tab. 6.2: Durchführung der Benchmark-Studie mit 22 verschiedenen Lernern, basierend auf sechs ML-Methoden mit jeweils unterschiedlichen Einstellungen (s. Tab. 8.4)

Unter Zuhilfenahme des im *mlr* Paket bereitgestellten Tuning-Wrappers (s. Sektion 5.1.1) lässt sich schließlich die Benchmark-Analyse durchführen. Die Anpassung des „besten“ Modells erfolgt hierbei basierend auf 5-facher CV, sodass jeweils 4/5 der Daten als Trainingsdaten und der verbleibende Teil als Testdaten zur Bestimmung der Güte des Modells genutzt werden können, wobei für genau Ausführungen hierzu auf Sektion 3.2 verwiesen sei. Insofern dabei

Hyperparameter-Tuning zum Einsatz kommt ist die Erfordernis eines Validierungsdatensatzes gegeben und in diesem Zuge wird schließlich auf genestetes Resampling zurückgegriffen, wobei entsprechend die äußere Schleife eine 5-fache CV darstellt und nun auch zur Ermittlung der „optimalen“ Hyperparameter in der inneren Schleife eine 5-fache Kreuzvalidierung genutzt wird. Zusätzlich sei vermerkt, dass mittels Grid Search (Resolution 5) die „optimalen“ Hyperparameter innerhalb des angegebenen Suchraums hervorgebracht werden sollen. Die genannten Einstellungen der (genesteten) Kreuzvalidierung begründen sich in Absprache mit A. Hapfelmeier und wurden aufgrund der hohen Rechenzeit bewusst nicht höher als auf genestete, 5-fache CV gesetzt.

6.2.2 Auswertung der Ergebnisse aus der Benchmark-Studie

Nachdem die in Kap. 5 beschriebene Shiny App auf Basis der in dieser Arbeit durchgeführten Benchmark-Studie entwickelt wurde, können entsprechend die Ergebnisse mittels der von *shinyBMR* ausgegebenen Grafiken und Tabellen präsentiert werden, indem das BMR-Objekt in die App eingelesen wird. Wie bereits angesprochen wird im Folgenden vermehrt das pAUC zur Beurteilung der Performance herangezogen.
Erste Analysen der Performance sind in Abb. 6.1 bereitgestellt. In dieser sind die erreichten pAUC-Werte der jeweiligen ML-Methode abgebildet, wobei eine zusätzliche Gruppierung nach der zugrunde gelegten Bearbeitung mit kein Tuning vs. Tuning bzw. kein SMOTE vs. SMOTE erfolgt. Obere Grafik umfasst hierbei die aggregierten BMRs, wohingegen in der unteren Grafik noch nach den unaggregierten Ergebnissen aufgeschlüsselt ist, sodass Letztere mittels Boxplots präsentiert werden und die zugehörigen aggregierten Ergebnisse grau hinterlegt sind. Gleichzeitig sind die ML-Methoden in aufsteigender Reihenfolge nach der im Mittel erzielten Performance angeordnet. Die exakten Werte der aggregierten Performance pro Lerner können der sich im Anhang befindenden Abb. 8.5.6 entnommen werden.
Generell lässt sich keine direkte Vorrangstellung in Anbetracht der vorgenommenen Bearbeitung erkennen. So scheinen die auf räumlicher Klassifikation beruhenden Methoden kNN und SVM besonders durch vorhergehendes Tuning einschließlich SMOTE eine Leistungssteigerung zu erfahren, wohingegen derartige Behandlung der Daten für den cTree die mit geringste Performance hervorbringt. Allgemein birgt SMOTE eher eine Verminderung der Prädiktionsleistung von cTree, sowie dem darauf aufbauendem cForest. Allerdings zeigt sich im Vergleich der beiden Methoden eine deutliche bessere Performance für den cForest entgegen dem Entscheidungsbaum, wobei die Varianz des pAUC's innerhalb des cForest deutlich herabgesetzt ist, was auf die in Sektion 4.3 geringe Robustheit von Entscheidungsbäumen bei kleinen Änderungen in den Daten hindeutet. Insbesondere sei in diesem Zuge nochmal erwähnt, dass Ensemblemethoden eben genau an diesem Punkt ansetzen und Prozeduren mit hoher Varianz, aber geringem Bias - wie eine derartige durch den cTree gegeben ist - aufbessern. In Anbetracht niedriger Varianz sollte insbesondere die Leistung des cForest's, sowie auch des

glmnet's - unabhängig von vorausgehenden Bearbeitungen - hervorgehoben werden. Insbesondere scheinen die ML-Methoden glmboost, cForest und glmnet relativ unabhängig von vorhergehender Bearbeitung mit Tuning oder SMOTE zu sein, nachdem die Mediane pro Lerner dicht beieinander liegen.

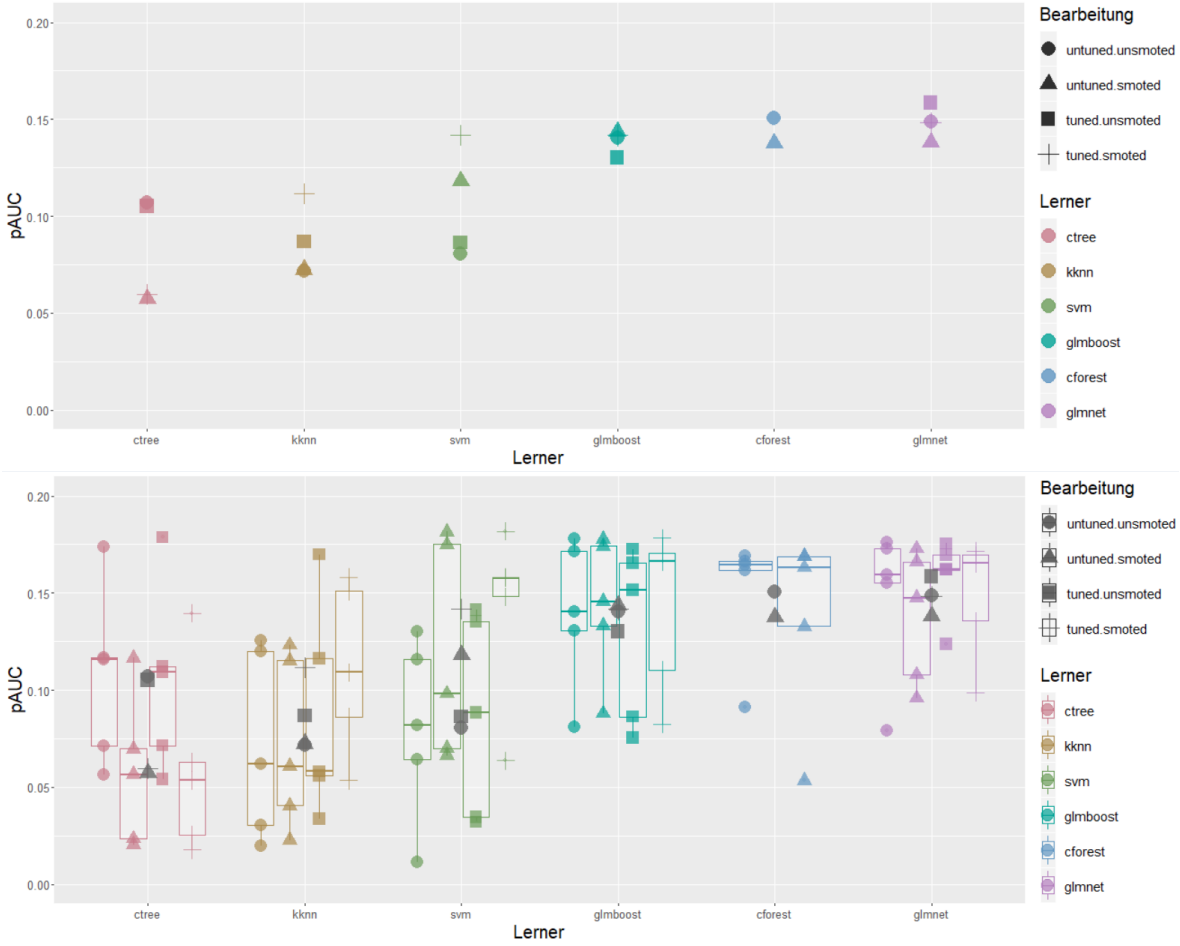


Abb. 6.1: Performance (pAUC) der 22 Lerner in Anwendung auf Daten zu akuter Cholangitis: Gruppierung nach ML-Methodik und Bearbeitung mit Tuning/SMOTE. Oben: aggregierte BMRs; unten: unaggregierte BMRs.

Das pAUC ist derart gewählt, dass sich die Werte auf einem Intervall von $pAUC \in [0; 0.2]$ ansiedeln, wobei 0.2 dem maximalen, angestrebten Wert markiert. In Anbetracht der aggregierten BMRs kann hier das höchste pAUC mit einem Wert von 0.159 für das glmnet mit Hyperparameter-Tuning verzeichnet werden, für welches die Trainingsdaten vorhergehend nicht mit SMOTE bedacht wurden. Der cForest mit Default-Einstellungen lässt sich in diesem Zuge mit einem pAUC von 0.151 auf Rang zwei einordnen. Rang drei und vier sind wiederum dem glmnet zuzusprechen, welches mit Default-Settings ein pAUC von 0.149 erreicht und unter Tuning mitsamt SMOTE auf einen Wert 0.148 kommt. Rang 5, 7 und 8 sind glmboost zuzusprechen und insgesamt scheint sich somit herauszukristallisieren, dass

die Daten gut anhand von Linearkombinationen ausgewählter Einflussgrößen innerhalb des GLM's modellierbar wirken. Auch dann, wenn das AUC zur Beurteilung der Performance herangezogen wird, ergibt sich ein sehr ähnliches Bild. Dieses zeigt sich sowohl im Vergleich von Abb. 6.1 und 8.11, als auch durch die Vergabe von Rängen bei der Beurteilung der mittleren Performance anhand des AUC's. So kann auch hier das glmnet mit Hyperparameter-Tuning, ohne SMOTE als Gewinner-Methode ausgemacht werden, während Rang 2, 3 und 4 zwar eine Neuvergabe der Plätze verglichen mit denen des pAUC's erfahren, jedoch nach wie vor von dem glmnet mit Tuning und SMOTE, sowie dem glmnet und dem cForest mit Default-Setting belegt sind. Gleichzeitig sind Rang 5, 6 und 7 nunmehr von entsprechenden Lernern des glmboost besetzt. Zu derartigen Aussagen kommt man auch unter Einbeziehung der Grafiken aus Abb. 8.12, wobei die s.g. Heatmaps letztlich nur eine weitere Möglichkeit darstellen die Ergebnisse der Benchmark-Studie zugänglich zu machen. Dabei wird die Differenz in der Leistung der einzelnen Lerner nicht wie bei Boxplots über die Spannweite der Gütemaße, sondern mittels farblicher Abstufungen ersichtlich gemacht.

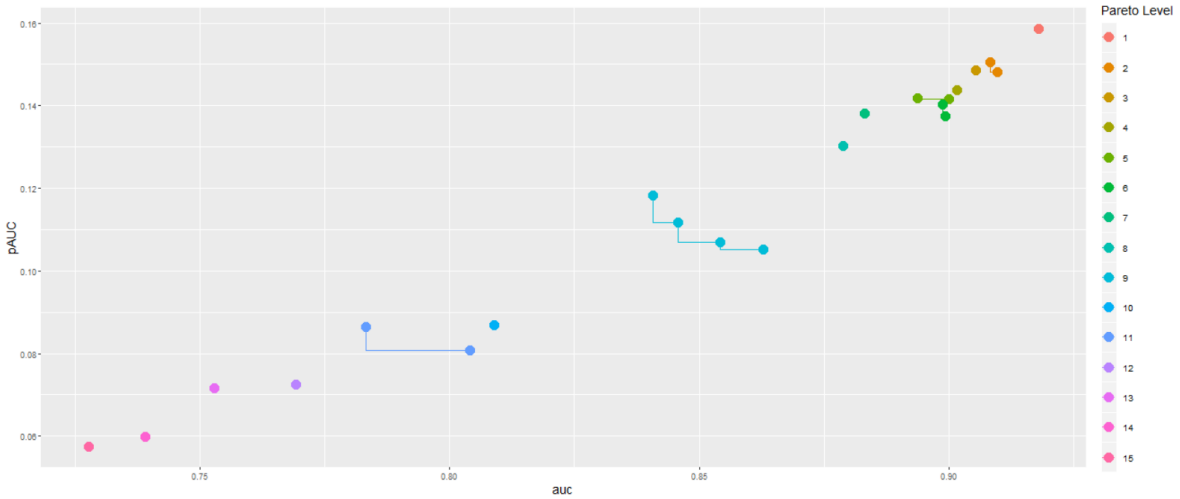


Abb. 6.2: Skyline-Level-Plot mit Berücksichtigung der Gütemaße AUC und pAUC: Pareto-Dominanz des glmnet's mit Tuning gegenüber den übrigen Lernern in Anwendung auf Daten zu akuter Cholangitis.

Es gilt nun der Frage nach zu gehen, ob das glmnet - mit Hyperparameter-Tuning, ohne SMOTE - in Anwendung auf die vorliegenden Daten als „klares" Gewinnermodell deklariert werden kann oder ob die auf den niedrigeren Rängen angesiedelten Lerner ggf. als gleichermaßen gut bzw. - Bezug nehmend auf die Pareto-Front (s. 5.2.2) - als nicht von dem genannten glmnet dominiert angesehen werden könnten. Nutzt man also die Beurteilungsmöglichkeit der Pareto-Dominanz und damit letztlich die in *shinyBMR* inbegriffene Implementierungen, so kann man zu einer eindeutigen Aussage kommen: Unter Berücksichtigung von sowohl AUC, als auch pAUC lässt sich das glmnet mit Tuning als Pareto-dominant gegenüber den übrigen Lernern innerhalb der Benchmark-Studie ausmachen. Mittels der top-k Selektion können

auch für die verbleibenden angewandten Methodiken das Level errechnet werden, zu dem diese schließlich der Pareto-Front angehören würden. Die bildliche Darstellung in Form des Skyline-Level-Plots findet sich in Abb. 6.2 und die zugehörigen Maßzahlen können Abb. 8.13 des Anhangs entnommen werden.
Insbesondere kann anhand von Abb. 6.2 unter gleichzeitiger Berücksichtigung von Abb. 8.13 die allgemein relativ starke Dominanz des glmnet's gegenüber den übrigen Lerner unter Betrachtung der beiden Gütemaße $pAUC = 0.159$ und $AUC = 0.918$ ausgemacht werden. Gleichzeitig wird bei Betrachtung der Skyline-Levels aber auch deutlich, dass der cForest ($pAUC = 0.151$, $AUC = 0.908$) ohne Tuning im Pareto-Set des zweiten Levels - gemeinsam mit dem glmnet bei Bearbeitung mit Tuning und SMOTE ($pAUC = 0.148$, $AUC = 0.910$) - inbegriffen ist und damit offensichtlich auch starke Performance bei Anwendung auf die Daten zu akuter Cholangitis zu verzeichnen hat. Da insbesondere der cForest eine Blackbox-Methode darstellt, begründen sich nunmehr die beispielhaften Veranschaulichungen für die Zugänglichkeit zur Interpretation mittels *iml* in Sektion 5.3.

6.2.3 Anpassung und Anwendung von glmnet zur Beurteilung der Effekte

Nachdem das glmnet als Gewinnermodell ausgemacht werden konnte, gilt es nun dieses konkret an die Daten anzupassen und nicht nur in Form von „Probeläufen“ innerhalb des Benchmarkings zu belassen. Insbesondere ist damit eine erneute Anpassung des Modells an alle vorliegenden Daten gekoppelt einschließlich der Bestimmung optimaler Hyperparameter. Für die genaueren mathematischen Hintergründe des GLM's mit Penalisierung über das Elastic Net sei Sektion 4.5 herangezogen.
Um das Hyperparameter-Tunings von Shrinkage und Mixing Parameter - λ und α - zu bewerkstelligen wird zunächst k-fache CV angewandt, was im *glmnet* Paket bereitgestellt ist. Als Resampling-Strategie wird sich wiederum auf eine 5-fache CV berufen, wobei auch die Suchintervalle wieder mittels Grid Search auf eine Fünfer-Sequenz abgesteckt werden. Als Gütemaß wird das AUC herangezogen und so lassen sich schließlich ein α von 0.25, sowie ein λ von 0.0093 als „optimale“ Hyperparameter ausmachen, nachdem sich für diese Einstellungen der maximale gemittelte Wert des kreuzvalidierten AUC's ergibt. Hervorhebend sei an dieser Stelle noch einmal erwähnt, dass der Begriff „optimal“ hier insbesondere auch deswegen als relativierte Beschreibung zu erachten ist, da das Hyperparameter-Tuning natürlich gewisser Zufälligkeit v.a. in Anbetracht der zufälligen Einteilung der Subdatensätze innerhalb der CV unterliegt und hinzukommen das Suchraster recht klein gewählt ist. Ausweitend sei erwähnt, dass die Kalkulationen anhand des *glmnet* Paketes sich auf eine Datenmatrix stützt, sodass alle hier inbegriffenen kategorialen Variablen gemäß der 0-1-Kodierung (Dummy-Kodierung) aufgenommen sind.
Mit den ermittelten Werten der Hyperparameter $\alpha = 0.25$ und $\lambda = 0.0093$ wird final das fokussierte glmnet angepasst, um Effektschätzer zu den einzelnen Einflussvariablen zu bestimmen.

Durch die Nutzung des Elastic Net's können also die Koeffizienten derjenigen Variablen, die als irrelevant zu erachten sind, auf Null geschrumpft werden und es wird gleichzeitig der Problematik von LASSO entgegen gesteuert, dass bei korrelierten Variablen nur eine Variable aus dieser Gruppe ausgewählt wird (s. 4.5.2). Hieraus resultieren dann letztlich die in Tab. 6.3 gelisteten Schätzer. Nachdem die Anpassung eines Logit-Modells zugrunde liegt, gilt es die β-Koeffizienten als logarithmiertes Chancenverhältnis/Risiko (Log Odds Ratio) aufzufassen. Zugleich sind Effekt binärer Größen im Bezug auf die jeweilige Referenzkategorie zu erachten, wobei hier stets „ja“ und „größer gleich“ als Referenzen gewählt ist. Mit einem Punkt versehen Schätzungen sind durch die Penalisierung gegen Null geschrumpft worden und damit als einflusslos zu erachten.

Prädiktor (Effekt ggü. Referenzkategorie)	β**-Koeffizient**	**Odds Ratio**
[Intercept]	[9.447]	$[1.2668 \cdot 10^4]$
Geschlecht (männlich)	-0.4234	0.6548
Mentale Verwirrtheit (nein)	-2.2051	0.1102
Vorhergehende Cholangitis Episode (nein)	-0.1141	0.8922
Vorhergehende Intervention (nein)	0.2558	1.2915
Katecholamine erfordernde Hypotension (nein)	-0.1739	0.8404
Organversagen (nein)	-1.2951	0.2739
Ursache der Cholangitis (nicht-maligne)	-0.6337	0.5306
Alter	0.0043	1.0043
Alter ($<$ 65 Jahre)	.	1
Alter ($<$ 75 Jahre)	.	1
Temperatur/Fieber	-0.2681	0.7648
Temperatur ($<$ 39 °C)	0.0498	1.0511
Anzahl der Leukozyten	0.0084	1.0084
Anzahl der Leukozyten ($<$ 15 G/l)	-0.2642	0.7678
Bilirubin-Gehalt	0.0121	1.0122
Bilirubin-Gehalt ($<$ 5 mg/dl)	0.5007	1.6498
Quick's Wert	-0.0076	0.9924
Quick's Wert ($<$ 50%)	0.3376	1.4016
Serumkreatinin	.	1
Serumkreatinin ($<$ 2 mg/dl)	-0.1996	0.8190
Anzahl an Blutplättchen	.	1
Anzahl an Blutplättchen ($<$ 100 G/l)	.	1

Tab. 6.3: Kovariableneffekte resultierend aus der Anpassung eines glmnet's mit $\alpha = 0.25$ und $\lambda = 0.0093$ an die Daten zu akuter Cholangitis zur Modellierung der IHM

Die errechneten Effektstärken sind jedoch kritisch zu beäugen, nachdem insbesondere keine Standardfehler und damit Konfidenzintervalle für diese berechnet werden können bzw. sollten. Y. Yue (2013) folgend ist es nicht sinnvoll Standardfehler wie im normalerweise üblichen Regressionskontext zu berechnen, da die Schätzer an sich durch die Penalisierung stark verzerrt sind. Noch einmal Bezug nehmend auf 4.5.2 begründet sich dies darin, dass die Prozedur der penalisierten Schätzung die Varianz der Schätzer reduziert, womit jedoch eine Erhöhung des Bias (Bias-Varianz-Tradeoff) einhergeht. Demgemäß bildet der Bias jedes Schätzer einer der Hauptkomponenten von dessen MSE, wohingegen die Varianz nur einen kleinen Teil beiträgt. Allerdings lässt sich der Wert dieses Bias nicht mit genügender Genauigkeit abschätzen, nachdem insbesondere Bootstrap-Prozeduren nur den Zugang zur Varianz eines Schätzer gewähren. Vertrauenswürdige Schätzungen des Bias wären einzig dann ermittelbar, wenn die wahren unverzerrten Effektschätzer zugänglich wären, was zumeist nicht der Fall ist, wenn ohnehin schon auf penalisierte Schätzmethoden zurückgegriffen wird.
Wenngleich sich zwar keine Aussagen bzgl. der Signifikanz gegebener Effekte aus Tab. 6.3 treffen lassen, können dennoch Risiko-Aussagen gemacht werden. Demgemäß zeigt sich u.a., dass für Männer das Risiko für IHM gegenüber Frauen sinkt, sodass sich dieses bei ca. 2/3 zugunsten der Männer einpendelt. Damit lässt sich ggf. die Frage nach einem möglichen Zusammenhangs mit der erhöhten Inzidenz von Gallenstein - was mitunter Auslöser von Cholangitis sein kann - im Hinblick auf das weibliche Geschlecht (vgl. Cheung, o. J.) aufwerfen. In diesem Zusammenhang bringt nämlich auch J. Vieira Barbosa (2018) den Nachweis, demzufolge Frauen häufiger von chronischer Cholangitis betroffen sind als Männer. Das Alter der Patienten hingegen scheint wenn überhaupt einen recht geringen Einfluss auf die IHM zu haben, sodass das Risiko mit der Erhöhung des Alter um ein Jahr multiplikativ um 1.0043 ansteigt, wobei die zwei dichotomisierten Altersvariablen durch die Penalisierung sogar als gänzlich einflusslos eingestuft werden.
Recht intuitiv mögen die folgenden Effekt erscheinen: Ein vermindertes Risiko für das Auftreten von IHM lässt sich in der Gruppe der Patienten keiner vorhergehenden Cholangitis (Faktor 0.8922), keine Katecholamine erfordernder Hypotension (Faktor 0.8404), sowie derjenigen mit nicht-krebsbedingter Ursache der Cholangitis (Faktor 0.5306) gegenüber der entsprechenden Referenzkategorie erkennen. Letzteres wirft allerdings die Frage nach einer potentiellen Effektverstärkung bezüglich der IHM auf, wenn man bedenkt, dass die maligne Form der Cholangitis ggf. Streuung der Krebszellen mit sich führen kann bzw. umgekehrt hierdurch bedingt ist. In Einklang mit Yokoe et al. (2013) lassen sich auch die Effekte der Risiko-Verminderung um knapp 89 % bei nicht vorliegender mentaler Verwirrtheit und um ca. 73 % bei nicht Auftreten von Organversagen gengüber der entsprechenden Referenz bringen. Für die Gruppe der Patienten, die keine vorhergehende Intervention erfahren, zeigt sich ein um knapp 30 % erhöhtes Risiko für das Eintrete des Todesereignisses verglichen mit denen, die eine vorherige Intervention vorweisen können, sodass sich wohl generell die Frage nach der Notwendigkeit

frühzeitiger Behandlungsmaßnahmen ergibt.
Fokussiert man nunmehr die Blutwerte lassen sich die folgenden Feststellungen machen: Eine steigende Anzahl an Leukozyten und damit von Infektparametern bedingt mit Anstieg der Anzahl an Leukozyten um eine Einheit die Erhöhung der Wahrscheinlichkeit für das Eintreten von IHM um 0.84 %. Gleichzeitig lässt die zugehörige, dichotomisierte Einflussgröße erkennen, dass Patienten mit einer Anzahl von unter 15 G/l Leukozyten ein vermindertes Risiko gegenüber der Referenzgruppe aufweisen. Die kontinuierliche Bilirubin-Variable vermag mit Anstieg dessen Wertes um 1 mg/dl das Risiko von IHM um den Faktor 1.0122 zu erhöhen. Hingegen zeigt das binäre Dependant ein stark erhöhtes Risiko bei der Patientengruppe unter 5 mg/dl Bilirubin - um ca. 65 % höher - gegenüber denen mit einem Bilirubin-Wert größer oder gleich 5 mg/dl. Der Quick's Wert deutet einen hierzu gegenläufigen Effekt an: mit dem prozentualen Anstieg dieses Wertes um eine Einheit vermindert sich das Risiko multiplikativ um 0.9924, während sich bei Gruppierung nach Patienten mit einem Quick's Wert unter 50 % zu der entsprechende Referenzkategorie ein umgekehrter Effekt erkennen lässt, sodass für erstere Gruppe eine um 40 % erhöhte Wahrscheinlichkeit für IHM zutage kommt. Liegt der Wert des Serumkreatinin's unter 2 mg/dl, so sind dem zugehörige Patienten einem geringeren Risiko - mit multiplikativer Änderung um 0.8190 - ausgesetzt verglichen mit der Referenzgruppe. Zugleich ist der Effekt der entsprechende kontinuierlichen Variable im Zuge der Penalisierung unter Betrachtung der logarithmierten OR auf 0 geschrumpft worden. Gleiches gilt für die beiden Einflussgrößen, die Informationen bezüglich der Blutplättchen beinhalten.
Zwar können - wie bereits zuvor erwähnt - die Effekt nicht im Hinblick auf ihre Signifikanz beurteilt werden, jedoch zeigte sich rückblickend auf die beispielhaften Analysen mittels *iml* (s. Sektion 5.3) auf Basis des cForest's eine teils deutliche Relevanz der Prädiktoren mentale Verwirrtheit, sowie auch Organversagen. Diese Feststellung verträgt sich insbesondere auch mit der hohen Effektstärke dieser beiden Einflussgrößen, die durch das glmnet zutage kommt. Abschließend lässt sich hier also der Anreiz vertiefender Analysen schaffen, sodass in diesem Zuge wohl auch andere ML-Methodiken angepasst werden sollten, um vergleichende Maßnahmen im Bezug auf die potentiellen Einflussgrößen auf die IHM anstellen zu können. Derartiges Vorgehen - also die unmittelbare Anpassung mehrerer, verschiedener Lerner - könnte insbesondere in der Anwendung von *iml* interessant sein, sodass die internen Abläufe der ML-Methoden in gewissem Maße zugänglicher und vergleichbarer würden und so ggf. auch die Rangvergabe innerhalb der Benchmark-Studie greifbar würde.

7 Zusammenfassung und Ausblick

Um diese Arbeit abzurunden, soll nachfolgend eine kurze Zusammenfassung der gewonnen Erkenntnisse und Ergebnisse dargeboten werden, ebenso wie sich darauf aufbauend ein Ausblick formulieren lässt. In diesem Sinne lässt sich die hier im Fokus stehende Benchmark-Studie als ein Werkzeug des ML's von nicht zu verachtender Bedeutsamkeit ausmachen. Generell befähigt diese zu unmittelbaren und aussagekräftigen Wettbewerbsanalysen verschiedener ML-Methoden, sodass vorzunehmende Analysen unter der Rechtfertigung der Vorrangstellung einer oder mehrerer Methodiken getätigt werden können. Damit drängt sich wohl die Empfehlung auf das Benchmarking - ggf. in klein angelegter Form - als grundlegende Vorarbeit bei der Durchführung von Analysen im Kontext des maschinellen Lernens einzuführen.
Im Zuge dessen sind Kenntnisse - im Falle nicht ausreichend großer Datenmengen - der vorzunehmenden Aufteilung in Trainings-, (Validierungs-) und Testdatensatz anhand unterschiedlicher Resampling-Strategien vorauszusetzen, sodass insbesondere das genestete Resampling Anwendung erfahren kann und damit auch gleichzeitiges Hyperparameter-Tuning in die Benchmark-Studie einfließt. Um schließlich aussagekräftige Ergebnisse aus dieser Analyse ziehen zu können, bedarf es zusätzlich der Auswahl eines oder mehrerer geeigneter Gütemaße, die sich u.a. an der Art des Problems - Klassifikations- oder Regressionsaufgabe - orientieren. Die sich aus der Benchmark-Studie ergebenden Performance-Wert zu den einzelnen Lernern sollten abgesehen von der Leistungsstärke auch unter Berücksichtigung von Faktoren wie etwa Laufzeit des Algorithmus, sowie Komplexität und Interpretationsmöglichkeiten getroffen werden. Im Falle der Beurteilung anhand mehrerer Gütemaße sollten Hilfsmittel wie etwa die s.g. Pareto-Front in Betracht gezogen werden, sodass die Vorrangstellung der Lerner innerhalb von mindestens zwei Dimension ausgemacht werden kann. Des Weiteren sollte auch auf visuelle Bewertungen der Benchmark-Ergebnisse nicht verzichtet werden.
Neben alledem ist natürlich die Auswahl der ML-Methoden mitunter ein bedeutender Bestandteil der Benchmark-Studien. Um nun von der bisher allgemein gehaltenen Beschreibung abzuwenden, seien an dieser Stelle kurz ausgewählte Methodiken für die konkrete Anwendung auf das vorliegendes Klassifikationsproblem dieser Arbeit namentlich erwähnt. Hierzu gilt es nun explizit die Methode der k-Nearest Neighbors (kNN), sowie Support Vector Machines (SVM) zu nennen. Des Weiteren bietet sich die Verwendung von Entscheidungsbäumen mit besonderem Fokus auf Bäume (cTrees), die auf konditionaler Inferenz fußen, und die Ausweitung dessen auf Random Forests bzw. spezifisch cForests an. Letzteres lässt sich als Ensemblemethode deklarieren, wobei hierunter auch das Boosting oder vertiefend das (Componentwise) Gradient Boosting, welchem u.a. glmboost angehört, fällt. Da die bisherige Auswahl sich auf non-parametrische Verfahren stützt, kann dem Rechnung getragen werden, indem eine parametrische ML-Methode mit der Einbeziehung des Logit-Modell's mitsamt Penalisierung bzw. genauer des glmnet's in die Wettbewerbsanalyse aufgenommen wird.

Zur konkreten Durchführung derartiger Benchmark-Studien bietet sich innerhalb der Software R die Nutzung des *mlr* Paketes an, welches insbesondere den Vorteil birgt, dass eine standardisierte Schnittstelle für inbegriffene ML-Methoden dargeboten wird. Anhand dessen können die implementierten Lerner durch s.g. Wrapper um zahlreiche Funktionalitäten erweitert werden, womit sich u.a. Imbalance-Korrektur und hierzu zählender Methode SMOTE vornehmen lässt. Außerdem sind einige Zusatz-Pakete bereitgestellt, sodass die Funktionalitäten von *mlr* noch weiter ausgedehnt werden können. Beispielsweise kann dabei mittels *mlrHyperopt* eine Parameterraum-Konfiguration erfolgen, womit letztlich ein vollautomatisches Hyperparameter-Tuning ermöglicht wird.
Wie dies bereits Erwähnung fand, sollten die Ergebnisse des Benchmarkings auch visuelle Darstellung erfahren. Nachdem derartige Mittel innerhalb von *mlr* jedoch begrenzt sind, findet die Ausarbeitung der App *shinyBMR* ihre Begründung. Hierdurch sollen neben grafischen Anschauungen und Vergleichsmöglichkeiten auch ein Überblick zu dem vermeintlich komplexeren Benchmark-Objekt, welches aus der Durchführung einer Benchmark-Studie mittels *mlr* resultiert, offenbart werden. Zugleich wird dem Benutzer auf interaktive Weise ermöglicht die Entscheidung für ein oder mehrere Gewinner-Modelle auf unterschiedliche Art und Weise zu beurteilen. Um nunmehr auch der Tatsache Rechnung zu tragen, dass die Anpassung eines derartigen finalen Modells im Kontext des ML's zumeist mit der Unzugänglichkeit zu Interpretationsmöglichkeiten einhergeht, werden in *shinyBMR* auf Grundlage des Paketes *iml* Lösungsansätze hierzu bereitgestellt. Dabei wird zum Großteil auf s.g. Modell-agnostische Methoden zurückgegriffen, sodass grob formuliert über das Abändern des Inputs und anschließendes Messen der sich ergebenden Änderungen im Prädiktions-Output dieses Ziel für jeglichen Lerner des SL's verfolgt werden kann. Derartige iml-Methoden sind als entkoppelt von den Analysen des Benchmarkings innerhalb von *shinyBMR* anzusehen und verlangen daher den spezifischen Upload des fokussierten Modells mitsamt dem zugehöriger Daten.
Die kritische Auseinandersetzung mit der App *shinyBMR* regt - abgesehen von potentiell wahrzunehmenden, kleinen Änderungsmöglichkeiten - v.a. zu ausweitendem Ausbau und zusätzlicher Verknüpfung mit den innerhalb des Benchmarkings assoziierten Funktionalitäten an. So wäre es beispielsweise denkbar die gesamten, der Benchmark-Analyse vorhergehenden Prozeduren in den Rahmen der App einzubetten, sodass dem Anwender eine einfache, interaktive Nutzung des ML-Werkzeuges bereitgestellt wird, wobei insbesondere keine tiefergehenden Kenntnisse im Umgang mit der Programmierung derartiger Studien bzw. *mlr* von Nöten sind. Solche Vorstellungen wären ggf. über die Verbindung von *shinyBMR* mit der die Funktionalitäten von *mlr* umfassenden App *shinyMlr* realisierbar. In Anbetracht von hier hauptsächlich fokussierten klinischen Studien wären wohl auch über den Regressions- und Klassifikationskontext hinausgehende Benchmark-Analysen im Bezug auf Survival- oder auch Clusterdaten wünschenswert, sodass dies innerhalb von *shinyBMR* Implementierung finden könnte. Insbesondere sollten jedoch Ausweitung in Hinblick auf weitere Wrapper - abgese-

hen von Tuning und SMOTE - bedacht werden. Kritisch zu beäugen bleibt die unweigerlich mit Benchmark-Studien einhergehende erforderliche Rechenleistung, sodass sich auch Fragen vereinfachter Implementierung oder Möglichkeiten der Zeiteinsparung aufwerfen.
Unter Anwendung der bis hierher erlangten Erkenntnisse und Nutzungsmöglichkeiten der ausgearbeiteten App *shinyBMR* kann schließlich eine Benchmark-Studie zu Daten akuter Cholangitis durchgeführt werden. Anhand der konkreten Wettbewerbsanalyse zwischen den zuvor genannten, ausgewählten Klassifikationsmethodiken, welche jeweils mit Default-Einstellungen, Hyperparameter-Tunging und/oder SMOTE bedacht sind, lässt sich - insbesondere mit Blick auf die Pareto-Dominanz beurteilt über das AUC und pAUC - die Vorrangstellung des glmnet's mit Tuning, ohne SMOTE ausmachen. Bezug nehmend auf die Anpassung dieses Modells an die Daten akuter Cholangitis können die im engeren Sinne „optimalen" Hyperparameter $\alpha = 0.25$ und $\lambda = 0.0093$ durch CV hervorgebracht werden, sodass schließlich das zugehörige glmnet zur Bestimmung von Effektstärken verwendet werden kann. Wenngleich aufgrund der Penalisierung keine unmittelbaren Aussagen über die Signifikanz des jeweiligen Einflusses gemacht werden sollten, können die Effekte - mit gewisser Vorsicht - zur Beurteilung des Risikos in Hinsicht auf die In-Hospitale Mortalität genutzt werden. Im Zuge der Penalisierung gilt es insbesondere die Variablen zur Anzahl der Blutplättchen in kontinuierlicher und dichotomisierter Form, das reelwertige Maß des Serumkreatinin's, sowie die zwei binären Altersvariablen (mit unterschiedlichem Schwellenwert) als einflusslos aufzufassen. Mit Blick auf die binären Einflussgrößen lässt sich v.a. den Variablen Organversagen und mentale Verwirrtheit ein hoher Effekt nachsagen, jeweils mit deutlich vermindertem Risiko bei Ausprägung „nein" gegenüber der Referenzklasse. Mit Blick auf die beispielhaften Analysen über iml basierend auf einem cForest scheint diese Feststellung untermauert zu werden.
Resümierend lässt sich insbesondere durch die hier getätigten Ausarbeitungen zu Daten akuter Cholangitis festhalten, dass Benchmark-Studien i.A. wohl mehr Anwendung finden sollten, sobald auf groß angelegte Analysen im Rahmen des ML's gesetzt wird. Als limitierender Faktor ist hierbei v.a. die hohe Rechen- und damit Zeitaufwand zu bedenken, sodass in Abhängigkeit von der Größe der Studie, ebenso wie zur Verfügung stehender (technischer) Ressourcen einheitliche Rahmenbedingungen geschaffen werden sollten. Gleichzeitig sollte dennoch ML breitflächigere bzw. ausschöpfendere Nutzung v.a. innerhalb klinischer Studien erfahren. Sobald man sich in der Domäne klinischer Analysen findet, ist man an strikte, teilweise wohl aber auch veraltete Vorlagen und Methodiken gebunden. Es wäre dahingehend wünschenswert die Einbeziehung von ML in den Alltag klinischer Statistik voranzutreiben, sodass sich auch in diesem Bereich das volle Potential des Wissenschaftsgebiets um ML ausschöpfen lässt. Dabei ließe sich wohl v.a. durch die Anwendung von Benchmark-Studien ein hohes Maß an Gewährleistungen in vielerlei Hinsichten sicherstellen. Es gilt in diesem Sinne über vereinfachte Zugänglichkeiten zu derartigen Analyse-Methoden nachzudenken und so nicht zuletzt Werkzeuge wie *mlr* oder gar *shinyBMR* zu kommerzialisieren, verbessern und auszuweiten.

8 Anhang

8.1 Ergänzungen zu Kap. 2 - Maschinelles Lernen

8.1.1 Abgrenzung des ML's von den Termini AI, DL und Statistik

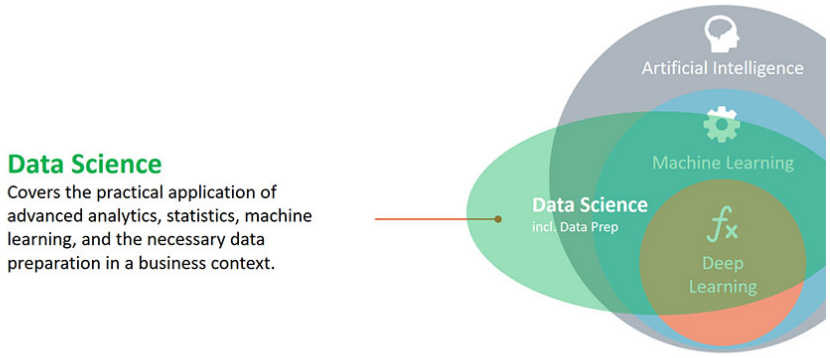
Künstliche Intelligenz (AI)

Mit der Begrifflichkeit künstlicher Intelligenz beginnend, sei im Wesentlichen auf Moeser (2017) verwiesen. Hierbei besteht das Ziel der AI - grob formuliert - in der Hervorbringung intelligenter Systeme/Maschinen, die menschenähnlich arbeiten, handeln und auf bestimmte Umstände bzw. Situationen reagieren können. Prinzipiell wird zumeist eine Unterteilung in schwache (weak AI) und starke (strong AI) künstliche Intelligenz - auch Superintelligenz genannt - vorgenommen. Ersteres umfasst Systeme, die auf die Lösung spezifischer Anwendungsgebiete fokussiert sind. Die Lösung erfolgt dabei mithilfe informatischer und mathematischer Ansätze, die auf Basis der gegebenen Anforderung agieren sollen und optimiert werden. Im Allgemeinen handelt es sich also um eine reaktivierbare Handlung, die ein gewisses Intelligenzlevel nicht übersteigt. In diesem Sinne erlangt das System also kein tiefgreifenderes Verständnis für das Problem, sondern reagiert mithilfe ihm bereitgestellter Methoden. Beispielhaft lassen sich für die schwache AI Mechanismen wie die Bild- und Spracherkennung, Navigationssysteme und Autokorrektur von Textdokumenten nennen.
Dagegen ist starke AI gekennzeichnet durch ebenbürtige oder höher gestellte intellektuelle Fähigkeiten wie die eines Menschen. Insbesondere ist die Handlungsweise bzw. Problemlösung des Systems nicht reaktiv, sondern geschieht aus eigenem Antrieb heraus und ist anpassungsfähig. Kennzeichnend sind also folgende Eigenschaften: logisches Denken, Entscheidungsfähigkeit auch bei unklaren Problemmustern, Fähigkeit zum Planen und Lernen, Kommunikation mittels einer natürlichen Sprache, sowie die Kombination aller Fähigkeiten zum Erlangen eines übergeordneten Ziels. Wenngleich man sich hier hinsichtlich der Definition einer starken KI relativ einig ist, so gibt es dennoch derzeit kein derartiges System. Teilweise ist es sogar umstritten, dass eine künstliche Intelligenz dieser Form überhaupt geschaffen werden kann.

Data Science (DS)

Entsprechend vorangehender Erklärungen innerhalb von Sektion 2.1 können ML und Statistik wohl zum großen Teil der Data Science untergeordnet werden, die sich - wie die Bezeichnung bereits vermuten lässt - mit der Extraktion von Wissen aus Daten beschäftigt. So sollte insbesondere hervorgehoben werden, dass das „Lernen“ im Namen von ML sich auf das Erwerben von Wissen aus einem gegebenen (Trainings-) Datensatz bezieht. Granville (2017a) unterstreicht hierbei, dass Data Science weit über das Feld von ML hinaus geht. So deckt Ersteres das gesamte Spektrum der Datenverarbeitung (Data Processing) ab und damit nicht

nur algorithmische oder statistische Aspekte. Damit ist Data Science ein großes, komplexes Feld, welches die Wissenschaft widerspiegelt, Daten zu gewinnen, extrahieren, kompilieren, prozessieren, analysieren, interpretieren, modellieren, visualisieren, berichten und präsentieren [vgl. Castrounis (2016)]. Demgemäß finden sich hier der Schnittpunkt zahlreicher Bereiche und Techniken, wie der Mathematik, Computerwissenschaften und Programmierung, Statistik und Datenmodellierung, künstlicher Intelligenz, Datenpräparation und vielem mehr. Dieser Aspekt soll mithilfe der Abb. 2.1 ergänzenden Grafik in Abb. **??** verdeutlicht werden, wobei hier der Begriff „business“ wohl besser als „science“ verstanden werde und damit Data Science im Kontext wissenschaftlichen Arbeitens erachtet werden sollte.



Quelle: Mierswa (2017b)

Abb. 8.1: Hierarchischer Struktur von AI, ML, DL mit zusätzlicher Einbeziehung von DS

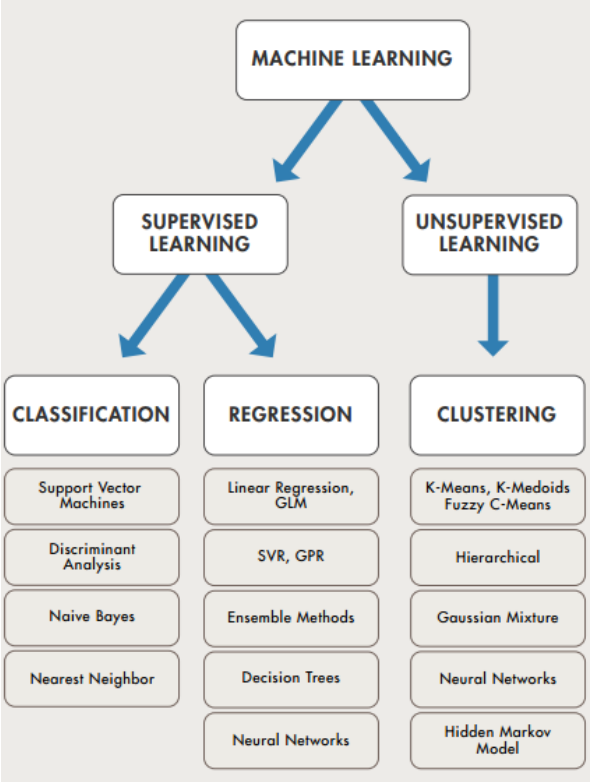
Deep Learning (DL)

Zum Abschluss dieser Sektion sollte noch der Begriff des Deep Learnings (DL) vom maschinellen Lernen abgegrenzt werden. Wie Abb. 2.1 oder Abb. **??** zu entnehmen ist, handelt es sich bei DL um einen Teilbereich des ML. Genauer gesagt spiegelt das DL dabei gemäß Mierswa (2017a) eine spezifische Klasse von Algorithmen des ML's wider, welchen komplexe neuronale Netze zugrunde liegen. Jüngste Methoden im parallelen Programmieren, machten diese Algorithmen überhaupt erst möglich. Genauer soll in diesem Kontext hier nun aber nicht auf die Konzepte des DL's eingegangen werden, sondern stattdessen der Fokus wieder auf ML gesetzt werden und mittels der nachfolgenden Sektion die Arbeit in diesem Sinne vertieft werden.

8.1.2 Supervised (SL) gegenüber unsupervised Learning (USL)

Das nachfolgende Untersektion soll dazu dienen, die Unterschiede zwischen den zwei Hauptästen des ML's zu verdeutlichen. Hierbei gilt es supervised (SL) von unsupervised Learnings (USL) abzugrenzen. Die Verdeutlichungen dazu orientieren sich im Wesentlichen an James

et al. (2013, Kapitel 2.1.4), de Mello und Ponti (2018, Kapitel 1.2 und 1.3) und Soni (2018). Für einen ersten groben Überblick über die Unterteilung des ML's, sowie die den einzelnen Lernarten zugehörigen Methoden kann Abb. 8.2 herangezogen werden. Wie bereits in der vorangegangen Sektion 2.2 umrissen, findet sich hier die Aufteilung der Lernalgorithmen in Klassifikations-, Regressions- und Clustering-Aufgaben. Im Detail wird nicht auf alle der in Abb. 8.2 aufgeführten Methoden eingegangen. Stattdessen findet sich eine genaue Beschreibung der in dieser Arbeit verwendeten Methoden in Kap. 4. Ebenso sei daran erinnert, dass es sich beim ML um ein äußerst komplexes und weitläufiges Gebiet handelt, womit sich in Abb. 8.2 einige weitere - teilweise abgewandelte - Methodiken ergänzen lassen.



Quelle: Granville (2017b)

Abb. 8.2: Zusammenfassung (herkömmlicher) ML Methoden

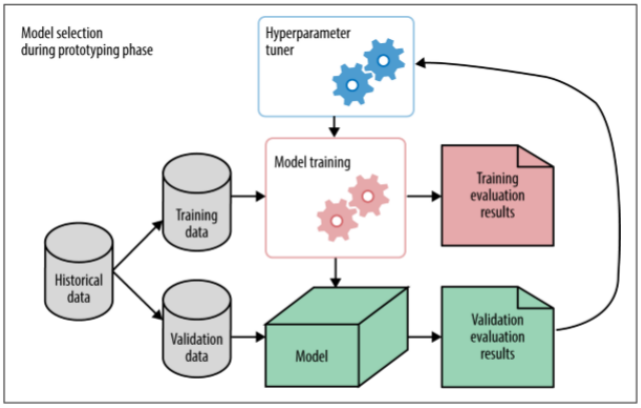
Bisherige Erklärungen basieren im Wesentlichen auf SL: zu jeder Beobachtung $\boldsymbol{x}_i$, mit $i = 1, \ldots, N$ gibt es eine assoziierte Prädiktionsmessung y_i. Hierbei steht die Anpassung eines Modells im Mittelpunkt, welches möglichst gut die Verbindung der Präditkionsvariablen zum Response beschreibt. Ziel des Ganzen ist es zumeist, dieses Modell für relativ exakte Prädiktionen der Zielvariable zukünftiger Beobachtungen [Prädiktion] zu nutzen oder zum besseren Verständnis des Zusammenhangs von Einflussvariabl(en) und dem Response [Inferenz]. Zusammenfassend werden innerhalb des SL's also dem ML-Algorithmus bereits gekennzeichnete (prelabeled) Input-Beispiele übergeben, wobei beabsichtigt wird zu dem bestmöglichen Klas-

sifizierer $\hat{f} : \boldsymbol{x} \to y$ zu konvergieren, sodass die Labels neuer Beispiele mit hoher Genauigkeit prognostiziert werden können. In diesen Kontext können nun vielerlei klassische statistische Lernmethoden dem SL untergeordnet werden, die den Feldern der Klassifikation und Regression angehören. Dazu zählen beispielsweise die lineare und logistische Regression, gemischte additive Modelle (GAMs), neuronale Netzwerke, Entscheidungsbäume, sowie Random Forests (RFs) oder auch Boosting und Support Vector Machines (SVMs). Auf den Großteil der genannten Methodiken wird in den Sektionen von Kap. 4 genauer eingegangen.
Vielmehr gilt es hier nun aber den Unterschied des SL's zum USL zu verdeutlichen: gemäß James et al. (2013, Kapitel 2.1.4) beinhaltet letzteres eine größere Herausforderung, nachdem in dieser Situation für jede der $i = 1, \ldots, N$ Beobachtungen ein Vektor von Messungen $\boldsymbol{x}_i$ vorhanden ist, jedoch kein mit diesen assoziierter Response/Label y_i. Es kann folglich kein Regressionsmodell genutzt werden, da keine Zielvariable vorhanden ist, die prädiktiert werden könnte. Entsprechend ist man zu einem „blinden" Vorgehen gezwungen, wobei also von unsupervised Learning gesprochen werden kann, nachdem der Response/Label fehlt, mittels dessen die Analysen überwacht werden könnten. Im Unterschied zum SL steht nun also nicht mehr das Problem einer bestmöglichen Zuordnung $\hat{f} : \boldsymbol{x} \to y$ im Mittelpunkt, sondern die Analyse der Anordnung gegebener Punkte im Input-Raum und damit das Erkennen von Mustern ohne die explizite Bereitstellung von Labeln.
Wie kann nun aber mit einer derartigen Problemstellung bei USL innerhalb statistischer Möglichkeiten umgegangen werden? Letztlich geht es zumeist darum die Beziehung zwischen den beobachten Variablen zu identifizieren bzw. bestimmte Muster zu erfassen. Demgemäß bietet sich beispielsweise also eine Cluster-Analyse an, womit man auf die bereits in Sektion 2.2 kurz beschriebene Methodik des Clusterings zurückgreift. Jedoch beschränkt sich das USL nicht einzig auf Clustering-Methoden und so sollte hier insbesondere auch die Dichteschätzung aufgeführt werden.
Als die im Kontext des USL's wohl am häufigsten genutzten Methoden sind hier das k-Means Clustering, sowie die Hauptkomponentenanalyse (Principal Component Analysis = PCA) zu nennen. Zusammenfassend finden Algorithmen des maschinellen Lernens dieser Art ihre Anwendung folglich im Bereich explorativer Analysen, sowie der Dimensionenreduktion. Hier sei darauf hingewiesen, dass es im Feld des USL's i.A. nicht möglich ist, die Güte (Performance) unterschiedlicher Modelle zu vergleichen, nachdem keine Labels gegeben sind.
Abschließend gilt es jedoch noch zu berücksichtigen, dass eine dargelegte Unterteilung in überwachte und nicht-überwachte Lernmethodiken nicht immer derartig klar auszumachen ist. Es gibt durchaus Situationen, in denen eine Mischform von SL und USL angewandt werden sollte, um die vorliegende Problematik händeln zu können. Derartige Schwierigkeiten kommen beispielsweise zu Tage, wenn ein Teil der Zielvariablen recht schwer/teuer zu beschaffen ist. So liegen also für eine gewisse Zahl m der n Beobachtungen sowohl die Prädiktionsmessungen, als auch der zugehörige Response vor. Für den verbleibenden Teil $n - m$ der Subjekte, können

aber aufgrund bestimmter Umstände, die Einflussvariablen relativ leicht erhoben werden, wohingegen die entsprechende Zielvariable/Label nicht erfasst werden kann. In diesem Fall spricht man dann von halb-überwachtem (semi-supervised) Lern-Problem. Gesucht wird in diesem Sinne nämlich eine Methode, deren Lernalgorithmus sowohl die m Messungen mit beobachteter Zielgröße involvieren kann, also auch die verbleibenden $n - m$ Beobachtungen, für welche kein derartiges y_i realisiert werden konnte. Derartige Problematiken sind in dieser Arbeit allerdings nicht weiter von Bedeutung, weshalb die Thematik an dieser Stelle nicht weiter vertieft wird.
Nachdem die genutzten Methodiken sich auf den Bereich des SL's beschränken, soll auf eine vertiefende Beschreibung der Algorithmen des USL's verzichtet werden. Stattdessen wird der Fokus auf erstere genannte ML-Methoden, sowie deren Vergleich untereinander gesetzt. Im Zusammenhang des Vergleichens bzw. Gegenüberstellens von ML-Methoden stößt man unausweichlich auf die Wettbewerbsanalyse und damit auf das s.g. Benchmarking bzw. Benchmark-Studien. Dies wird mitsamt zugehöriger Thematiken nachfolgendes Kap. 3 ausmachen.

8.2 Ergänzungen zu Kap. 3 - Benchmark-Studie

8.2.1 Ergänzung Trainings-, Validierungs- und Testdatensatz - Prototyping



Quelle: (Zheng, 2015, Figure 3-1.)

Abb. 8.3: Veranschaulichung der Prototyping-Phase: Verwendung der Validierungs- und Trainingsdaten

8.2.2 Sammlung von Performance-Maßen

Klassifikationsaufgaben

Die nachfolgenden Performance-Maße für Klassifikationsaufgaben sind aus der Konfusionsmatrix ableitbar und die gewählten Bezeichnungen lassen sich entsprechend mittels Tab. 3.1 nachvollziehen. Die konkrete Berechnung der Performance-Maße orientiert sich an Jiao und Du (2016).

Performance-Maß	**Berechnung**
Sensitivität/True-Positive-Rate	$Sens = TPR = \frac{\#TP}{\#TP+\#FN} = \frac{\#TP}{RP}$
Spezifität	$Spez = \frac{\#TN}{\#TN+\#FP} = \frac{\#TN}{RN}$
False-Positive-Rate	$FPR = 1 - Spez = \frac{\#FP}{\#TN+\#FP}$
Accuracy	$ACC = \frac{\#TN+\#TP}{\#TN+\#TP+\#FN+\#FP} = \frac{\#TN+\#TP}{\Omega}$
Mean Missclassification Error	$MMCE = 1 - ACC = \frac{\#FP+\#FN}{\Omega}$
Positive-Predictive-Value/Precision	$PPV = \frac{\#TP}{\#TP+\#FP} = \frac{\#TP}{PP}$
False-Discovery-Rate	$FDR = 1 - PPV$
Jaccard-Index	$J = \frac{\#TP}{\#TP+\#FP+\#FN} = \frac{\#TP}{\Omega-\#TN}$
F_1-Score	$F_1 = \frac{2\cdot\#TP}{2\cdot\#TP+\#FP+\#FN} = \frac{2\cdot PPV\cdot Sens}{PPV+Sens}$
Balanced Accuracy	$BACC = \frac{1}{2}(Sens + Spez)$
Matthew's Correlation Coefficients	$MCC = \frac{\#TP\cdot\#TN-\#FP\cdot\#FN}{\sqrt{(\#TP+\#FP)+(\#TP+\#FN)+(\#TN+\#FP)+(\#TN+\#FN)}}$

Tab. 8.1: Performance-Maße für Klassifikationsaufgaben, ableitbar aus Konfusionsmatrix in Tab. 3.1

Regressionsaufgaben

Die nachfolgenden Performance-Maße stellen eine Sammlung der häufig verwendeten Maße zur Beurteilung der Güte im Falle Regressionsaufgaben dar und sind in Anlehnung an Witten et al. (2011, Kap. 5) bereitgestellt. Wiederum sind auf einem Testdatensatz basierte, prognostizierte Werte durch ein Zirkumflex ($\wedge$) kenntlich gemacht, um so tatsächliche Werte $y_1, \ldots, y_N$ von prognostizierten Werten $\hat{y}_1, \ldots, \hat{y}_N$ unterscheiden zu können. Zudem wird ein waagrechter Strich ($-$) oberhalb von Vektoren weiterhin zur Kenntlichmachung des arithmetischen Mittels genutzt.

Performance-Maß	**Berechnung**
Mean-Squared Error	$MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$
Root Mean-Squared Error	$RMSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2}$
Mean-Absolute Error	$MAE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{n=1}^{N} \lvert y_n - \hat{y}_n \rvert$
Relative-Squared Error	$RSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\sum_{n=1}^{N} (y_n - \hat{y}_n)^2}{\sum_{n=1}^{N} (y_n - \overline{y})^2}$
Root Relative-Squared Error	$RRSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sqrt{\frac{\sum_{n=1}^{N} (y_n - \hat{y}_n)^2}{\sum_{n=1}^{N} (y_n - \overline{y})^2}}$
Relative-Absoulte Error	$RAE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\sum_{n=1}^{N} \lvert y_n - \hat{y}_n \rvert}{\sum_{n=1}^{N} \lvert y_n - \overline{y} \rvert}$
Correlation Coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}},\ S_{PA} = \frac{\sum_{n=1}^{N} (\hat{y}_n - \overline{\hat{y}})(y_n - \overline{y})}{N-1},$ $S_P = \frac{\sum_{n=1}^{N} (\hat{y}_n - \overline{\hat{y}})}{N-1},\ S_A = \frac{\sum_{n=1}^{N} (y_n - \overline{y})}{N-1}$

Tab. 8.2: Performance-Maße für Regressionsaufgaben

8.2.3 SMBO

Die in dieser Sektion inbegriffene Erklärung zur SMBO sollte insbesondere als Ergänzung zu Abb. 3.3 angesehen werden und soll nun mit der nachfolgenden Auflistung schrittweise erläutert werden:

1. Ziehen eines initialen Designs von $n_{initial}$ Punkten $\boldsymbol{x}^{(j)}$, wobei $j = 1, \ldots, n_{initial}$ ist, und Evaluierung von f an diesen Punkten, um so die Outcomes $y^{(j)} = f(\boldsymbol{x}^{(j)})$ bestimmen zu können.

2. Anpassung des surrogaten Modells an alle evaluierten Punkte $\boldsymbol{x}^{(j)} \in \mathcal{X}$ und zugehörige $y^{(j)}$ - in Form des Tupels $(\boldsymbol{x}^{(j)}, y^{(j)})$ - aus vorherigem Schritt als Datengrundlage zur Generierung des initialen surrogaten Modells. Einer der Hauptfaktoren, der die Wahl des surrogaten Modells steuert, ist die Struktur des Input-Raumes $\mathcal{X}$. Insbesondere im Falle von $\mathcal{X} \subset \mathbb{R}^d$ empfiehlt sich das s.g. Kriging, wobei ein entsprechender Ansatz durch den „Efficient Gobal Optimization“ (EGO) Algorithmus gegeben ist.

3. Ein Ergänzungskriterium (engl.: „Infill Criterion“) stellt C weitere Punkte $\boldsymbol{x}^{(j+c)}$ bereit, mit $c = 1, \ldots, C$. Dieses Kriterium ist definiert auf $\mathcal{X}$ und operiert auf der surrogaten Funktion $\hat{f}$ derartig, dass möglichst „vielversprechende“ Punkte für die Optimierung gewählt werden. Vielversprechend bedeutet in dem Sinne, dass die Punkte entweder einen

guten erwarteten, objektiven Wert haben oder hohes Potential, um die Qualität des surrogaten Modells zu verbessern. Hierbei fällt die Wahl zumeist auf das Maß „Expected Improvement“ (EI) oder aber auch „Maximum Probability of Improvement“ (MPI).

4. Evaluierung der neuen, durch Schritt (3) bereitgestellten Punkte anhand von f und Hinzufügen der neuen Tupel $(\boldsymbol{x}^{(j+c)}, y^{(j+c)})$ zum Design.

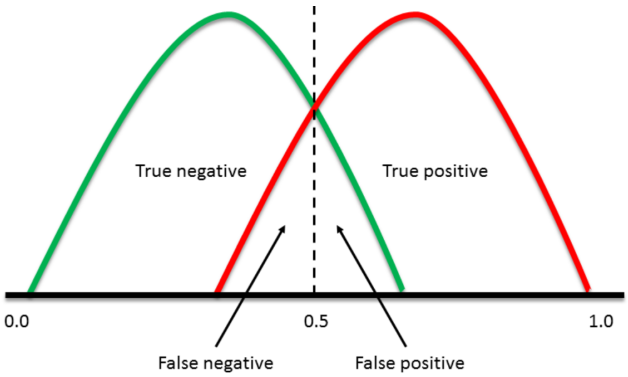
5. Bei Nicht-Erschöpfung des Budgets oder eines anderen terminierenden Kriteriums Wiederholung ab Schritt (2), ansonsten Übergang zu Schritt (6).

6. Rückgabe der bereitgestellten Lösung für das Optimierungsproblem.

8.2.4 ROC-Analyse

Probabilistische Klassifikatoren und Threshold

Zu beachten gilt, dass die bisher betrachteten Klassifikationen zwar binären Types sind, allerdings häufig die genutzten Methodiken Wahrscheinlichkeiten für die jeweiligen Klassenzugehörigkeiten zurückgeben und damit gewissermaßen einen kontinuierlichen Output liefern, weshalb dann von probabilistischen Klassifikatoren gesprochen wird. Nachfolgende Beschreibungen zur ROC-Kurven gehen ausschließlich von derartigen probabilistischen Klassifikatoren aus. Mittels der zurückgegebenen Wahrscheinlichkeiten lassen sich die Verteilungsfunktionen der beiden Klassen wie in Abb. 8.4 darstellen, wobei Rot die positive und Grün die negative Klasse abbildet.



Quelle: Hallinan (2014)

Abb. 8.4: Beispielhafte Veranschaulichung zur ROC-Kurve: Klassenzuweisung in Abhängigkeit von festgesetztem Schwellenwert θ

Gleichzeitig ist es nun also erforderlich einen Schwellenwert (Threshold) θ festzusetzen, der in Abhängigkeit von der berechneten Wahrscheinlichkeit die Klassenzuweisung ermöglicht. Wenngleich in Abb. 8.4 $\theta = 0.5$ gewählt wurde, so kann allgemein jedoch die folgende Formulierung geltend gemacht werden, um einen probabilistischen auf einen diskreten bzw. binären Klassifikator zurückzuführen:

$$\text{Klasse} = \begin{cases} \text{negativ (grün)}, & \text{falls } P(X) < \theta \\ \text{positiv (rot)}, & \text{falls } P(X) \geq \theta \end{cases}$$

Konstruktion der ROC-Kurve

Mit dem Hintergrundwissen der vorhergehenden Absätze kann nun die Beschreibung zur Erzeugung der ROC-Kurve erfolgen, wobei hier gleichzeitig Bezug auf Fawcett (2005) genommen sei. Zunächst werden die Daten gemäß ihrem Score bzw. der Wahrscheinlichkeit ihrer Klassenzugehörigkeit folgend sortiert. Jeder Punkt der Stufenfunktion ist dann gekennzeichnet durch einen ihn hervorbringenden (Score-) Schwellenwert θ. Ein Schwellenwert von $\theta = \infty$ erzeugt den Punkt $(0, 0)$ im ROC-Raum. Mit Herabsetzen dieses Wertes ändert sich die Zahl der Zuordnung zur positiven bzw. negativen Klasse und damit letztlich die TPR, sowie FPR, was anhand von Abb. 3.4 verdeutlicht werden soll. Dies führt entsprechend also zur Kennzeichnung von Punkten unterschiedlicher Koordinaten im ROC-Raum, sodass sich durch - stufenweises - Verbinden der Punkte eine monoton steigende Funktion ergibt, welche im Punkt $(1, 1)$ endet.

Iso-Accuracy Lines (IALs)

Ausweitend kann nun auch ein direkter Zusammenhang zwischen dem ACC und der TPR bzw. FPR hergestellt werden. [s. Scheipl et al. (2018, Kap. 8, S. 20)]

$$\begin{aligned} ACC = \frac{\#TP + \#TN}{n} &= \frac{\#TP}{n_+} \cdot \frac{n_+}{n} + \frac{n_- - \#FP}{n} \\ &= \frac{\#TP}{n_+} \cdot \frac{n_+}{n} + \frac{n_-}{n} - \frac{\#FP}{n_-} \cdot \frac{n_-}{n} \\ &= TPR \cdot \pi_+ \quad + \pi_- - FPR \cdot \pi_-, \end{aligned}$$

mit n = *Anzahl der Beobachtungen*

n_+, n_- = *Anzahl der positiven, negativen Beobachtungen*

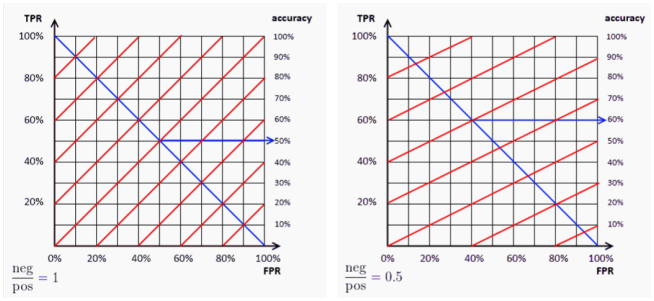
π_+, π_- = *Anteil der positiven, negativen Beobachtungen*

Dies kann folglich auch umgeschrieben werden, sodass die TPR in Abhängigkeit von dem ACC, sowie der FPR mittels Intercept und Steigungs-Parameter dargestellt werden kann:

$$TPR = \frac{ACC - \pi_-}{\pi_+} + \frac{\pi_-}{\pi_+} \cdot FPR \tag{8.1}$$

Aus Gl. 8.1 wird ersichtlich, dass der Quotient $\frac{\pi_-}{\pi_+}$ den Slope-Parameter darstellt. Demgemäß ergeben sich durch Änderung der Anteile positiver, sowie negativer Beobachtungen unterschiedliche Steigungen der TPR. Ändert man die Genauigkeit ACC, so führt dies zu paral-

lelen Linien mit demselben Slope-Parameter, nachdem das ACC im Intercept enthalten ist. Folglich kann geschlossen werden, dass „höhere“ Linien im Sinne nord-westlicher Lage besser hinsichtlich des ACC's sind, da diese dann mit einer größeren Genauigkeit einhergehen.
Trägt man nun für einen festen Steigungsparameter unterschiedliche Werte des ACC's wie in Abb. 8.5 ab, so ergeben sich die s.g. Iso-Accuracy Lines (IAL's), wobei also auf diesen jeder Punkt dasselbe ACC aufweist. Um umgekehrt für einen interessierenden Punkt innerhalb dieser Graphik das jeweilige ACC zu bestimmen, muss dieser Punkt als Schnittpunkt der zugehörigen roten IAL mit der absteigenden, blau gekennzeichneten Diagonalen ausgemacht werden. Die mathematische Begründung zu diesem Sachverhalt kann mithilfe von Gl. 8.4 nachvollzogen werden, deren Gültigkeit mithilfe von Gl. 8.2 und 8.3 hergeleitet wird.



Quelle: (Scheipl et al., 2018, Kap. 8)

Abb. 8.5: Beispielhafte Veranschaulichung zur ROC-Kurve: Iso-Accuracy Linien (IAL's) und Bestimmung des zugehörigen ACC's

Für die absteigende, blaue Diagonale aus Abb. 8.5 gilt:

$$TPR = 1 - FPR \leftrightarrow FPR = 1 - TPR \tag{8.2}$$

Zudem gilt gemäß 8.1:

$$TPR = \frac{ACC - \pi_-}{\pi_+} + \frac{\pi_-}{\pi_+} \cdot FPR \tag{8.3}$$

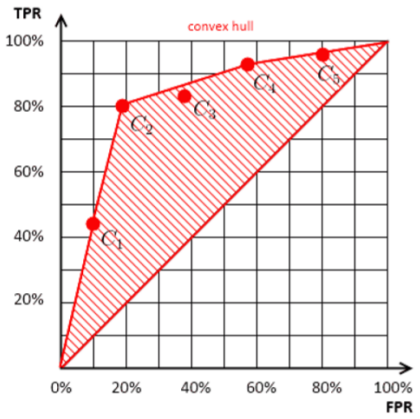
Zur Bestimmung der Schnittpunkte der IAL's mit der absteigenden Diagonalen kann entsprechend die nachfolgende Gleichung gelten gemacht werden:

$$\begin{aligned} TPR &= \frac{ACC - \pi_-}{\pi_+} + \frac{\pi_-}{\pi_+} \cdot (1 - TPR) \\ &= \frac{ACC}{\pi_+} - \frac{\pi_- \cdot TPR}{\pi_+} \\ &= \frac{ACC - \pi_- \cdot TPR}{\pi_+} \\ \Leftrightarrow ACC &= \pi_+ \cdot TPR + \pi_- \cdot TPR \quad \text{und} \quad \pi_+ + \pi_- = 1 \\ ACC &= TPR \end{aligned} \tag{8.4}$$

Daraus kann geschlussfolgert werden, dass am Schnittpunkt der absteigenden Diagonalen und der jeweiligen IAL der Wert des ACC's gleich dem der TPR ist. [vgl. Scheipl et al. (2018, Kap. 8)]

ROC Convex Hull (ROCCH)

Angenommen man hat nun fünf Klassifikatoren $C_1, C_2, \ldots, C_5$ und damit einen Punkt pro Klassifikator im ROC-Raum. Wie kann generell geprüft werden, ob einer dieser tendenziell den übrigen unterlegen ist? Dazu kann die s.g. konvexe Hülle - zu Englisch „Convex Hull" - herangezogen werden, weshalb man abgekürzt hier auch von der ROCCH spricht. Diese Hülle wird von einem Set von Punkten bzw. Klassifikatoren gebildet und entspricht einer stückweise linearen, konkav nach unten verlaufenden „Kurve" , wie eine solche beispielhaft in Abb. 8.6 zu sehen ist. Gemäß Provost und Fawcett (2001) ist ein Klassifikator ausschließlich dann potentiell optimal, wenn er auf dieser ROCCH liegt. Demnach ergibt sich für ein steigendes x bzw. FPR, ein monotoner (nicht-steigender) Slope-Parameter mit $k-1$ diskreten Werten, wobei k die Anzahl der ROCCH definierenden Klassifikatoren ist, einschließlich der degenerierten Klassifikatoren, welche die Endpunkte der Hülle bilden [vgl. Fawcett und Niculescu-Mizil (2007, S. 7)]. Entsprechend werden immer diejenigen Punkte im ROC-Raum verbunden, für die sich ein maximaler Steigungsparameter ergibt. Durch Wiederholen dieses Schrittes endet man schließlich im Punkt $(1, 1)$ und erhält alle Klassifikatoren mit minimal zu erwartenden Kosten auf der ROCCH.



Quelle: (Scheipl et al., 2018, Kap. 8)

Abb. 8.6: Beispielhafte Veranschaulichung zur ROC-Kurve: Konvexe Hülle (ROCCH)

Die ROCCH impliziert nun, dass Klassifikatoren, die sich auf der Hülle befinden bzw. diese bilden, darunter liegende - wie in Abb. 8.6 C_3 - dominieren und die höchste Genauigkeit im Sinne des ACC's für einige Klassenverteilungen bereitstellen. Hierbei sei erwähnt, dass Klassifikatoren, welche sich unterhalb der Hülle befinden, suboptimal im Hinblick auf niedriger TPR oder höhere FPR verglichen mit denjenigen auf der ROCCH sind.

Wie ggf. bereits deutlich geworden ist, repräsentiert jedes Linienelement der ROCCH eine IAL für eine bestimmte Klassenverteilung (Slope) und ein ACC. Hierbei gilt folgender Zusammenhang:

- $\frac{\pi_i}{\pi_+} > 1$:
  - Verteilung mit mehr richtig-negativen Beobachtungen
  - Steilerer Slope-Parameter
  - Klassifikatoren weiter links (weniger falsch-positiv) sind genauer
- $\frac{\pi_i}{\pi_+} < 1$:
  - Verteilung mit mehr richtig-positiven Beobachtungen
  - Flacherer Slope-Parameter
  - Klassifikatoren weiter rechts (mehr falsch-positiv) sind genauer

Entsprechend kann der optimale Klassifikator in Abhängigkeit vom Steigungsparameter bzw. dem Verhältnis von negativen zu positiven Beobachtungen bestimmt werden, indem die zugehörige IAL ermittelt wird, sowie deren Schnittpunkt mit der ROCCH. Dieser lässt auf den entsprechenden Klassifikator schließen.

8.3 Ergänzungen zu Kap. 4 - Ausgewählte ML-Methoden

8.3.1 Bagging: Out-of-Bag (OOB) Fehlerschätzung

Die nachfolgenden Erklärungen richten sich nach Breiman (1994), sowie Breiman (1996) und Zhou (2012, Kap. 3.2). Wie bereits angesprochen kann der Testfehler des Bagging-Modells geschätzt werden, ohne das die Nutzung von Kreuzvalidierung oder einer Annäherung über einen Validierungsdatensatz erforderlich ist. Insbesondere sind die OOB-Schätzungen gegenüber der Kreuzvalidierung unverzerrt. Wie Breiman (1994) indizierte, ist die Wahrscheinlichkeit - gegeben T Trainingsdaten -, dass das t-te Sample $0, 1, 2, \ldots$-mal gezogen wird, approximativ Poisson-verteilt mit $\lambda = 1$. Entsprechend ergibt sich die Wahrscheinlichkeit, dass dieser t-te Datenpunkt mindestens einmal gezogen wird als $1 - (1 - 1/n)^n \xrightarrow{n\to\infty} 1 - (1/e) \approx 0.632$. Anders ausgedrückt bleiben für jeden Basis-Lerner innerhalb des Baggings in etwa 36.8% des originalen Trainingsdatensatzes während des Trainingsprozesses unbenutzt. Somit können diese „Out-of-Bag“ (OOB) Daten für die Bestimmung der Güte der Basis-Lerner und schließlich für die Berechnung des Generalisierungsfehlers des Bagging-Ensembles herangezogen werden. Um nun die OOB-Schätzung bestimmen zu können, ist es zunächst erforderlich den Subset des Trainingsdatensatzes jedes einzelnen Basis-Lerners zu kennen. So kann nun $\hat{f}^{OOB}(\boldsymbol{x})$ als OOB-Prädiktion von $\boldsymbol{x}$ errechnet werden, wobei jeweils nur die Lerner eingeschlossen sind,

die nicht auf Basis von $\boldsymbol{x}$ trainiert wurden. Damit kann f^{OOB} beispielsweise folgendermaßen beschrieben werden:

$$\hat{f}^{OOB}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{b=1}^{B} \mathbb{I}(f^b(\boldsymbol{x}) = y) \cdot \mathbb{I}(x \notin \mathcal{L}^b) \tag{8.5}$$

Schließlich kann dann die OOB-Schätzung für den Generalisierungsfehler des Baggings - wie in Gl. 8.6 dargestellt - berechnet werden.

$$\epsilon^{OOB} = \frac{1}{|\mathcal{L}|} \sum_{(\boldsymbol{x},y) \in \mathcal{L}} \mathbb{I}(f^{OOB}(\boldsymbol{x}) \neq y) \tag{8.6}$$

Diese OOB-Beobachtungen können auch auf andere Weise als zur Bestimmung des Generalisierungsfehlers genutzt werden. Beispielsweise bei der Verwendung von Entscheidungsbäumen als Basis-Lerner um die posteriori Wahrscheinlichkeit jedes Knotens eines jeden Baumes zu berechnen. Ebenso sind die OOB-Daten aber ein hilfreiches Werkzeug bei s.g. Random Forests, welche eine Erweiterung des Baggings darstellen.

8.3.2 RF: Variable Importance

Im allgemeinen Fall werden RFs eine Reihe von Variablen übergeben, die als potentielle Einflussgrößen auf den interessierenden Response erachtet werden können. Dabei weisen aber die wenigsten Variablen die gleiche Wichtigkeit bzw. Effekt auf bzw. sind wohl die meisten Variablen völlig ohne Einfluss auf diese Zielgröße. Um die Relevanz der einzelnen j Variablen innerhalb der RFs ausmachen zu können, ist es sinnvoll den Beitrag der Input-Variablen zur Prädiktion auszumachen. Zwei mögliche Lösungsansätze hierzu werden in Anlehnung an James et al. (2013, Kap. 15.3.2), sowie Scheipl et al. (2018, Kap. 4) in dieser Untersektion beschrieben.

Eine Möglichkeit die s.g. Variablen-Wichtigkeit - vom Englischen „Variable Importance“ abgeleitet - zu messen, lässt sich anhand der bereits in Sektion 3.4 dargestellten Gütemaße bewerkstelligen. Dabei wird bei jedem Split von Baum $\hat{T}^b$ die Verbesserung innerhalb des jeweiligen Splitkriteriums (z.B. Gini-Index) als Maß der Wichtigkeit der entsprechenden Splitvariable zugeschrieben. Diese Steigerung der Genauigkeit kann dann für jede Variable über alle Bäume akkumuliert und gemittelt werden, womit sich schließlich ein Möglichkeit zur Messung der Variablen-Wichtigkeit ergibt. Nachdem bei jedem Split, an welchem die betrachtete Variable ausschlaggebend ist, die Abnahme des jeweiligen Kriteriums verglichen mit dem vorherigen Zustand begutachtet wird und dieses schließlich über alle Bäume des Waldes aufsummiert wird, sind diejenigen Variablen als einflussreich zu erachten, die einen hohen Wert des Wichtigkeitsmaßes aufweisen.

Doch es kann auch auf eine andere Methodik und damit auf die OOB-Stichproben zurückgegriffen werden. Wie bereits zuvor erwähnt beschränkt sich die Nutzung der OOB-Stichproben

nämlich nicht nur auf die Berechnung des Generalisierungsfehlers. Um die Relevanz einer Variable als Prädiktor beurteilen zu können, berechne man zunächst die Prognosegenauigkeit dieser, beispielsweise über das Genauigkeitsmaß ACC. Diese wird für die jeweilige OOB-Stichprobe bestimmt, nachdem der b-te Entscheidungsbaum erzeugt wurde. Anschließend werden die Werte der j-ten Variable innerhalb der OOB-Stichproben zufällig permutiert und erneut die Genauigkeit der Vorhersage erfasst. Der sich aus der Permutation ergebende Verlust der Genauigkeit - Differenz vor und nach Permutation - wird über alle Bäume gemittelt und dient schließlich als Maß für die Wichtigkeit der j-ten Variable innerhalb des RF's. Ist dieser Wert groß, so impliziert dies wiederum eine stärkere Verbindung zwischen Prädiktor und Response.
Auch wenn beide Methodiken zumeist wohl recht ähnliche Ergebnisse/Rankings ergeben, so sind die Messungen hinsichtlich der Wichtigkeit anhand der OOB-Stichproben einheitlicher über die Variablen hinweg. Dies lässt sich dadurch begründen, dass durch die Randomisierung effektiv eine Art Nulleffekt - ähnlich wie durch das Nullsetzen einer Variable etwa im linearen Modell - hervorgerufen wird. Entsprechend wird dann also nur der Effekt der ausgeschlossenen Variable auf die Prädiktion nicht gemessen. Passt man stattdessen das gesamte Modell ohne diese Variable an, so würden andere Einflussgrößen als Surrogate erachtet werden, was letztlich also die Effekte innerhalb des Modells ändert.
Letztlich kann dann also der Verlust an Genauigkeit - ausgedrückt als Prozentsatz des maximalen Verlusts - für jede Variable j in einer Grafik bzw. genauer als Balken-/Säulendiagramm dargestellt werden. Demzufolge ist der Balken bzw. die Säule von derjenigen Variable am größten, deren Weglassen aus dem Modell/Nichtbeachtung des Effektes am stärksten zum Tragen kommt und entsprechend die wohl wichtigste Variable innerhalb des RF's darstellt.

8.3.3 Adaptives Boosting (AdaBoost)

Adaptives Boosting (AdaBoost) stellt eine Boosting-Methode für binäre Klassifikationen dar und soll nun folgend mithilfe von Scheipl et al. (2018, Kap. 7), sowie Schapire und Freund (2012) beschrieben werden. Kernstück des Ganzen bildet die sequentielle Anwendung eines Basis-Lerners auf gewichtete Trainingsbeobachtungen. Dabei wird nach jeder Anpassung des Basis-Lerners mehr auf falsch klassifizierte Instanzen fokussiert, indem diese aktuellen Beobachtungen der Fehlklassifikation mit höherer Gewichtung in der nächsten Iteration Berücksichtigung finden.
Man gehe nun von einer binären Zielvariable aus, die entweder der negativen oder positiven Klasse zugeteilt werden kann, sodass gilt $y \in \{-1; 1\}$, und einem schwachen Basis-Lerner eines Hypothesen-Raumes $\mathcal{H}$. Des Weiteren nehme man Modelle des Basis-Lerners $b^{(m)}$ an, die als binäre Klassifikatoren in $\mathcal{Y} = \{-1; 1\}$ überführen. Eine Prädiktion wird schließlich

durch lineare Kombination der einzelnen Basis-Modelle $b^{(m)}$ wie in Gl. 8.7 durchgeführt.

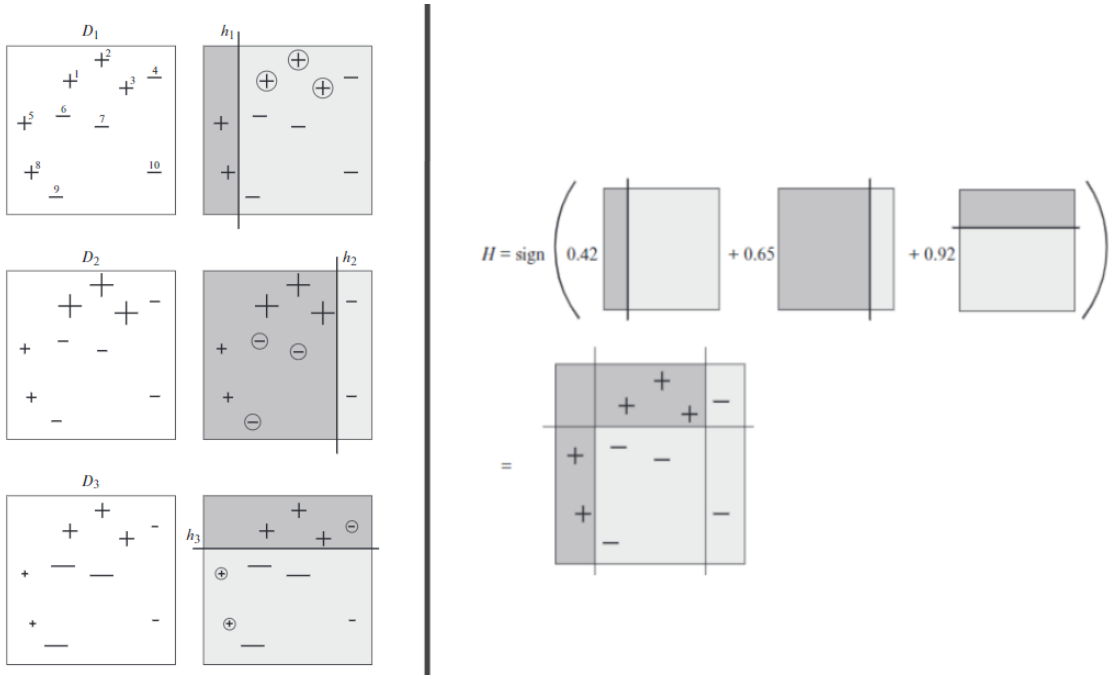
$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \omega^{(m)} \cdot b^{(m)}(\boldsymbol{x}) \tag{8.7}$$

mit $b^{(m)}$ = *m-tes Basis-Modell zu den Beobachtungen* $\boldsymbol{x}$,

$\omega^{(m)}$ = *Gewichtung zu dem m-ten Basis-Modell.*

Die Gewichte $\omega^{(m)}$ werden dabei über den Boosting-Algorithmus bestimmt und entsprechend denjenigen Basis-Modellen höhere Effekte zugesprochen, die eine höhere prädiktive Genauigkeit aufweisen. Gleichzeitig stellt die Anzahl an Iterationen M den wichtigsten Tuning-Parameter dar. Die diskrete Prädiktionsfunktion ergibt sich entsprechend als:

$$h(\boldsymbol{x}) = sign(f(\boldsymbol{x})) \in \{-1; 1\} \tag{8.8}$$



Quelle: Schapire und Freund (2012)

Abb. 8.7: Beispielhafte Veranschaulichung zu AdaBoost
links: 3 Iterationen des Algorithmus mit jeweiliger Klassifikation über einen schwachen Basis-Lerner und vorzunehmender Gewichtung der Beobachtungen;
rechts: Kombination der drei Basis-Lerner zu einem starken Klassifikator

Das beschriebene Vorgehen bei AdaBoost lässt sich auch gut bildlich illustrieren. Für die folgende Erklärung sei daher auf Abb. 8.7 Bezug genommen: man gehe von $n = 10$ Beobachtungen und $M = 3$ Iterationen aus, sowie beispielsweise dem Stamm eines Entscheidungsbaumes als Basis-Lerner. Vereinfacht trifft dieser hier Klassifikationen in Form von vertika-

len oder horizontalen Einteilungen der Ebene. Die tatsächlich beobachteten y-Werte treten jeweils in Form eines negativen $-$ oder positiven $+$ Labels zutage. Gleichzeitig veranschaulichen grau hinterlegte Grafiken die durch den Basis-Lerner vorgenommene Aufteilung, wobei dunkelgraue Flächen in der m-ten Iteration die Klassenzuweisung zu „+“ und entsprechend hellgraue Flächen zur „−“ Klasse verdeutlichen. Zudem soll anhand der Größe der Labels ersichtlich gemacht werden, welche Beobachtungen aufgrund der fehlerhaften Zuweisung in der vorhergehenden Iteration mit einer größeren Gewichtung bei der Anpassung des nächsten Basis-Lerners einhergehen. Insbesondere findet während der ersten Iteration also keine Gewichtung statt.
Die Kombination über unterschiedliche Gewichtung der drei schwachen Basis-Lerner ist schließlich im rechten Teil der Abb. 8.7 dargestellt und entspricht damit dem Kernstück der AdaBoost-Prozedur. Der so entstehende finale Klassifikator weist in diesem Beispiel sogar nach nur drei Iterationen keinerlei Fehlzuweisung mehr auf.
Die Anpassungen über AdaBoost folgen der Struktur eines additiven Modells mit Basisfunktionen $b^{(m)}(\boldsymbol{x})$. Insbesondere kann gemäß Hastie et al. (2008) gezeigt werden, dass die Prozedur von AdaBoost der Minimierung der exponentiellen Verlustfunktion in jeder Iteration m entspricht und demzufolge Gl. 8.9 Anwendung findet. Hierbei wird über ω und b minimiert und $\hat{f}^{(m-1)}$ stellt die Boosting-Anpassung während Iteration $m-1$ dar.

$$\sum_{i=1}^{n} exp\Big(-y_i \cdot (\omega \cdot b(x_i) + \hat{f}^{(m-1)}(x_i))\Big) \tag{8.9}$$

mit $\hat{f}^{(m-1)}$ = *Angepasstes Boosting-Modell zu Iteration* $m-1$.

Langwährende Diskussionen bestehen beim AdaBoost hinsichtlich des Aspekts von Overfitting. Ausschlaggebend ist hierbei das Stopp-Kriterium bzw. die Anzahl der Iterationen M: ein zu großer Wert von M führt zu äußerst komplexen Lösungen und damit zu Overfitting, wohingegen ein zu klein gewähltes M zu wenig Anpassung an die Daten und damit Underfitting hervorruft. Wenngleich es durchaus zu Overfitting kommen kann, so kann experimentell (s. beispielsweise Schapire und Freund (2012)) gezeigt werden, dass die AdaBoost-Prozedur relativ resistent gegenüber dieser Problematik ist und dadurch an Attraktivität in der Anwendung als ML-Methode gewinnt.

8.3.4 Logit-Modell (mit Penalisierung)

Annahmen im GLM mit Logit-Link

Hierbei wird wie im linearen Modell (LM) die Annahme getroffen, dass die binären Zielvariablen für die $i = 1, \ldots, N$ Personen bedingt unabhängig sind. Gleichzeitig gilt im Rahmen der GLM's, dass die Zielgrößen nicht normalverteilt sein müssen, jedoch besteht die Annahme einer Verteilung, die der Exponentialfamilie zugehört, also in diesem Fall die Binomialverteil-

lung. In Abgrenzung zum LM setzt das GLM keinen linearen Zusammenhang zwischen der abhängigen und den unabhängigen Variable voraus, allerdings eine lineare Beziehung zwischen dem im Sinne der durch die Link-Funktion transformierten Response. Auch gilt die Annahme unabhängiger Fehler, wobei diese wiederum nicht normalverteilt sein müssen. Gleichzeitig ist keine Homogenität der Varianzen erforderlich und ist tatsächlich in einigen Fällen aufgrund der Modellstruktur nicht einmal möglich, sodass es zu Überdispersion kommen kann. Des Weiteren sei zu erwähnen, dass $\boldsymbol{\eta}$ als linearer Prädiktor bezeichnet wird und demgemäß $\eta_i = \sum_{q=0}^{P} x_{iq}\beta_q$ gilt. Insgesamt lässt sich hier also $\boldsymbol{X} = (\mathbf{1}, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_N)$ als eine $q \times (N+1)$ Matrix auffassen, welche die Beobachtungen der einzelnen Instanzen zu den jeweiligen Prädiktoren, sowie den Offset inne hat, und $\boldsymbol{\beta}$ als Vektor, der die Effekte der einzelnen Einflussgrößen mitsamt Intercept beinhaltet.
Zusammenfassend liegen dem GLM mit Logit-Link also die in Tab. 4.2 gelisteten Annahmen zugrunde. Gleichzeitig gilt es noch einmal genauer den Zusammenhang zwischen Responsefunktion $h(\cdot)$ und Linkfunktion $g(\cdot)$ zu verdeutlichen:

$$\begin{gathered} E(y_i|\boldsymbol{x}_i) = \mu_i = \pi_i = h(\boldsymbol{x}_i^T\boldsymbol{\beta}) \\ \Longleftrightarrow \\ g(\pi_i) = g(\mu_i) = h^{-1}(\mu_i) = h^{-1}[E(y_i|\boldsymbol{x}_i)] = \boldsymbol{x}_i^T\boldsymbol{\beta} = \eta_i \end{gathered}$$

Maximum-Likelihood-Schätzung (MLE) für GLM mit Logit-Link

Die nachfolgenden Beschreibungen zum MLE orientieren sich hauptsächlich an Fahrmeir et al. (2009, S. 198-201). Vorrangiges Ziel der statistischen Inferenz ist - wie bei der LM's auch - die Schätzung der Effektstärken, welche in den β-Parametern inbegriffen sind, sowie das Testen von Hypothesen bezüglich dieser. Diese Schätzungen basieren auf der (vollständigen) Maximum-Likelihood-Methode. Als Grundlage dienen hierbei die gegebenen Daten $(y_i, \boldsymbol{x}_i))$, wobei der Index $i = 1, \ldots, N$ für die jeweils betrachtete Person steht. Es ergibt sich nun aufgrund der (bedingten) Unabhängigkeit der Zielvariablen die s.g. Likelihood $L(\boldsymbol{\beta})$ als Produkt der einzelnen Dichten $y_i|\boldsymbol{x}_i$, welche über $\pi_i = E(y_i|\boldsymbol{x}_i) = h(\boldsymbol{x}_i^T\boldsymbol{\beta})$ vom zu schätzenden Parameter $\boldsymbol{\beta}$ abhängen. Entsprechend gilt dann:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(y_i|\boldsymbol{\beta}, \boldsymbol{x}_i)$$

Aufgrund der einfacheren Handhabung wird i.A. nun aber nicht die Likelihood, sondern stattdessen die logarithmierte Likelihood (log-Likelihood) $l(\boldsymbol{\beta}) = \log(L(\boldsymbol{\beta}))$ verwendet. Hierauf aufbauend soll nun das schrittweise Vorgehen zur Berechnung der β-Schätzer im Falle des Logit-Modells beschrieben werden.

Ermittlung der Likelihood Konkret führe man sich noch einmal vor Augen, dass im Wesentlichen die folgenden Sachverhalte gelten: $y_i \sim \text{Bin}(1, \pi_i)$ mit $\pi_i = P(y_i = 1) = E(y_i) = \mu_i$ und damit ist die Dichte darstellbar als $f(y_i|\pi_i) = \pi_i^{y_i} \cdot (1 - \pi_i)^{1-y_i}$. Es gilt zu beachten, dass über $\pi_i = h(\boldsymbol{x}_i^T\boldsymbol{\beta})$ die Dichte bei gegebenem $\boldsymbol{x}_i$ von $\boldsymbol{\beta}$ abhängt. So lässt sich nun die Likelihood bestimmen als

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{N} L_i(\boldsymbol{\beta}) = \prod_{i=1}^{N} \pi_i^{y_i} \cdot (1 - \pi_i)^{1-y_i},$$

wobei $L_i(\boldsymbol{\beta})$ den individuellen Likelihood-Beiträgen entspricht.

Bestimmung der log-Likelihood Wie bereits zu Beginn der Sektion erwähnt, bietet es sich an die log-Likelihood aufgrund umgänglicher Rechenmöglichkeit zu verwenden. Diese ergibt sich durch logarithmieren der Likelihood und hat für das binäre Logit-Modell dann entsprechend die folgende Form:

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{i=1}^{N} l_i(\boldsymbol{\beta}) = \sum_{i=1}^{N} [y_i \log(\pi_i) - y_i \log(1 - \pi_i) + \log(1 - \pi_i)] \\ &= \sum_{i=1}^{N} y_i \log\Big(\frac{\pi_i}{1 - \pi_i}\Big) + \log(1 - \pi_i), \\ \textit{wobei } \pi_i &= \frac{\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_i^T\boldsymbol{\beta})} \Leftrightarrow \log\Big(\frac{\pi_i}{1 - \pi_i}\Big) = \boldsymbol{x}_i^T\boldsymbol{\beta} = \eta_i \end{aligned}$$

Aufstellen der Score-Funktion Um nun schließlich konkret die Parameterschätzer $\hat{\boldsymbol{\beta}}$ ermitteln zu können, bedarf es vorab der s.g. Score-Funktion. Diese ergibt sich als erste Ableitung der log-Likelihood:

$$\boldsymbol{s}(\boldsymbol{\beta}) = \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \frac{\partial l_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} s_i(\boldsymbol{\beta}),$$

wobei $s_i(\boldsymbol{\beta})$ den individuellen Beiträgen zur Score-Funktion entspricht.

Gemäß der Kettenregel kann nun die nachstehende Gleichheit gefolgert werden:

$$\frac{\partial l_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{\partial l_i}{\partial \eta_i} \frac{\partial \eta_i}{\partial \boldsymbol{\beta}} = \Big[y_i - \frac{\exp(\eta_i)}{1 + \exp(\eta_i)}\Big] \cdot \boldsymbol{x}_i,$$

mit q-dimensionalen Vektor $\partial \eta_i / \partial \boldsymbol{\beta} = \boldsymbol{x}_i$.

Durch Einsetzen von π_i gilt schließlich $s_i(\boldsymbol{\beta}) = \boldsymbol{x}_i(y_i - \pi_i)$ und somit insgesamt:

$$\begin{aligned}\boldsymbol{s}(\boldsymbol{\beta}) &= \sum_{i=1}^{N} \boldsymbol{x}_i(y_i - \pi_i) \\ &= \sum_{i=1}^{N} \boldsymbol{x}_i\Big(y_i - \frac{\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}\Big)\end{aligned}$$

Es zeigt sich folglich für $s(\boldsymbol{\beta})$ eine nicht-lineare Abhängigkeit von $\boldsymbol{\beta}$. Gleichzeitig gilt aufgrund des Zusammenhangs $E(y_i) = \pi_i$, dass der Erwartungswert der Scorefunktion Null ist und damit $E(\boldsymbol{s}(\boldsymbol{\beta})) = 0$. Schließlich erhält man durch Nullsetzen der Score-Funktion, die s.g. ML-Gleichung, welche nach dem q-dimensionalen β-Vektor aufgelöst werden kann. Zur konkreten Lösung des nicht-linearen Gleichungssystems nach $\hat{\boldsymbol{\beta}}$ ist i.A. eine iterative Methode nötig, wobei sich hier beispielsweise der Newton-Raphson- oder Fisher-Scoring-Algorithmus anbieten.

Aufstellen der Informationsmatrix Des Weiteren bedarf es insbesondere zur Ermittlung der Kovarianzmatirx des MLE's $\hat{\boldsymbol{\beta}}$ die (beobachtete und die daraus abgeleitete) erwartete Informationsmatrix. Hierbei stellt das Logit-Modell einen Spezialfall dar, da beobachtete und erwartete Informationsmatrix identisch sind. Entsprechend können diese durch die mit -1 multiplizierte zweite Ableitung der log-Likelihood ermittelt werden:

$$H(\boldsymbol{\beta}) = -\frac{\partial^2 l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = E(-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}) = \mathrm{Cov}(\boldsymbol{s}(\boldsymbol{\beta})) = E(\boldsymbol{s}(\boldsymbol{\beta}) \cdot \boldsymbol{s}^T(\boldsymbol{\beta}))$$

Überblick zu Mechanismen der Regularisierung

Zunächst sollen kurz mittels Li und Chen (o. J.) ein kurzer Überblick über potentiell zur Verfügung stehende Verfahren zur Regularisierung gegeben werden. Im Wesentlichen sind hier die nachfolgenden drei Vorgehensweisen denkbar:

- Auswahl einer Untermenge der Prädiktoren (Subset Selection): Hierbei soll nur ein Teil derjenigen P Prädiktoren gewählt werden, von denen man einen tatsächlichen Zusammenhang mit der Zielgröße erwartet. Als potentielle Kriterien für die Variablenselektion können beispielsweise das adjustierte R^2, das Akaike (AIC) oder Bayes'sche Informationskriterium (BIC), sowie Mallow's CP oder die FPR in Betracht gezogen werden.
  - Best Subset Selection: Auswahl desjenigen Subsets, mit der besten Performance.
  - Schrittweise Selektion (sequentiell): Kann vorwärts (forward), rückwärts (backward) oder als Kombination der beiden Vorgehen erfolgen. Ersteres geht von einem Nullmodell aus und ergänzt schrittweise eine Prädiktor, der die höchste Verbesserung der Modellanpassung erzielt, solange bis keine signifikante Besserung

hinsichtlich eines gewählten Kriteriums mehr erfolgt. Für die rückwärts gerichtete Selektion gilt umgekehrtes und man beginnt entsprechend mit dem vollen Modell.

- Dimensionenreduktion: Projektion der P Prädiktoren in einen D-dimensionalen Subraum, sodass $D < P$.
  - Erfolgt durch Computation von D unterschiedlichen Linearkombinationen der Variablen.
  - Beispiel PCA (Principal Component Analysis): Finden eines niedrig-dimensionalen Repräsentanten des Datensatzes, der zugleich möglichst viel der vorliegenden Variation umfasst.
- Shrinkage: Anpassung des Modells mit allen P Prädiktoren (volles Modell) und anschließende Regulierung/Bestrafung irrelevanter Einflussgrößen (ggf. Selektion eingeschlossen).
  - Beispielhaft sind Ridge (keine Variablenselektion) und LASSO zu nennen, ebenso wie das elastische Netz.

Regularisierung mittels Shrinkage-Methoden

Wie bereits in Untersektion 4.5.2 angesprochen wurde, können die non-linearen Gleichungssysteme aus Gl. 4.31 durch eine Taylorentwicklung erster Ordnung für π_i angenähert werden und so weitere Rechenschritte ermöglicht werden. Die nachfolgenden Beschreibungen hierzu orientieren sich an Schimek (2003, Kap. 4).

$$\pi_i = \tilde{\pi}_i + \frac{\partial \pi_i}{\partial \beta_0}(\beta_0 - \tilde{\beta}_0) + \sum_{p=1}^{P} \frac{\pi_i}{\beta_p}(\beta_p - \tilde{\beta}_p) \tag{8.10}$$

Die in Gl. 8.10 genutzten Tilde-Zeichen oberhalb der Skalaren dienen der Kenntlichmachung von Approximationen, also beispielsweise $\tilde{\pi}_i$ für die Annäherung an π_i. Gleichzeitig ergeben sich die partiellen Ableitungen wie nachstehend dargeboten:

$$\frac{\partial \pi_i}{\partial \beta_0} = \pi_i \cdot (1 - \pi_i) \quad \text{und} \quad \frac{\partial \pi_i}{\partial \beta_p} = \pi_i \cdot (1 - \pi_i) x_{ip} \tag{8.11}$$

Nun folgend wähle man die vereinfachte Darstellung/Zusammenfassung $\pi_i \cdot (1 - \pi_i) = w_i$.

$$\begin{gathered} \mathbb{1}^T \tilde{\boldsymbol{W}}^T \mathbb{1} \beta_0 + \mathbb{1} \tilde{\boldsymbol{W}}^T \boldsymbol{X} \boldsymbol{\beta} = \mathbb{1}^T (\boldsymbol{y} - \tilde{\pi} - \tilde{\boldsymbol{W}} \tilde{\eta}) \\ \text{und} \\ \boldsymbol{X}^T \tilde{\boldsymbol{W}}^T \mathbb{1} \beta_0 + (\boldsymbol{X}^T \tilde{\boldsymbol{W}}^T \boldsymbol{X} + \lambda \mathbb{I}) \cdot \boldsymbol{\beta} = \boldsymbol{X}^T (\boldsymbol{y} - \tilde{\pi} - \tilde{\boldsymbol{W}} \tilde{\eta}) \end{gathered} \tag{8.12}$$

Hierbei stellt $\boldsymbol{W}$ eine Diagonalmatrix bestehend aus den Elementen w_i dar. Wie bereits bei der standardmäßigen logistischen Regression zeigt, ist der Parameter β_0 bestimmt durch den

Anteil der Einsen in $\boldsymbol{y}$. Das oben stehende linearisierte System kann nun mithilfe iterativer Techniken gelöst werden. Hierzu empfiehlt Schimek (2003) als angemessene Initialwerte beispielsweise $\tilde{\beta}_0 = log(\frac{\overline{y}}{1-\overline{y}})$, mit $\overline{y} = \sum_i = 1^N y_i/N$, sowie $\tilde{\boldsymbol{\beta}} = 0$ zu wählen.

8.4 Ergänzungen zu Kap. 5: Software-Nutzung und Entwicklung einer Shiny App

8.4.1 Beispiel-basierte Erklärungsmethoden

Bisher fanden lediglich Modell-agnostische Interpretationsmethoden Anklang im Rahmen der Sektion 5.3. Dem gegenüberstehend soll hier nun kurz das Konzept der Beispiel-basierten Erklärungsmethoden umrissen werden und der Grundgedanke einiger hier zugehöriger Methoden offen gelegt werden. Dies fußt im Wesentlichen auf Molnar (2018, Kap. 6).
Wie anhand der Namensgebung bereits ersichtlich wird, werden im Zuge derartiger Methoden spezifischen Instanzen aus den Daten gezogen, um das Verhalten eines ML-Modells zugänglich zu machen. Der Wesentliche Unterschied zu Modell-agnostischen Methoden ist hierbei, dass Beispiele aus den Daten zur Interpretationszwecken genutzt werden an Stelle von Zusammenfassungen, wie diese von Modell-agnostischen Verfahren bereitgestellt wird. Gleichzeitig macht es allerdings nur Sinn, derartige Beispiel-basierte Techniken zu nutzen, falls die genutzten Instanzen in verständlicher Weise dargestellt werden können. Entsprechend bieten sich diese insbesondere im Falle von Bild- oder Text-Analysen an, wobei einzelne Instanzen in strukturierter Form darstellbar sind. Hat man dagegen einen Datensatz mit einer großen Anzahl zugehöriger Features, so ist eine übersichtlich Darstellung einzelner Instanzen zumeist nicht gewährleistet und damit von Beispiel-basierten Interpretationsmethoden abzuraten.
Wie funktioniert nun aber das Vorgehen in diesem Kontext der Interpretationsmöglichkeiten von ML-Methoden. Dazu spiele man vereinfacht gesprochen zunächst das folgende Szenario durch: einem Arzt wird vom Patienten die Symptome seiner Krankheit beschrieben. Rein aufgrund der Analogien, die der Arzt zu einem seiner früheren Patienten erkennt, fordert er einen Bluttest an, da er nun bereits eine Vermutung bezüglich der vorliegenden Diagnose hat. Letztlich kann dieses Denken bzw. Schließen über Ähnlichkeitsstrukturen direkt als Basis der Beispiel-basierten Interpretationsmethoden erachtet werden. So fußen einige ML-Methoden gar explizit auf einer derartigen Vorgehensweise. Beispielsweise werden neue Prädiktionen innerhalb von Entscheidungsbäumen derartig bestimmt. Hat man beispielsweise innerhalb der neuen Daten einen Mann von über 70 Jahren, so läuft die Prädiktion seines Outcomes über spezifische Äste, die aufgrund der Ähnlichkeitsstrukturen zu den Daten, auf Basis deren der Entscheidungsbaum trainiert wurde, festgesetzt wurden. Man versucht nun also diese verhältnismäßig einfache Denkweise auf komplexe Blackbox-Methoden zu übertragen, um deren Entscheidungen zugänglich zu machen.

In diesem Rahmen lassen sich nun die nachfolgenden Methodiken der Beispiel-basierte Interpretationsweise zuordnen:

- „Counterfactual Examples“ (Kontrafaktische Beispiele): Beschreiben kausale Zusammenhänge in der Form, dass eine hypothetische Realität aufgebaut wird, die den beobachteten Fakten widerspricht: „Wenn A nicht eingetreten wäre (*Grund*), dann auch B nicht (*Ereignis*)“. Im ML-Kontext wird dann eine Instanz betrachtet, deren prognostizierter Outcome das Ereignis ist und der Grund bzw. die Ursachen dafür sind durch die spezifischen Feature-Werte gegeben. Letztlich wird innerhalb dieser Interpretationsmethode die Änderung im Outcome betrachtet, wenn man die Werte der Einflussgrößen dieses Subjekts in bestimmter Weise ändert. Damit wird das Ziel verfolgt bestimmen zu könne, wie eine Instanz sich verändern müsste, um eine signifikante Änderung in dessen Prädiktion zu verursachen. Auf Basis dieser kontrafaktischen Beispiele soll der Nutzer der Interpretationsmethodik ein Verständnis dafür bekommen, wie die ML-Methode Prädiktionen trifft.

- „Adversial Examples“ (Kontradiktorische Beispiele): Derartige Instanzen sind kontrafaktische Beispiele mit dem Ziel die ML-Methode zu täuschen und Schwachstellen auszumachen, nicht sie zu interpretieren.

- „Prototypes and Criticisms“ : Kann unabhängig vom ML-Kontext genutzt werden, eignet sich jedoch auch zur Interpretation von Blackbox-Methoden, wobei Prototypen eine Auswahl repräsentativer Instanzen innerhalb der Daten sind und Kritikpunkte sind diejenigen Instanzen, die durch diese Prototypen nicht ausreichend beschrieben werden.

- Einflussreiche Instanzen: Bestimmung derjenigen Datenpunkte, die den größten Einfluss auf die Parameter des Prädiktionsmodells bzw. die Prädiktion selbst haben, um so das Verhalten der ML-Methodik nachvollziehen zu können.

- kNN-Modell: Die Nutzung interpretierbarer Modelle für die Zugänglichkeit zu Blackbox-Methoden wurde bereits in der vorhergehenden Untersektion 5.3.6 behandelt. Die Nutzung des kNN-Algorithmus aus Sektion 4.1 hierzu ist allerdings den Beispiel-basierten Ansätzen zuzuordnen, nachdem diese ML-Methode als Instanz-basierter Algorithmus gilt.

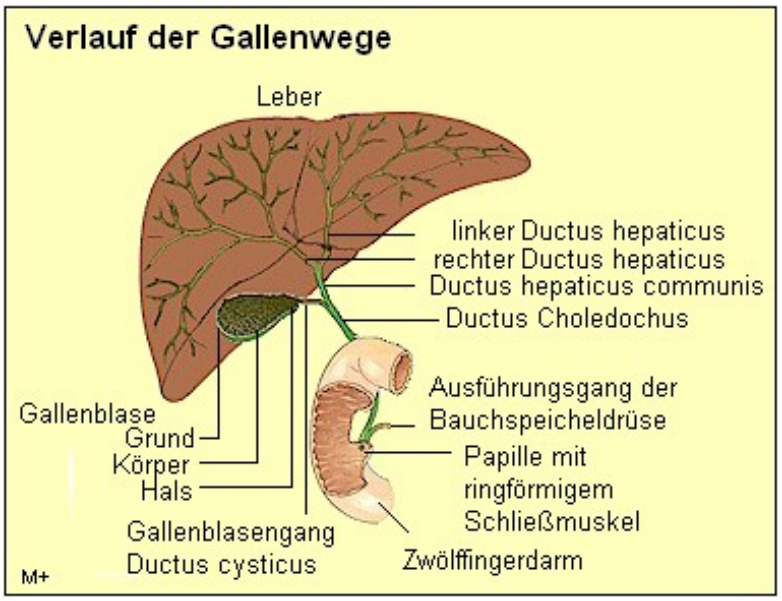
Weiter soll innerhalb dieser Arbeit nicht auf die einzelnen, lediglich kurz umrissenen Methodiken eingegangen werden, sondern stattdessen auf Molnar (2018) und die von diesem genutzten Quellen verwiesen sein.

8.5 Ergänzungen zu Kap. 6: Durchführung einer Benchmark-Studie basierend auf einem Datensatz zu akuter Cholangitis

8.5.1 Cholangitis - Aufbau und Funktion der Gallenwege

Anatomie der Gallenwege

Die s.g. Gallenwege beschreiben das kanalikuläre System, über welches die in der Leber produzierte Galle hin zur Gallenblase (Vesica fellea) bzw. zum Zwölffingerdarm (Duodenum) transportiert wird. Die eingehende Beschreibung der Gallenwege in dieser Sektion orientiert sich im Wesentlichen an AMBOSS - Fachwissen für Mediziner (2018). Bei den Gallenwegen unterscheidet man grundsätzlich zwei Abschnitte: den intrahepatischen (innerhalb der Leber gelegenen) und den extrahepatischen (außerhalb der Leber gelegenen) Teil.
Innerhalb der Leber beginnen die Gallenwege mit den s.g. Gallenkanälchen (Canaliculi biliferi), geht über in kurze Verbindungsstücke - die Hering-Kanälchen - und schließlich in die Gallengänge Ductuli biliferi interlobulares. Letztlich wird die Gallenflüssigkeit also über kleinere Kanälchen hin zu größer werdenden Kanälen zusammengeführt. Die intrahepatischen Gallenwege enden in den zwei Gallengängen Ductus hepaticus dexter (rechts) und sinister (links), welche zur Leberpforte führen.



Quelle: Wehner (o. J.)

Abb. 8.8: Aufbau der Gallenwege

Sobald die Gallenflüssigkeit die Leberpforte verlässt, betrachtet man die extrahepatischen Gallenwege, die sich von der Leberpforte bis zum Zwölffingerdarm (Duodenum) erstrecken (siehe 8.8). Zunächst bringt die Vereinigung des Ductus hepaticus dexter und sinister den

Ductus hepaticus communis hervor, welcher die Leber über die Leberpforte verlässt. Schließlich spaltet dieser Gallengang sich in zwei Gallenwege auf: Ductus cysticus und Ductus choledochus. Ersterer stellt eine Abzweigung vom Ductus hepaticus communis dar und leitet die Gallenflüssigkeit gegebenenfalls in die Gallenblase. Entsprechend ist diese für die Speicherung (Fassungsvermögen: ca. 70 ml), Eindickung und - je nach Bedarf - für die Freisetzung der Galle zuständig. Nach Abgang des Ductus cysticus setzt sich der Ductus hepaticus communis in Form des Ductus choledochus fort und mündet schließlich im Zwölffingerdarm.

Transport der Galle

Die Gallenflüssigkeit wird durch die metabolisch aktiven Leberzellen (Hepatozyten) in einer Menge von etwa 700 ml pro Tag erzeugt. Die Galle wird insbesondere im Zwölffingerdarm benötigt, wo ihr die Neutralisierung des stark sauren, aus dem Magen kommenden Verdauungsbreis zukommt. Zudem werden mithilfe der in der Gallenflüssigkeit enthaltenen Gallensäure nicht-wasserlösliche Fettbestandteile, wie etwa Fettsäuren und Steroide, durch Mizellenbildung innerhalb des Verdauungstraktes löslich und entsprechend auch resorbierbar gemacht. Eine weitere wichtige Aufgabe spielt die Gallensäure im Hinblick auf die Ausscheidung nichtwasserlöslicher Produkte wie Bilirubin, Cholesterin und auch von Medikamenten. Angemerkt sei an dieser Stelle, dass die Gallensäuren zum großen Teil rückresorbiert werden und über den s.g. enterohepatischen Kreislauf wiederum in die Leber gelangen.
Von der Leber gebildete Galle fließt zunächst über den Ductus hepaticus und den Ductus cysticus in die Gallenblase, wo sie eingedickt, wobei diese auf 10% ihres Volumens reduziert wird, und gespeichert wird. Mit Beginn des Verdauungsvorganges kontrahiert die Gallenblase und die konzentrierte Galle wird über den Ductus choledochus in das Duodenum abgegeben.

8.5.2 Imputation fehlender Werte

Fehlende Werte sind innerhalb dieser Arbeit insbesondere deshalb als kritisch zu betrachten, nachdem einige ML-Methoden nicht anwendbar sind, insofern diese „Schwachstellen“ nicht auf irgendeine Weise beseitigt werden. Allgemein sieht man sich wohl bei den meisten Analysen von Datensätzen mit dem Problem fehlender Werte konfrontiert. Ein mögliches Vorgehen wäre alle Beobachtungen, die ein oder mehrere fehlende Einträge aufweisen, von weiteren Analysen auszuschließen, was als „Complete-Case Analysis“ bezeichnet wird. Dies hat jedoch zur Folge, dass zum einen ein starker Informationsverlust auftreten kann - insbesondere dann, wenn viele fehlende Werte innerhalb der Daten vorhanden sind - und zum anderen können die folgenden Analysen stark verfälscht bzw. verzerrt sein, falls die Systematik der fehlenden Werte in Abhängigkeit von der Ausprägung des unvollständig erhobenen Features steht.
In diesem Zuge etablierten sich s.g. Imputationsverfahren, mittels derer Datensätze auf plausible Weise vervollständigt werden sollen. Prinzipiell steht eine breite Palette möglicher Imputationstechniken zur Verfügung - beispielhaft sei hier auf Gelman und Hill (2006) verwiesen-,

die insbesondere auch in Anbetracht der vorliegenden Datensituation gewählt werden sollten. So ist mitunter auch der Typus der unvollständigen Einflussvariable zu entscheidend über mögliche Imputationsmethoden. Des Weiteren verlangen viele dieser Methoden die Festsetzung von Tuning-Parametern, wie etwa die Imputation kontinuierlicher Größen über den kNN-Algorithmus (siehe 4.1) die Eingabe des Hyperparameters k - für die zu betrachtenden nächsten Nachbarn - erfordert.
Für die im Rahmen dieser Arbeit verwendeten Daten wurde eine Imputation der fehlenden Werte mittels der s.g. MissForest-Technik von Stekhoven und Bühlmann (2011) gewählt. Die Motivation hinter diesem Imputationsverfahren stellt der Umgang mit jeglichem Typus der Input-Daten, sowie möglichst wenige zu treffende Annahmen über die Struktur dieser Einflussgrößen dar. Diese Aspekte veranlassten Stekhoven und Bühlmann (2011) schließlich zur Nutzung von RF als nicht-parametrische Methode zur Bewältigung dieser Aufgabe. Dabei bedient man sich eines iterativen Imputationsschemas, bei welchem im ersten Schritt ein RF auf Basis der beobachteten Daten trainiert wird, gefolgt von der Prognose fehlender Werte im zweiten Schritt und iterativem Vorgehen.
Nachdem fehlende Werte bei dem im Rahmen dieser Arbeit analysierten Daten nur mit geringer Häufigkeit auftreten und zudem die Imputation keinen zentralen Aspekt hier darstellen soll, wird auf ausweitende Erklärungen an dieser Stelle verzichtet und stattdessen auf Stekhoven und Bühlmann (2011) für die Implementierung des Algorithmus, sowie die kompetitive Gegenüberstellung dieses Imputationsverfahrens zu anderen verwiesen.

8.5.3 Deskriptive Analysen

8.5.4 Beschreibung der Studie und der Daten - Univariate Analyse von Prädiktoren der Sterblichkeit

Die nachfolgende Tab. 8.3 wurde den Analysen von Schneider et al. (2016) entnommen.

Binäre und dichotomisierte Risiko-Prädiktoren	**Inzidenz der Sterblichkeit**	**Odds Ratio [95%-KI], p-Wert**
Alter ≥ 65	4.0%(23/579)	$1.1[0.5; 2.3], p = 0.736$
Alter < 65	3.5%(214/402)	
Alter ≥ 75	4.3%(13/303)	$1.2[0.6; 2.5], p = 0.588$
Alter < 75	3.5%(24/678)	
männlich	2.9%(16/549)	$1.7[0.8; 3.4], p = 0.129$
weiblich	4.9%(21/432)	
Fieber (Temp. ≥ 39 °C)	3.8%(2/52)	$1.0[0.2; 4.4], p = 1.000$

Fieber (Temp. ≥ 39 °C)	3.8%(34/905)	
vorhergehende Episode von Cholangitis	3.8%(17/446)	$1.0[0.5; 1.9], p = 1.000$
keine vorhergehende Episode von Cholangitis	3.7%(20/535)	
Indwelling Stent (Stent-Therapie)	3.4%(16/466)	$0.8[0.4; 1.7], p = 0.619$
Ohne Indwelling Stent	4.1%(21/515)	
PTC als Intervention	6.1%(19/310)	$2.4[1.2; 4.6], p = 0.011$
Ohne PTC als Intervention	2.7%(18/671)	
Geistige Verwirrung	25.9%(29/112)	$37.6[16.6; 85.0], p < 0.001$
Keine geistige Verwirrung	0.9%(8/869)	
Vorhergehende Intervention (ERC/PTC)	5.0%(19/383)	$0.6[0.3; 1.2], p = 0.125$
Ohne vorhergehende Intervention	3.0%(18/598)	
Katecholamine erfordernde Hypotension	18.3%(11/60)	$7.7[3.6; 16.6], p < 0.001$
Keine Katecholamine	2.8%(26/921)	
Unzureichende Drainage	8.3%(20/240)	$4.5[2.2; 9.1], p < 0.001$
Angemessene Drainage	2.0%(14/706)	
Organversagen m. allen Parametern (TG13)	15.7%(34/217)	$47.1[14.3; 155.2], p < 0.001$
Kein Organversagen	0.4%(3/764)	
Leukozyten > 15 G/l	6.9%(12/174)	$2.3[1.1; 4.8], p = 0.026$
Leukozyten ≤ 15 G/l	3.1%(25/807)	
Bilirubin ≥ 5 mg/l	6.4%(29/452)	$4.5[2.0; 9.9], p < 0.001$
Bilirubin < 5 mg/l	1.5%(8/529)	
Aspartat-Aminotransferase (AST) > 70	5.3%(19; 357)	$2.1[1.0 - 4.3], p = 0.129$
AST ≤ 70	2.6%(14; 532)	
Alanin-Aminotransferase (ALT) > 70	3.0%(14/462)	$0.7[0.3; 1.3], p = 0.242$
ALT ≤ 70	4.6%(23; 502)	
Quick's Wert $< 50\%$	16.7%(9; 54)	$6.4[2.8 - 14.5], p < 0.001$
Quick's Wert $\geq 50\%$	3.0%(28; 927)	
Serumkreatinin > 2 mg/dl[3]	13.2%(5; 38)	$4.3[1.5 - 11.8], p = 0.012$
Serumkreatinin ≤ 2 mg/dl[3]	3.4%(32; 943)	

Anzahl Blutplättchen $< 100\ 000/\text{mm}^3$	$9.8\%(5;51)$	$3.0[1.1-8.2], p = 0.038$
Anzahl Blutplättchen $\geq 100\ 000/\text{mm}^3$	$3.4\%(32;930)$	
Bakteriämie	$7.7\%(14/82)$	$2.8[1.4;5.6], p = 0.004$
Keine Bakteriämie	$2.9\%(23/799)$	
Bakteriämie	$7.7\%(14/82)$	$2.8[1.4;5.6], p = 0.004$
Maligne (bösartige) Genese	$1.9\%(9/472)$	$3.0[1.3;6.5], p = 0.004$
Nicht-maligne Genese	$5.5\%(28/509)$	

Tab. 8.3: Studie zu akuter Cholangitis: Univariate Analyse der Prädiktoren für Sterblichkeit, übernommen von Schneider et al. (2016)

8.5.5 Einstellungen bei Default-Setting, sowie Hyperparameter-Tuning und SMOTE

Lerner	Parameter	Bezeichnung	Typ	Default	Suchraum
SMOTE	sw.rate	Faktor für Upsampling der Minority-Klasse	Numerisch	1	$[1, 20]$
	sw.nn	Anz. nächster Nachbarn	Numerisch	5	$[2, 9]$
kNN	k	Anz. nächster Nachbarn	Integer	3	[1,20]
	kernel	Kern	Character	*'optimal'*	
cTree	testtype	Berechnung Verteilung von Teststatistik	Character	*'Bonferroni'*	
	mincriterion	Wert Teststat., der überschritten werden muss für Split ($1 - p$-Wert)	Numerisch	0.95	$[0.7, 1]$
	maxdepth	Max. Tiefe des Baumes	Integer	30	$[3, 30]$
	minbucket	Min. Anz. an Beob. in terminalen Knoten	Numerisch	7	$[5, 50]$
	minsplit	Min. Anz. an Beob. für Durchführung eines Splits	Numerisch	20	$[5, 50]$
cForest	testtype	Berechnung Verteilung von Teststatistik	Character	*'Bonferroni'*	
	mincriterion	Wert Teststat., der überschritten werden muss	Numerisch	0.95	

		für Split ($1 - p$-Wert)			
	replace	Ziehen der Beob. mit oder ohne Zurücklegen	Logisch	*TRUE*	
	ntree	Anz. anzupassender Bäume (cTrees)	Numerisch	5	
	mtry	Anz. an Input-Variablen, die pro Knoten zufällig als Kandidaten gezogen	Numerisch	5	
SVM	cost	Bestrafungsparameter	Numerisch	0	[-15, 15]
	gamma	Einflussreichweite eines Trainingsbeispiels	Numerisch	$log2(1/p)$	[-15, 15]
	kernel	Verwendter Kern-Typ	Character	*'radial'*	
glmboost	family	Spezifizierung Verlustfkt.	Objekt	*Binomial (link='logit')*	
	mstop	Initiale Anz. an Boosting-Iterationen	Numerisch	3.32	[-3.32, 6.64]
	nu	Länge der Schritte	Numerisch	0.1	$[0, 1]$
glmnet	family	Art des Response	Character	*binomial*	
alpha	Mixing Parameter	Numerisch $(1 - \alpha)/2 \cdot \|\|\beta\|\|_2^2 + \alpha\|\|\beta\|\|_1$ $\alpha = 0$ Ridge, $\alpha = 1$ LASSO	1	$[0, 1]$	
	lambda	Bestrafungsparameter	Numerisch	0	$[-10, 3]$

Tab. 8.4: Default-Settings und Einstellungen des Hyperparameter-Tunings: SMOTE-, sowie Klassifikationsmethoden mit Parameter-Konfiguration. Ohne Suchraum $\hat{=}$ kein Tuning.

8.5.6 Auswertung der Ergebnisse aus der Benchmark-Studie

	Name and Art of Task	auc	pAUC	Learner	Tuning	SMOTE
1	data, classification	0.804	0.081	svm	untuned	unsmoted
2	data, classification	0.899	0.140	glmboost	untuned	unsmoted
3	data, classification	0.905	0.149	glmnet	untuned	unsmoted
4	data, classification	0.753	0.072	kknn	untuned	unsmoted
5	data, classification	0.854	0.107	ctree	untuned	unsmoted
6	data, classification	0.908	0.151	cforest	untuned	unsmoted
7	data, classification	0.841	0.118	svm	untuned	smoted
8	data, classification	0.902	0.144	glmboost	untuned	smoted
9	data, classification	0.883	0.138	glmnet	untuned	smoted
10	data, classification	0.769	0.072	kknn	untuned	smoted
11	data, classification	0.728	0.057	ctree	untuned	smoted
12	data, classification	0.899	0.137	cforest	untuned	smoted
13	data, classification	0.783	0.086	svm	tuned	unsmoted
14	data, classification	0.879	0.130	glmboost	tuned	unsmoted
15	data, classification	0.918	0.159	glmnet	tuned	unsmoted
16	data, classification	0.809	0.087	kknn	tuned	unsmoted
17	data, classification	0.863	0.105	ctree	tuned	unsmoted
18	data, classification	0.894	0.142	svm	tuned	smoted
19	data, classification	0.900	0.142	glmboost	tuned	smoted
20	data, classification	0.910	0.148	glmnet	tuned	smoted
21	data, classification	0.846	0.112	kknn	tuned	smoted
22	data, classification	0.739	0.060	ctree	tuned	smoted

Abb. 8.10: Tabelle und Zusammenfassung der (aggregierten) Ergebnisse aus Benchmark-Studie zu akuter Cholangitis.

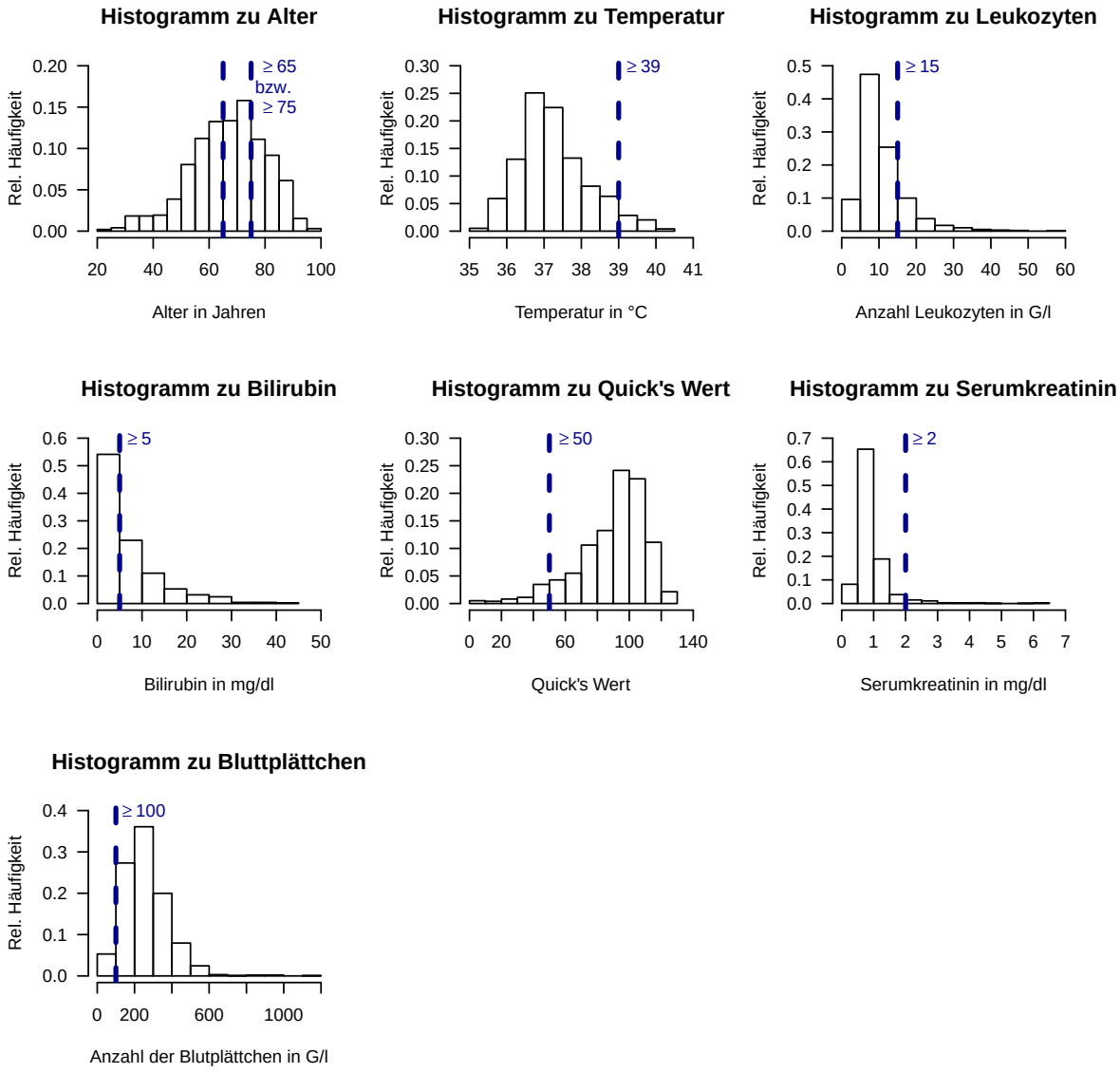


Abb. 8.9: Histogramme zu numerischen Prädiktoren; Blaue Linie zur Kenntlichmachung desjenigen Punktes, der die zusätzlichen Dichotomisierung der Variablen ausmacht

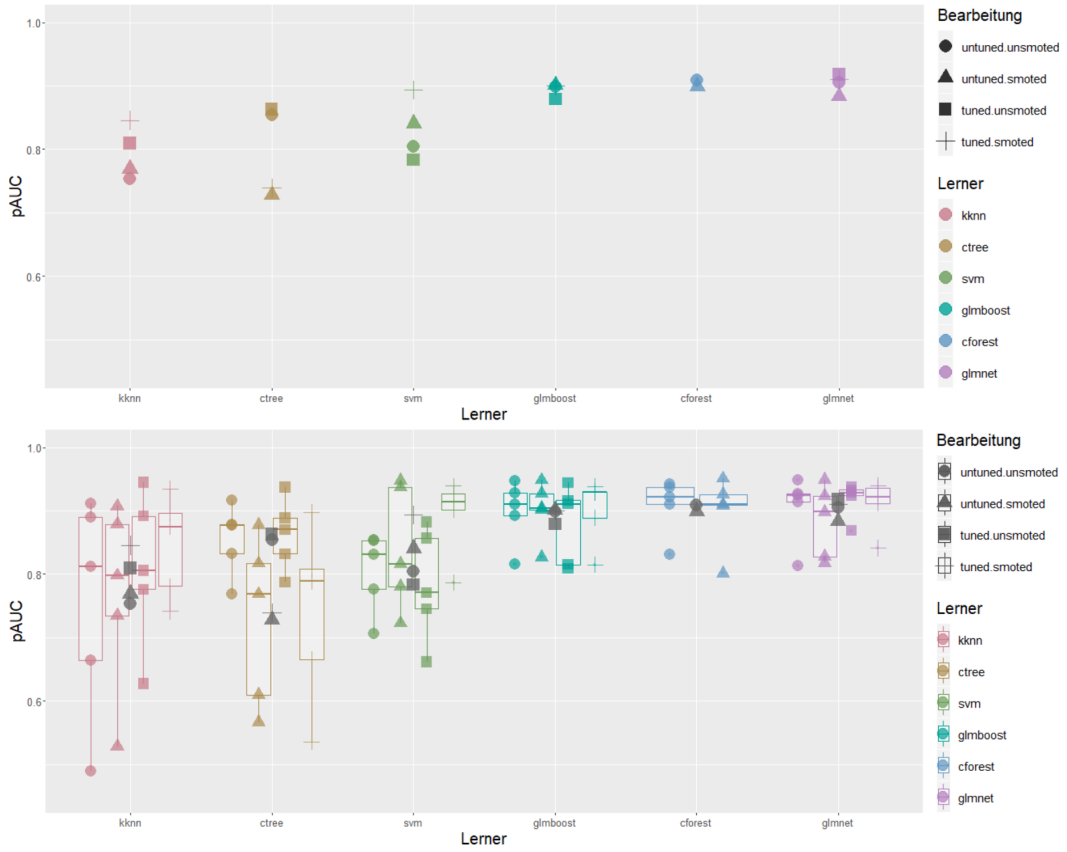


Abb. 8.11: Performance (AUC) der 22 ausgewählten Lerner in Anwendung auf Daten zu akuter Cholangitis: gruppiert nach ML-Methodik und vorgenommenem Tuning/SMOTE.
Oben: aggregierte BMRs; unten: unaggregierte BMRs.

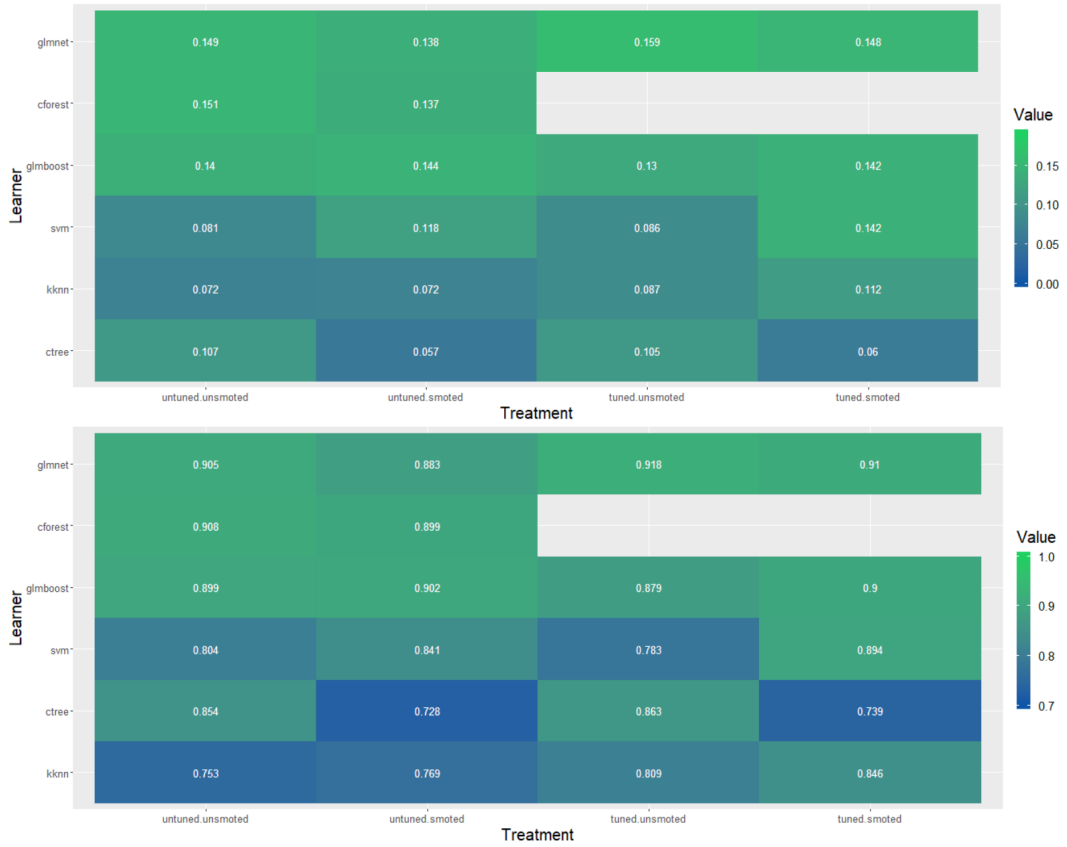


Abb. 8.12: Heatmap zur Performance der 22 ausgewählten Lerner in Anwendung auf Daten zu akuter Cholangitis: gruppiert nach ML-Methodik und vorgenommenem Tuning/SMOTE.
Oben: Beurteilung über pAUC; unten: Beurteilung über AUC.

	Name and Art of Task	auc	pAUC	Learner	Tuning	SMOTE	Pareto Level
1	data, classification	0.918	0.159	glmnet	tuned	unsmoted	1
2	data, classification	0.908	0.151	cforest	untuned	unsmoted	2
3	data, classification	0.910	0.148	glmnet	tuned	smoted	2
4	data, classification	0.905	0.149	glmnet	untuned	unsmoted	3
5	data, classification	0.902	0.144	glmboost	untuned	smoted	4
6	data, classification	0.894	0.142	svm	tuned	smoted	5
7	data, classification	0.900	0.142	glmboost	tuned	smoted	5
8	data, classification	0.899	0.140	glmboost	untuned	unsmoted	6
9	data, classification	0.899	0.137	cforest	untuned	smoted	6
10	data, classification	0.883	0.138	glmnet	untuned	smoted	7
11	data, classification	0.879	0.130	glmboost	tuned	unsmoted	8
12	data, classification	0.841	0.118	svm	untuned	smoted	9
13	data, classification	0.846	0.112	kknn	tuned	smoted	9
14	data, classification	0.854	0.107	ctree	untuned	unsmoted	9
15	data, classification	0.863	0.105	ctree	tuned	unsmoted	9
16	data, classification	0.809	0.087	kknn	tuned	unsmoted	10

Abb. 8.13: Tabelle zu Skyline-Levels mit Berücksichtigung der Gütemaße AUC und pAUC: Pareto-Dominanz des glmnet's mit Tuning gegenüber den übrigen Lernern in Anwendung auf Daten zu akuter Cholangitis.

shinyBMR: Analysis and Interpretation of Benchmark Studies (with mlr and iml)

This package shall provide an interactive framework for the analysis of benchmark studies, which have been carried out based on the tools/functionalities provided in **mlr**.

As these benchmark studies tend to come up with a huge bandwidth of information **shinyBMR** shall mainly offer a opportunity for users to get an easily accessible, but highly informative overview of their enforced benchmark study based on **mlr**. This is basically done by the use of interactive, graphical tools as well as structured and detailed summaries.

Another point coming up when performing such analyses is the question about explainability and interperatbility. Many of the machine learning (ML) methods are so called blackbox methods, meaning there is not such an easy way - like for example in logistic regression - for calculating the output when the input/data and the focused model are provided. In this sense most of the ML methods return your results without eyplaining the way they were able to come up with this output. But there is a way out of this situation: one can make use of so called model-agnostic methods provided in the **iml** package to offer the user a basis for understanding the causes of decisions made by the machine.

This ShinyApp **shinyBMR** focuses on the before mentioned aspects and its usage is explained in the latter documentation, that can be structured as follows:

I. Analysis of Benchmark Results (BMR)

1. Import of the Benchmark Results (from **mlr**)
2. Overview of the Information contained in the Benchmark Object
3. Graphical Analysis of the competing Methods included in the Benchmark Study

II. Access to the Interpretation of Blackbox ML-Methods (IML)

1. Import and Overview of the focused Data Set
2. Import and Overview of the (Blackbox) Model
3. Access to Interpretability provided by different tools of the **iml** package

Keep in mind that the shown pictures/graphics not necessarily belong to the same data respectively to the same BMR object as different use cases shall be visualized.

Installation and starting shinyBMR

You can install this package from github with help of the **devtools** package:

```
devtools::install_github("tess-st/shinyBMR")
```

Starting the ShinyApp:

```
runshinyBMR()
```

If `rJava` fails to load, **this link** might be helpful.

Welcome Page

When opening the app a starting page *Welcome* is popping up. Beside some introductional text there are also some helpful links listed, which are useful in case of benchmark studies, in particular, when accomplished with the **mlr** package.

When having a closer look at the interface one should recognize the construction of the app organized in the tabs selectable right under the headline. The following descriptions contain information about the actual usage of these tabs or rather their functionalities.

I. Analysis of Benchmark Results (BMR)

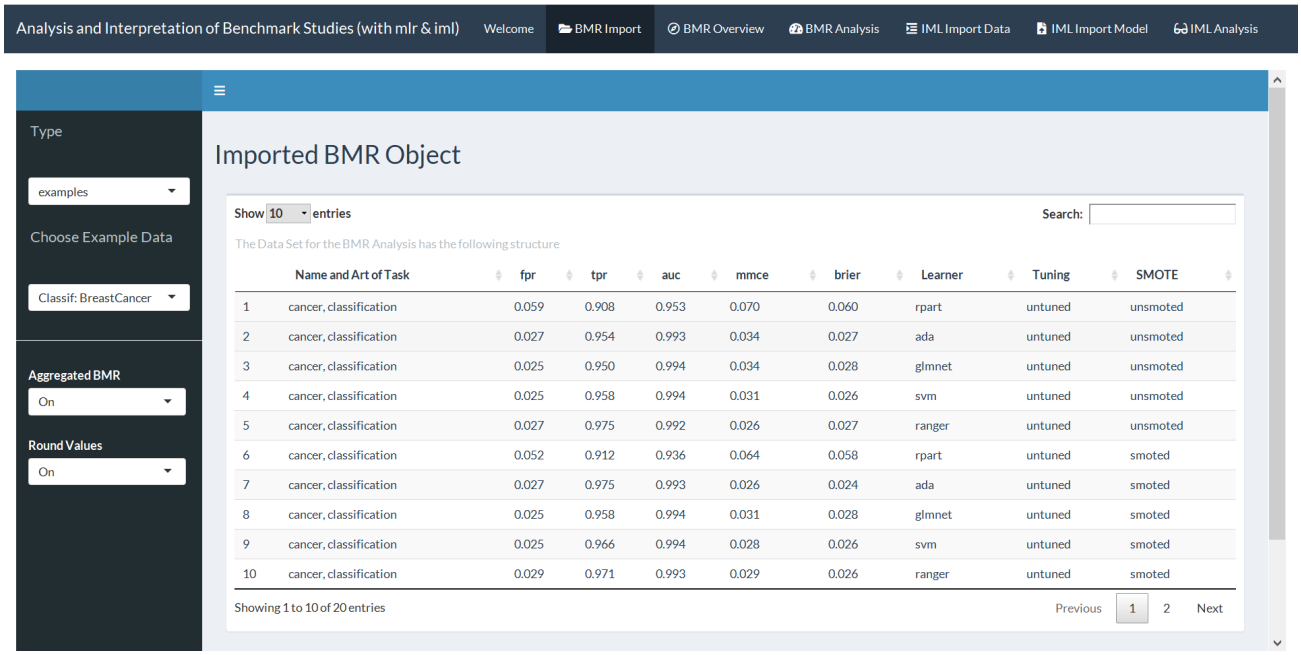
1. Import of the Benchmark Results

As now the concrete analysis of benchmark studies will be focused on, you first need to upload your benchmark object. By selecting the ***BMR Import*** tab the preface shows an exemplary data frame. There are two example data sets provided, which can be used to obtain the functionalities of **shinyBMR** supplied for the analysis of benchmark studies. One can choose between a classfication and a regression example. Former one is carried out at the **Breast Cancer** data set of the **mlbench** package. The regression example based on the **Longley's Economic** data set is available in the **datasets** package. The conducted benchmark experiments based on the two example data sets can be found in the downloaded package when navigating to "*...\shinyBMR\inst\shinyBMR\examples\R-Code*".

It is now your turn to upload the BMR object, which you have saved as .RDS file by clicking on the *Type* selection button in the sidebar and choosing *RDS*. When doing so, a new sidebar tab is opening telling you to *Choose RDS File*, which is containing your benchmark experiment. This BMR object has to be set up via the **mlr** package or - more accurately - via the follwoing function:

```
benchmark(learners, tasks, resamplings, measures, keep.pred = TRUE,
  models = TRUE, show.info = getMlrOption("show.info"))
```

With the help of this function different learning methods can be applied to one or several data sets with the aim to compare and rank the algorithms in respect of one or more performance measures. At the moment **shinyBMR** is only set up for the analysis of benchmark experiments performed on a single data set as this is the main focus of most clinical studies involving machine learning methods. Since the actual usage of the benchmark function requires a few more steps - which need to be done on basis of **mlr** - it might be useful to go through the **mlr help page**.



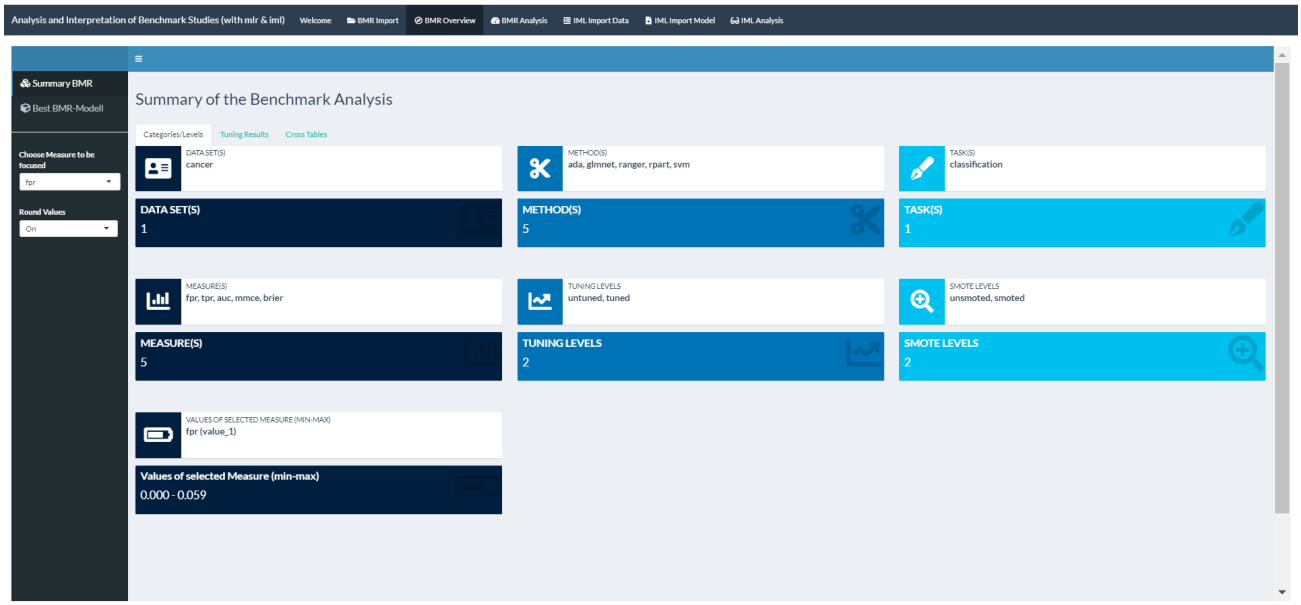
Finally, having uploaded your BMR object, **shinyBMR** will show a data table containing all relevant information about your study: the first row contains *Name and Art of Task*. The following row(s) holds the *Value of Measure*, which the analysis is based on. Of course all of the measures you have used in your study will be displayed with their corresponding values for the specific learner in the table. Each *Learner*, who is taking part in the competition, is listed. The next row contains information about the *Tuning* status and - in case of a classification task - is followed by a row showing the *SMOTE* status. Other developments of the learners then tuning and SMOTE have not been implemented in **shinyBMR** yet, as these wrappers seem to be the main focus when it comes to clinical studies using methods of machine learning.

By default you only see the aggregated performance of each learner. Clicking *Aggregated BMR* offers the opportunity to also show the unaggregated performance containing the values of each learner per iteration. Additionally the values of the measure(s) are rounded by default for a better overview. This can be undone by making the corresponding selection at *Round Values*.

2.Overview of the Information contained in the Benchmark Object

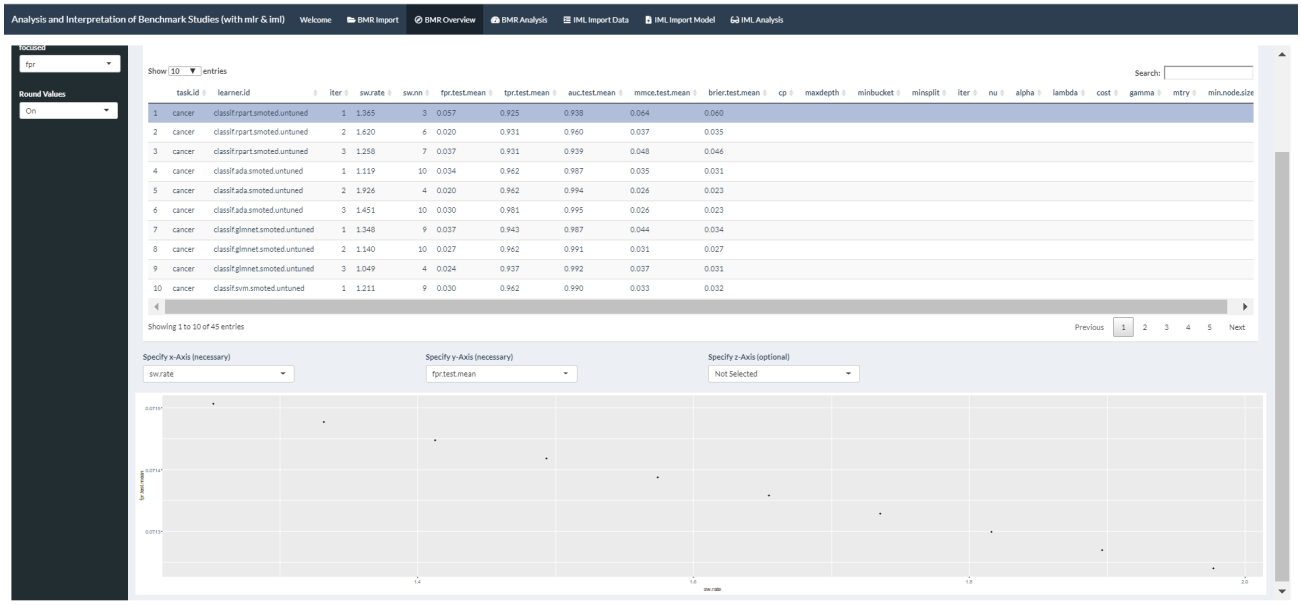
Summary BMR

As the BMRs tend to become quite huge objects this tab ***BMR Overview*** is made up to summarize the results in a clear and structured way. First of all, summarizing the results of the whole benchmark study, you have to navigate to the ***Summary BMR*** tab on the sidebar. Doing so, there is now the sub-tab ***Categories/Levels*** displayed, which basically breaks down the information of the imported BMR object to only the most relevant.



This is done by revealing the levels of the binary and factor variables of the data table, which were shown in the tab ***BMR Import*** before. In this way the user gets instructed aboute the Name of the *DATA SET(S)*, the competing learners or - to say it differently - the *METHOD(S)*, the art of the *TASK(S)*, the *MEASURE(S)* for evaluating the performance, the *TUNING LEVELS* (untuned, tuned), as well as - in case of classification - the *SMOTE LEVELS* (unsmoted, smoted). Each piece of information is provided in one box, that contains the name of the levels and a second box right under the first telling the user the number of categories. Additionally one can query the range of the *VALUES OF SELECTED MEASURE (MIN-MAX)*. If the benchmark analysis was performed on more than one measure you can specifically *Choose Measure to be focused* via the selection tab on the sidebar. Again rounded values are set per default, but this can be changed via the tab *Round Values*, which is also placed on the sidebar.

In case that the ML methods, which are competing in the benchmark study, have further wrapped hyperparameter tuning implemented, one can have a closer look at the *Tuning Results* by selecting the corresponding sub-tab. Doing so, a data table will open up, that shows the tuning results/parameters of all - in some way tuned - learners per iteration or in other words of the unaggregated data set. Besides the performance measure(s) you can easily examine the somehow optimal fitted parameters for a specific learner for the corresponding iteration. Also for this data table the shown values of the performance measures are rounded by default, whereby this setting can be changed with help of the input selection on the sidebar.

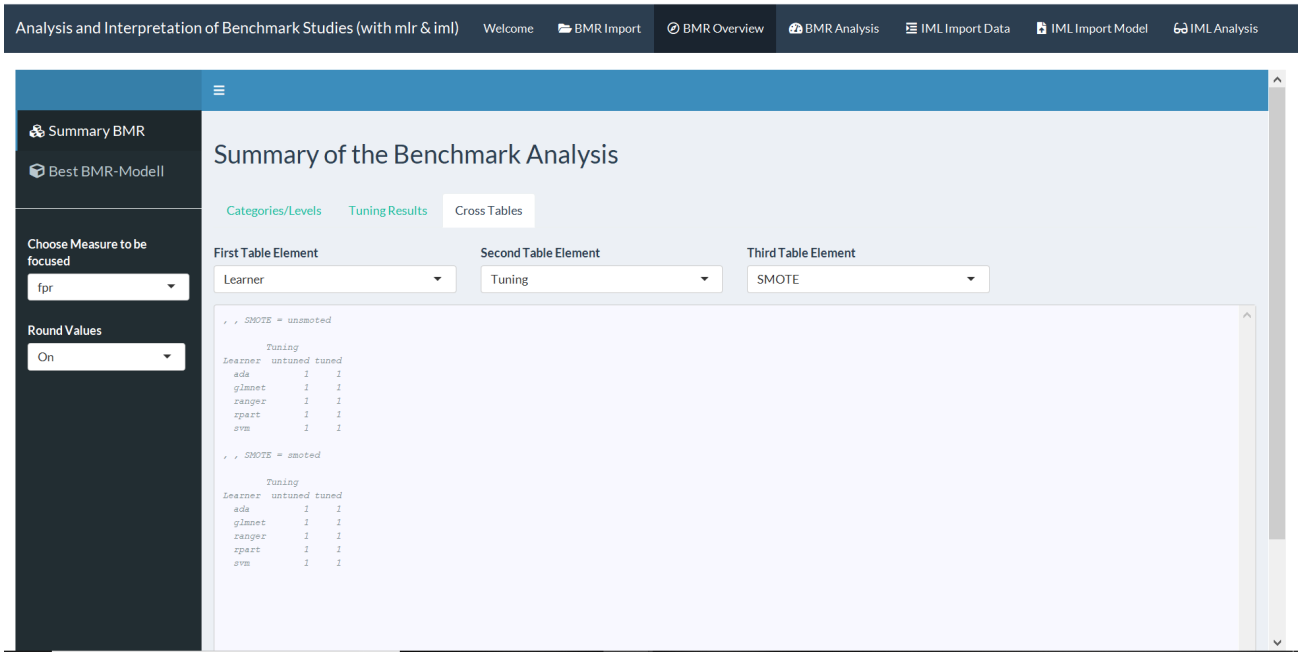


But this data table can be used even more interactively: choose one row respectively a method within a focused iteration to get three selection tabs in return. These can be used to create a plot for visualizing the hyperparameter validation path. Thia can be quite useful for determining the importance or effect of a particular hyperparameter on some performance measure and/or optimizer. This functionality is based on a function provided in the **mlr** package:

```
plotHyperParsEffect(hyperpars.effect.data, x = NULL, y = NULL,
  z = NULL, plot.type = "scatter", loess.smooth = FALSE,
  facet = NULL, global.only = TRUE, interpolate = NULL,
  show.experiments = FALSE, show.interpolated = FALSE,
  nested.agg = mean, partial.dep.learn = NULL)
```

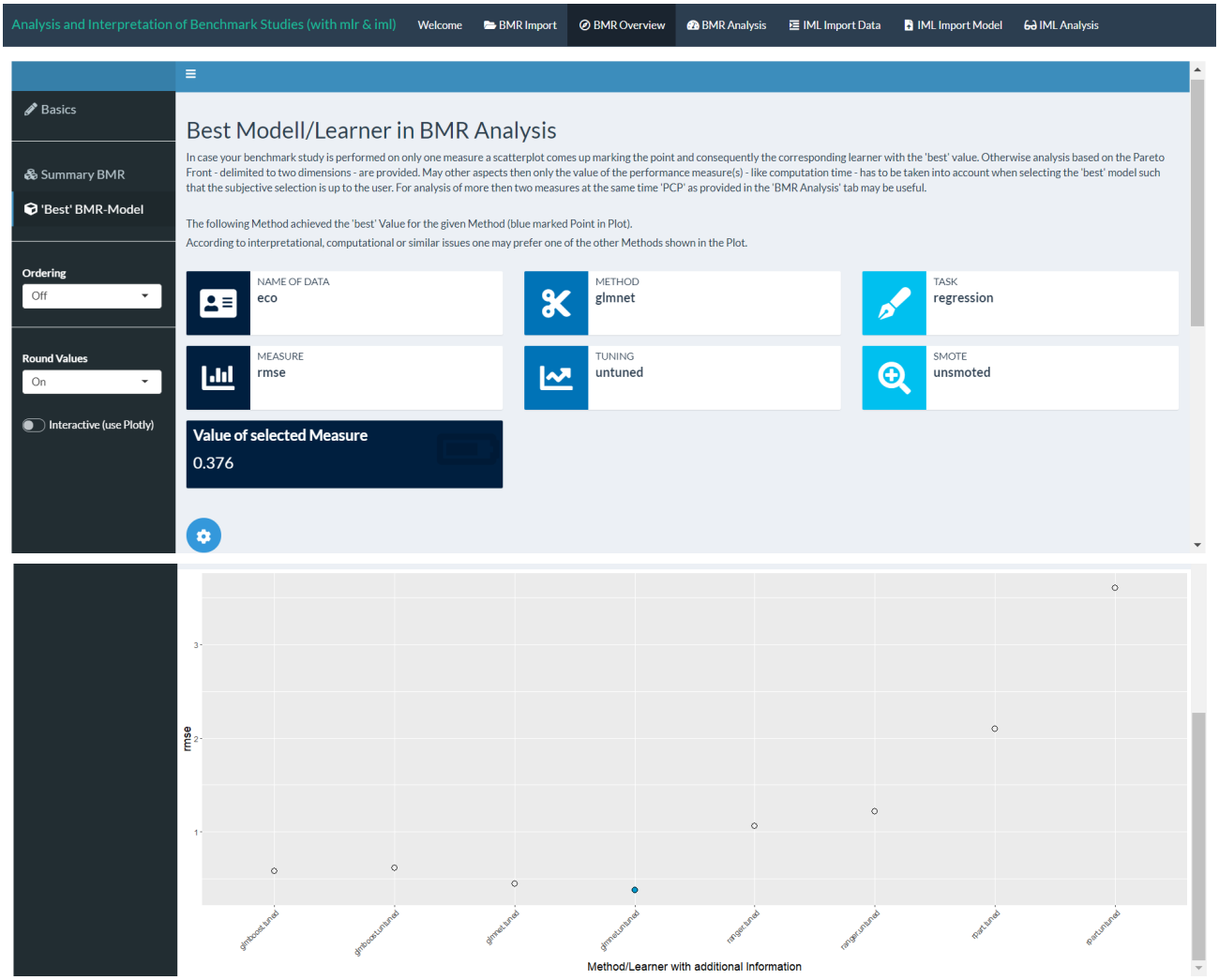
As the before listed function may already implements the first two input selection options, let the user now specify what should be plotted on the x- and on the y-axis in order to visualize the hyperparameter validation path. The third selection tab is optional for plotting an extra axis for a particular geom. This could be for the fill on a heatmap or color aesthetic for a line.

In case that there are still some questions left concerning the structure of the BMR object one can make use of the sub-tab *Cross Tables*. As this should be self-explanatory the subject will not be deepened further.



'Best' BMR-Model

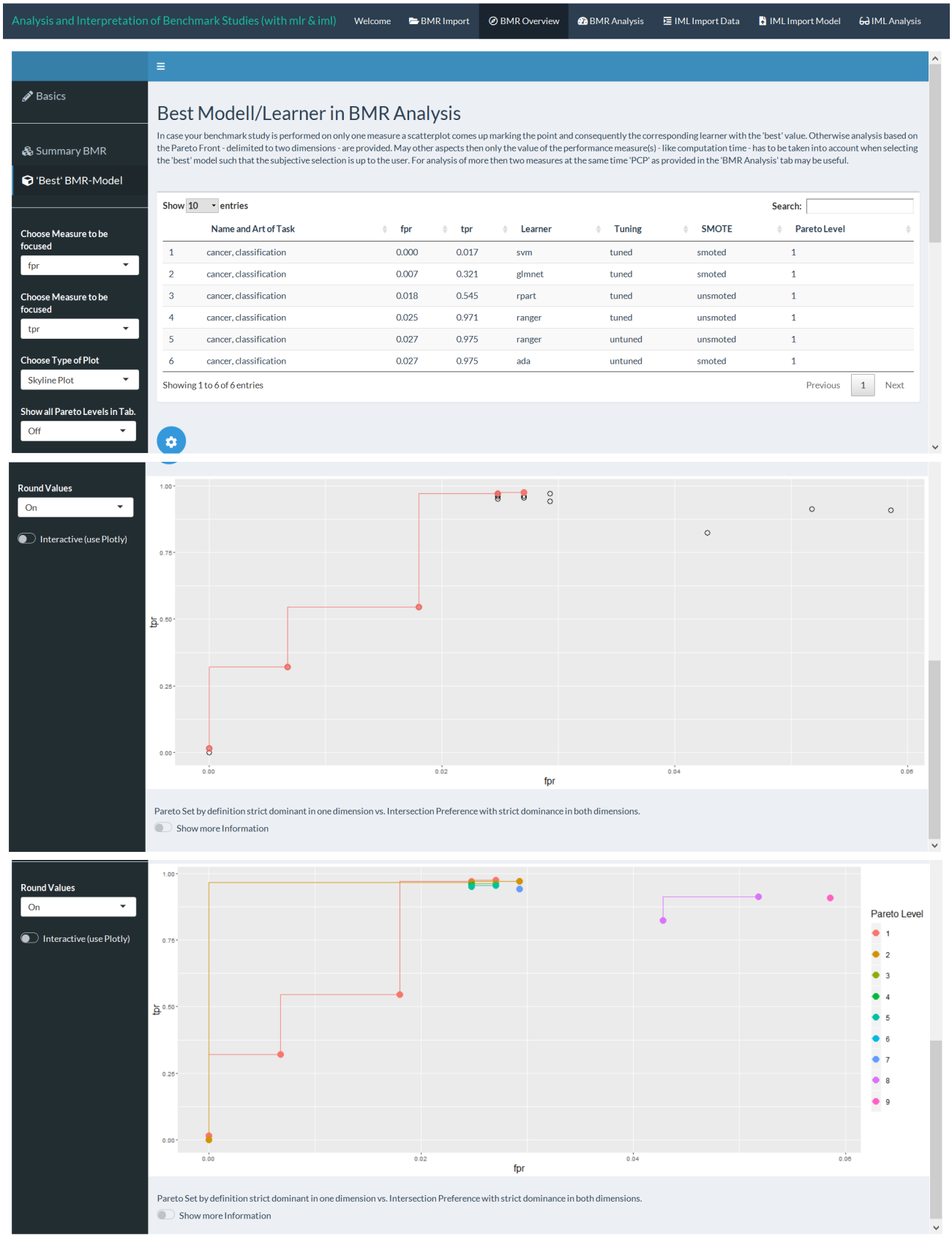
Having it made so far the question arises, which of the models may performs best on the given data set respectively within the BMR object. For this the tab 'Best Model' provides the opportunity to get a first impression of the ranking of the competing learners. In case the performance is assessed based on only one measure a scatterplot containing the single learners plotted against the specific measure is showing up. A blue point marks the 'best' value of the belonging machine learning model. May further aspects - e.g. computation time - have to be taken into account when selecting the (subjective) best learner.



Elsewise, in case the BMR data set contains more then one performance measure a different analysis page will be provided. Routines to select and visualize the maxima for a given strict partial order are shown. This especially includes the computation of the so called Pareto frontier, also known as (Top-k) Skyline operator. These functionalities can be found in the rPref package.

Definition of a Pareto frontier: 'A set of nondominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective. On the other hand a solution x* is referred to as dominated by another solution x if, and only if, x is equally good or better than x* with respect to all objectives.'

In this sense Pareto optimal BMR methods are shown in the table and marked by 'Level = 1'. By connecting these points included in the Pareto set the Pareto front gets displayed. One can also select the option 'Skyline Level Plot' displaying all levels of Pareto Fronts based on the top-k Selection. The definition of Pareto fronts itself requires strict dominance in only one dimension, while in the other one the measures could be equally good or better, leading to possibly overlapping front lines. Instead demanding strict dominance in both dimensions will solve this and probably will develop different compositions of the Pareto sets. This different selections can be made with the tab *Choose Type of Plot* on the sidebar.



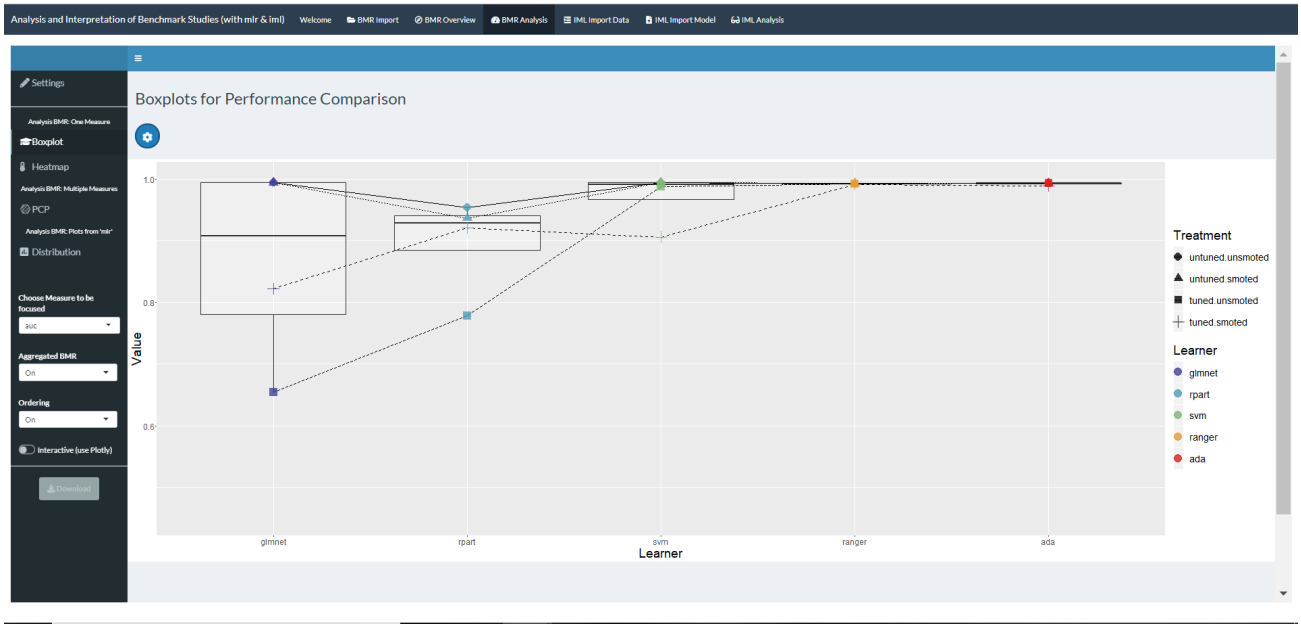
Analyses with help of the Pareto dominance for more than two dimensions are not yet provided. In case of more than two performance measures it may be helpful to make use of the so called Parallel Coordinate Plot (PCP) provided in the next tab.

3. Graphical Analysis of the competing Methods included in the Benachmark Study

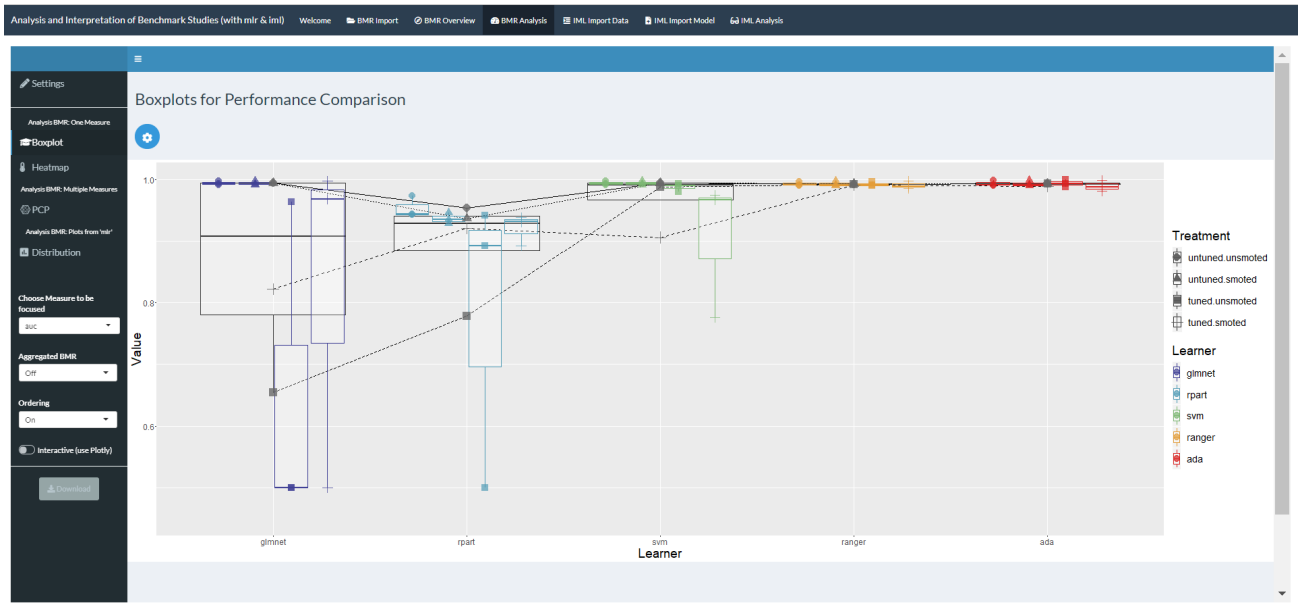
Having done the analyses so far it is now meaningful to display the results of the benchmark study in form of plots. Doing so you can go to the *BMR Analysis* tab. As a side note it should be mentioned that the settings under the second horizontal line on the sidebar are shown/hidden dependent on the selected type of plot.

Boxplot

For graphical analyses concerning one specific performance measure, which is included in your BMR object, you can make use of the *Boxplot* and *Heatmap* tab on the sidebar. The *Boxplot* tab provides - as the name already indicates - boxplots for a selected measure, which is depending on the user's input at *Choose Measure to be focused*. In order to get the specific plot in return one has to accept the chosen selections by clicking on the tooltip button - as the settings are highly reactive and have to be computed based on the selections. In case one has four different treatments (tuning vs. no tuning; SMOTE vs. no SMOTE) of one aggregated learner there will be boxplots displayed for the performance of each aggregated ML method. Elsewise, the performance per group of learners is only marked by a point on the specific scale of the selected measure. Notice that each group of learners receives an own colour as well as each treatment group (tuning vs. no tuning; SMOTE vs. no SMOTE) is labeled with a characteristic symbol.



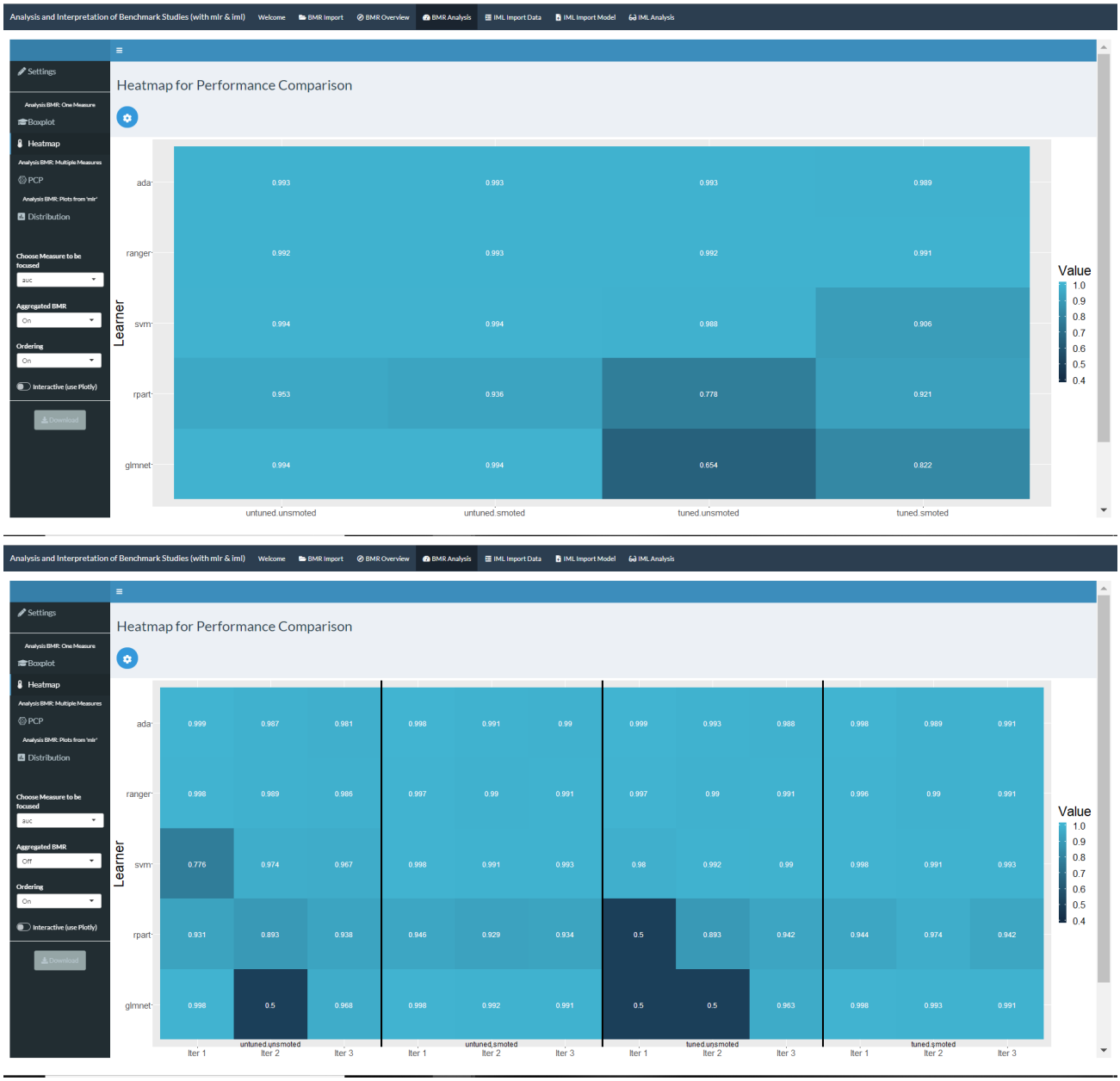
You can go deeper into detail by selecting *Aggregated BMR* as "Off" and thus adding the boxplots containing the values per iteration per treatment group of the respective learning method to the plot. The aggregated results remain in the plot as grey marked points connected by the specific lines marking the treatment group. These lines can be turned off with help of the tooltip. Many more selections can be set up as part of the tooltip, basically concerning the plotting - in particular, changing the colors and zooming in/out of the plot - and label options as well as the collection of the size.



Furthermore, the boxplots can be arrangedy according to their performance rank by *Ordering* the methods going from worst to best respectively left to right of the x-axis. The switch button *Interactive (use Plotly)* provides the created graphic as interactive plot, meaning the plot returns the information available by mouse click on the specific part/point of the plot. This functionality is gained by using the Plotly package. After having set up your final boxplot selctions the plot can now be downloaded via the *Download* button.

Heatmap

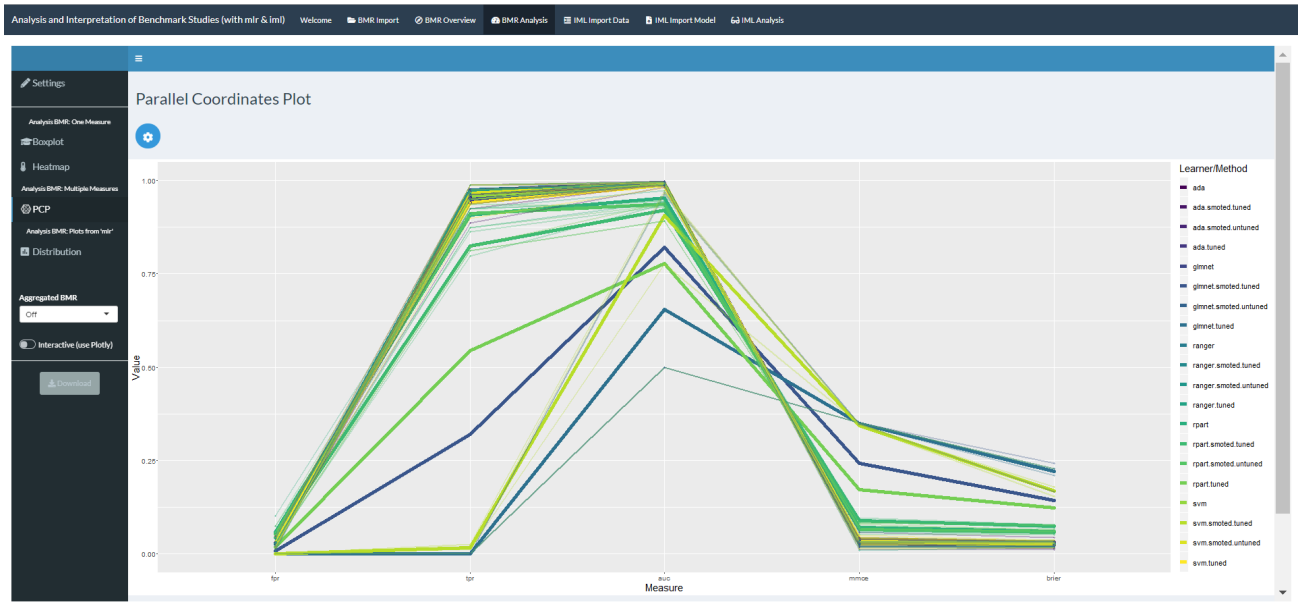
Another way of analyzing the BMR object on the basis of one performance measure is provided by the *Heatmap* tab displaying a heatmap with color range of the ML methods marking low and high performance values depending on *Choose Method to be focused*. Therefore, the single learners are grouped on the y-axis and plotted against the respective treatment on the x-axis, so that each cell created this way contains the performance value of this combination of method vs. treatment group. When switching *Aggregated BMR* to "Off" the treatment group and consequently the x-axis gets also split up in the single iterations or - to say it differently - the unaggregated results are shown. In turn *Ordering* rearranges the plot based on the means of aggregated performance per used ML method. Again also an interactive plot becomes available by turning the button *Interactive (use Plotly)* on. In the end, the user defined plot can be downloaded with help of the *Download* button.



As before, the heatmap is only plotted when the tooltip is chosen in order to confirm the settings. The tooltip provides options that can be particularly used to change colors, labels and the size of the heatmap.

PCP

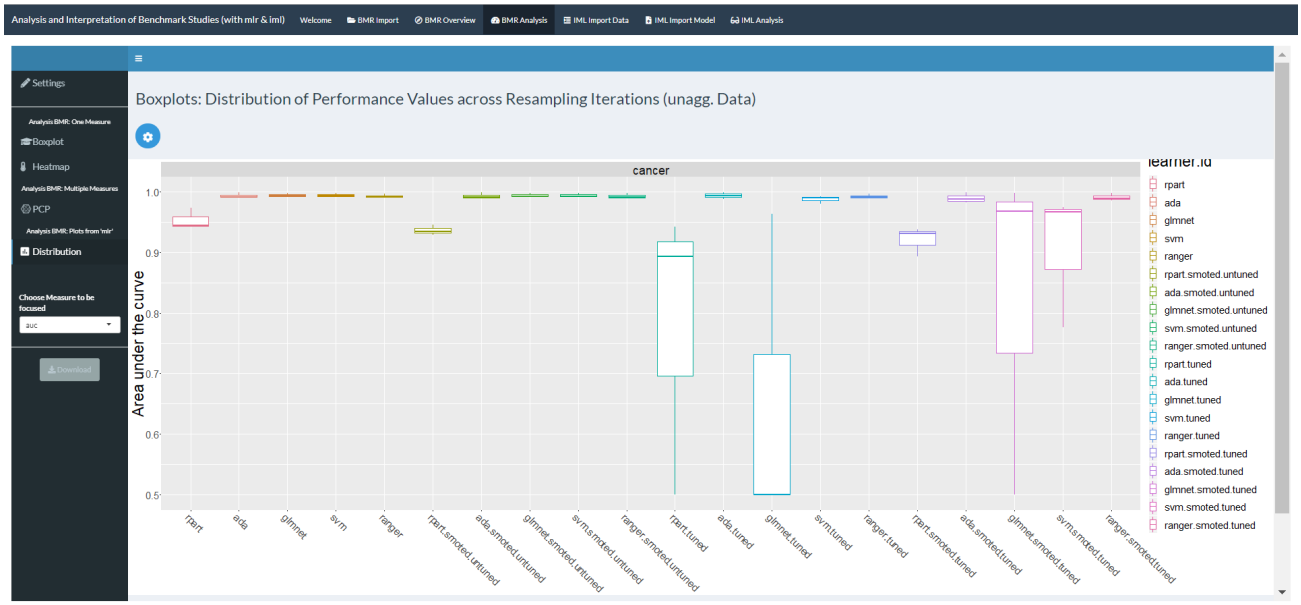
In case that you are generally interested in the learners performance to be compared over all measures included in your BMR object one can rely on the Parallel Coordinates Plot *PCP* for further analyses. This type of visualisation is especially useful for the plotting of multivariate, numerical data - in this case multiple performance measures. Doing so the user gets offered a overview of the relationship between the performance measures, which are achieved by the single ML methods. As reasonable consequence the PCP can only be used for analysis when at least two comparable measures are contained in the BMR object.



As for the plots mentioned before the options should be set by clicking on the tooltip button. Here some of the plotting options as well as the labels and the size can be controlled. Furthermore, the input of *Aggregated BMR* can be switched "Off" resulting in additional, shadowed lines marking the connection the performance estimates of the single measures grouped by learner per iteration. Also the PCP can be plotted *Interactive (use Plotly)* as well as downloaded via the *Download* button.

Distribution

Beside the so far presented graphical analysis methods, one can also make use of the plots integrated in the **mlr** package. Since most of these plots concentrate on analyzing benchmark studies, which are performed on multiple data sets - what has not been implemented in **shinyBMR** (yet) - and therefore focus on ranks, there is actually only one plot provided, that can be used in case of benchmark studies designed as expected for analyses with **shinyBMR**.



The graphic of **mlr** described plots the distribution of performance values across the resampling iterations and therefore of the unaggregated BMR object. The user can choose between box- or violin-plots (see tooltip)

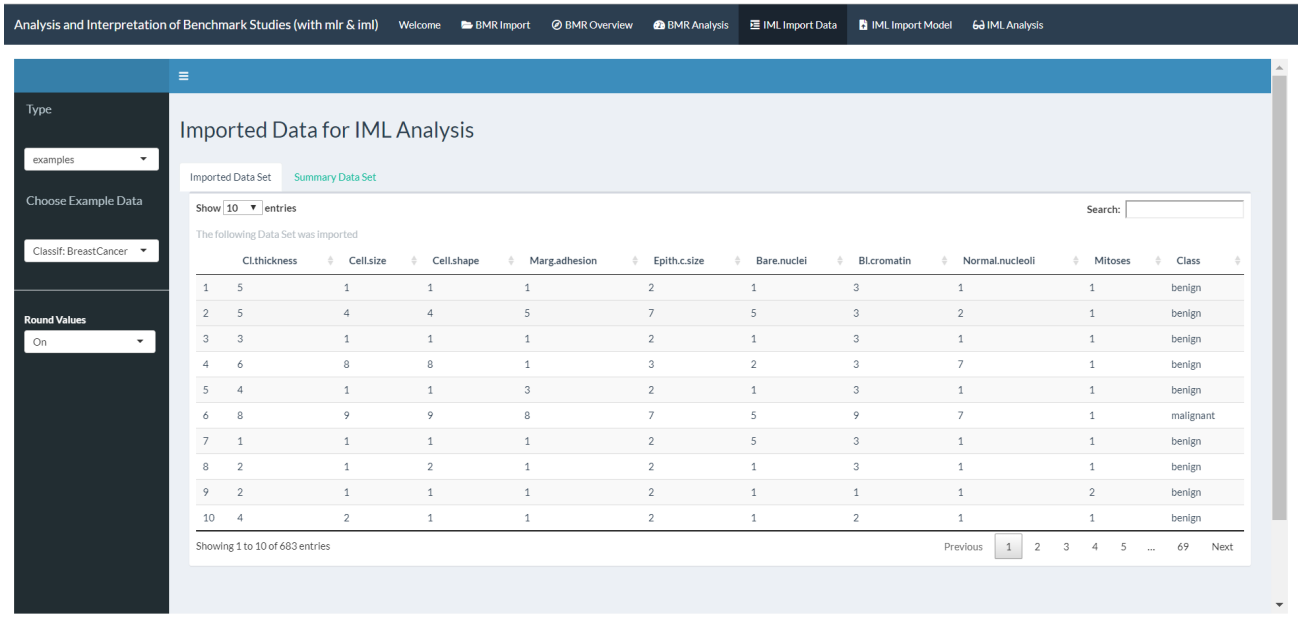
for the selection of *Choose Method to be focused* input on the sidebar.

II. Access to the Interpretation of Blackbox ML-Methods (IML)

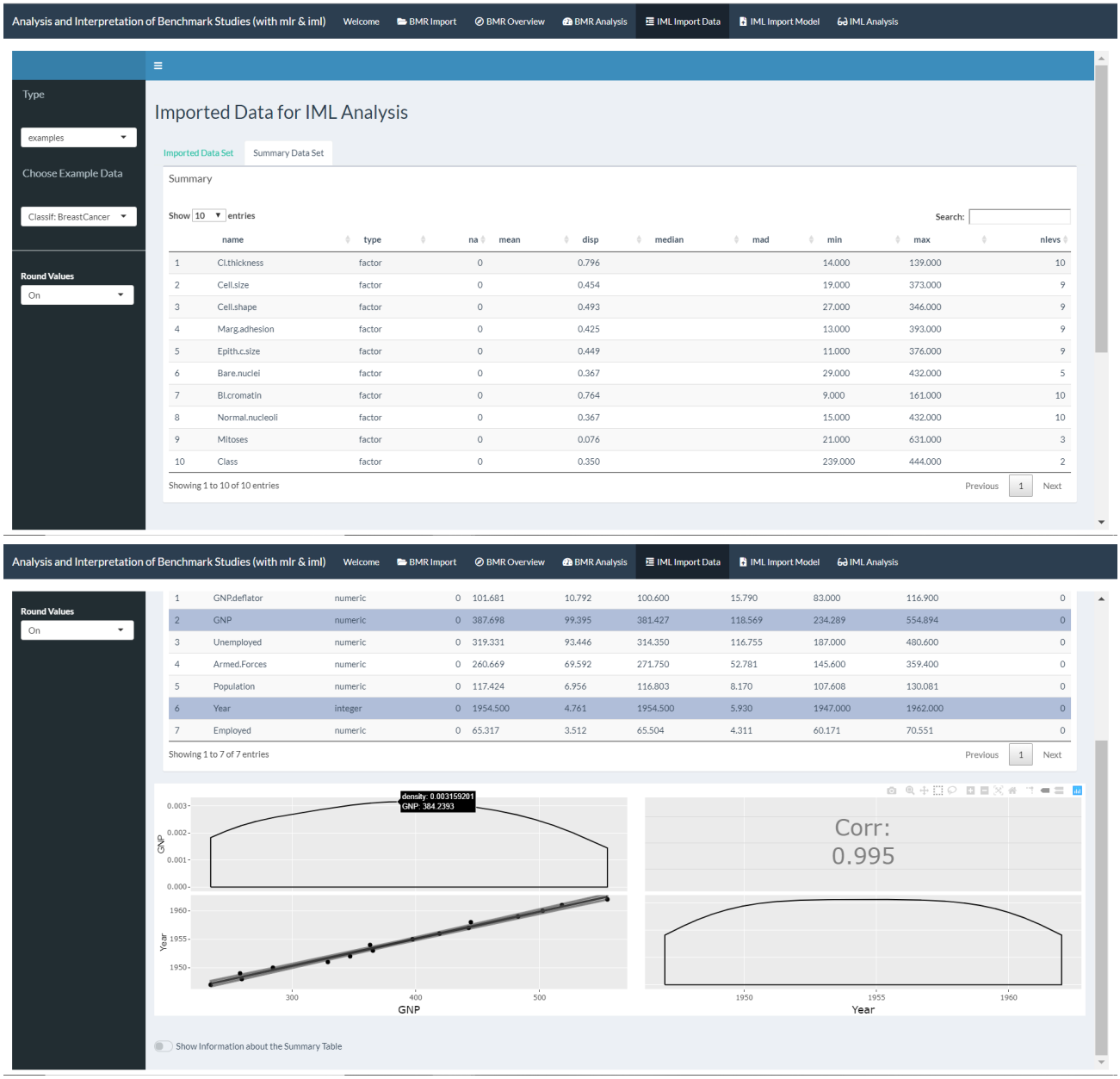
Having figured out, which ML method to use for your analysis of a specific data set and adjusting the final model you may end up using a blackbox method with high complex or even not tangible interpretation approaches. For these cases the iml package offers an opportunity to explain predictions of these ML methods. **shinyBMR** now also provides the option to use the functionalities of this package graphics in an interactive framework. The steps necessary for receiving the model-agnostic interpretability methods of your focused model are described in the following sections. As a side note it should be mentioned, that one can fit the final model easily with help of the additional package **shinyMlr** provided within the framewok of **mlr**.

1. Import and Overview of the focused Data Set

To get an interpretation approach for your model, you first of all have to provide the data set your model has been trained on. Therefore navigate to the *IML Import Data* tab in the **shinyBMR** app. As starting window the **Breast Cancer** data is displayed as example for a classification task in the sub-tab *Imported Data Set*. As further example one can select a regression task, which is in this case established via the **Longley's Economic** data and will be provided when the specific alternative is called at *Choose Example Data* on the sidebar. One can upload a data set by choosing the correct *Type* - in this case CSV, Rdata or RDS - of the file and navigating to the corresponding file with help of the *Browse* input tab. As soon as the upload is completed the data table will be displayed. Again, one can spezify *Round Values* on the sidebar for rounding options.



Having uploaded the data set you can now navigate to the sub-tab *Summary Data Set* for getting a summary of all variables included in form of a data table. For each of the variables *name* you get information about the *type* and the sum of missing values *na*. Depending on the type of the variable additionally the *mean*, measure of dispersion *disp*, *median*, *mad*, minimum *min*, maximum *max* and sum of levels *nlevels* are provided. In casee one needs more help in order to understand the summary one can switch the input *Show Information about the Summary Table* below the data table to get additional information.

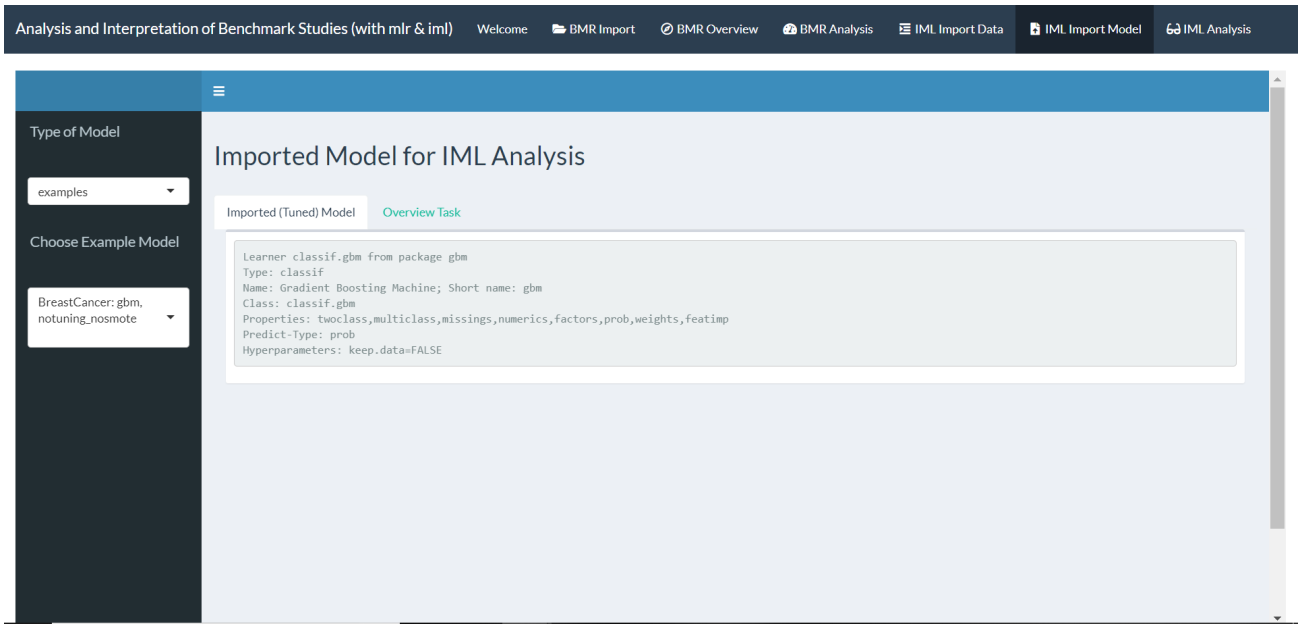


The *Summary Data Set* tab contains even more opportunities for visualizing information of your uploaded data. By selecting one or multiple rows of the summary, plots are displayed right under the data table. The diagnostics show a plot for each variable one by one and of their relationship to each other. Where the actual plot depends on the type of variable respectively the combination of types when more than one row is selected. By default two different ways of comparison of each variable with each can be choosen with the result, that either the *Density* or count of the respective variable *Histogram* along the diagonal is displayed. The plotting options are provided as part of the GGally package extending the ggplot2 package. Note that - as the Breast Cancer data does only contain factor variables - the picture above shows the plotting results, when choosing some of the variables comprised in Longley's Economic example data.

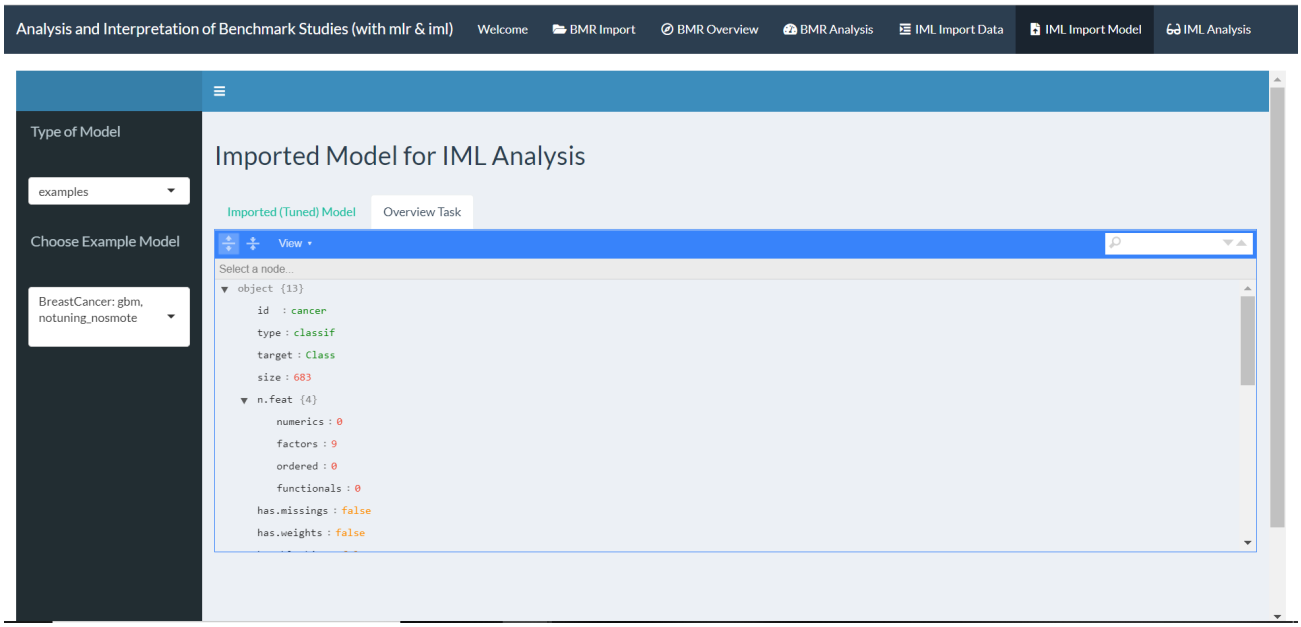
2. Import and Overview of the (Blackbox) Model

After your data is transferred to **shinyBMR** one can now upload the model of interest, which has been trained on the specific data set. Therefore navigate to the *IML Import Model* tab. As before, you can either select an example model trained on one of the example data sets - whereby the corresponding data set has to be

selected in tab *IML Import Data* - or upload your own model by choosing the format available of the specific model - in this case as RDS or Rdata file - and navigating to this model with help of the *Browse* tab.



In the sub-tab *Imported (Tuned) Model* some information about the model ypu have uploaded is displayed. As default a model trained on the Breast Cancer data is selected as starting window here. As soon as you have switched between the example models or have chosen your own model the corresponding information will be supplied.

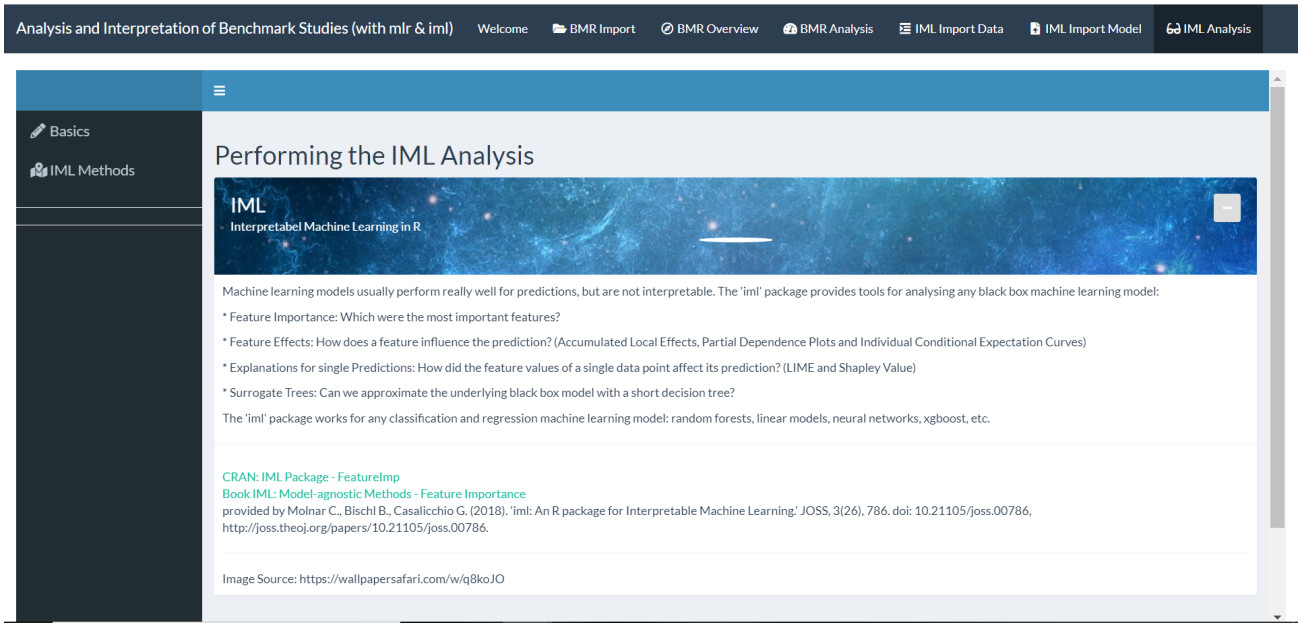


Since the models, which have been trained with help of the **mlr** package, always contain the task they have been trained on, one can have a detailed look on this by navigating to the sub-tab *Overview Task*, where a list with the retrievable information about the task will be shown.

3. Access to Interpretability provided by different tools of the **iml** package

Having completed the two steps mentioned before of *IML Import Data* and *IML Import Model* one can now move on to the concrete analysis of the ML method with help of the **iml** package. Before, it might be helpful

to go through the iml book for details concering - which are provided in the framework of this app - the model-agnostic interpretability methods. A short overview of the implemented iml methods as well as some helpful links are displayed as starting interface when selecting the according tab *IML Analysis* in the header of **shinyBMR**.



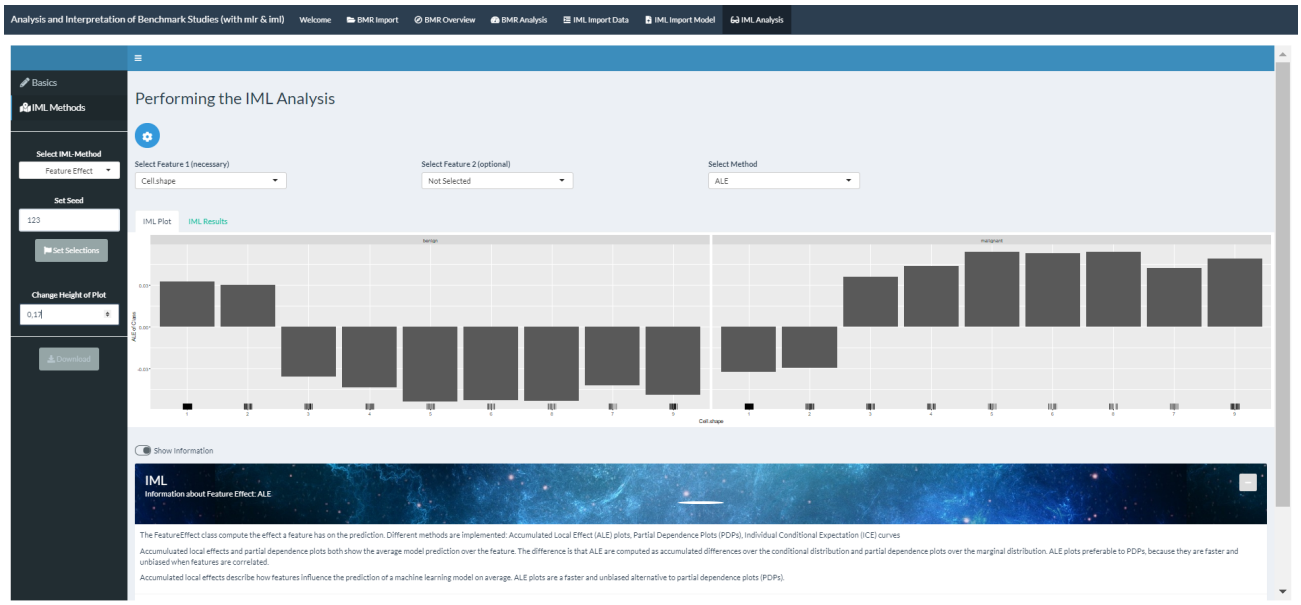
The concrete iml methods are provided when navigating to the *IML Methods* tab, which is located at the sidebar. Selecting this tab will lead to a new window with additional options in the sidebar. First of all, you have to *Select IML-Method* that you are interested in. One of the following iml methods can be chosen:

- Feature Importance
- Feature Effects
    - Partial dependence plots (PDP)
    - Individual conditional expectation plots (ICE)
    - Accumulated local effects (ALE)
- Feature Interaction
- Local Model: Local Interpretable Model-agnostic Explanations (LIME)
- Shapley Values (for explaining single predictions)
- Tree Surrogate

As these iml methods are somehow depending on coincidence, it is recommended to *Set Seed* at the sidebar tab to get reproducible results. Per default this seed is set to *123*. Some of the iml methods require necessary selections like the instance or variable of interest. In this case the demanded input is directly shown in the plotting window under the tooltip depending on the selection in the *Select IML-Method* tab. The tooltip is also individually reacting to your chosen iml method and contains additional options for the corresponding iml analysis tools as well as information about the settings selectable. To confirm your settings and finally plotting the iml methods you have to click on the *Set Selections* button. Every time you make a change in any selection of your settings, these have to be newly set by clicking on the *Set Selections* button again and thereby recalculating the specific results of the iml method.

While calculation is performed a spinner shows up in the plotting window and will be replaced by the corresponding plot as soon as computations have been finalised. There is one sub-tab in the plotting frame

selectable to show the actual *IML Plot*. Besides this, one can navigate towards the sub-tab *IML Results* to display the calculation results the plot is based on.



The sidebar of *IML Methods* also provides a selection panel for setting the height of the iml plots. Additionally one can download the latest plot by clicking the *Download* button and choosing the directory as well as the file name for the plot to be saved. Furthermore there is a slider in the plotting window to *Show Information* about the selected iml method. Here you can also use the links provided to navigate towards the corresponding chapters in the iml book such as to the according CRAN side holding the specific R function.

Literaturverzeichnis

A.Goldstein, Kapelner, A., Bleich, J. und Pitkin, E. (2013). Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation, *Journal of Computational and Graphical Statistics* **24**.

Alpaydin, E. (2010). Lecture Slides for: Introduction to Machine Learning, *Technical Report 2*, MIT Press. online verfügbar unter: `https://www.cmpe.boun.edu.tr/~ethem/i2ml2e/`.

AMBOSS - Fachwissen für Mediziner (2018). Printausgabe Fokus Physikum 2018/2019 - Band 1, `https://www.amboss.com/de/wissen/Gallenblase_und_Galle`. aufgerufen am 04.09.2018.

Apley, D. W. (2016). Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.

Austin, D. (2014). How to Grow and Prune a Classification Tree, `http://www.ams.org/publicoutreach/feature-column/fc-2014-12`. aufgerufen am 29.10.2018.

Bamber, D. (1975). The Area above the Ordinal Dominance Graph and the Area below the Receiver Operating Characteristic Graph, *Journal of Mathematical Psychology* **12**(4): 387–415.

Bardenet, R. (2012). *Towards adaptive learning and inference - Applications to hyperparameter tuning and astroparticle physics*, Dissertation, Université Paris Sud, Paris.

Becker, T. (2019). Mächtige Trias: KI, Machine Learning und menschgetragene Datenanalyse, *PC-Welt* .
URL: *https://www.pcwelt.de/ratgeber/Maechtige-Trias-KI-Machine-Learning-und-menschgetragene-Datenanalyse-10617155.html*

Bergstra, J. und Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research* **13**(1): 281–305.

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G. und Jones, Z. M. (2016). mlr: Machine Learning in R, *Journal of Machine Learning Research* **17**(170): 1–5.
URL: *http://jmlr.org/papers/v17/15-066.html*

Bischl, B., Mersmann, O., Trautmann, H. und Weihs, C. (2012). Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation, *Evolutionary Computation* **20**: 249–275.

Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J. und Lang, M. (2017). *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*. `http://arxiv.org/abs/1703.03373`. aufgerufen am 03.12.2018.

Börzsönyi, S., Kossmann, D. und Stocker, K. (2001). The skyline operator, *Proceedings of the 17th International Conference on Data Engineering*, IEEE Computer Society, Washington, DC, USA, S. 421–430.
URL: *http://dl.acm.org/citation.cfm?id=645484.656550*

Boulesteix, A.-L., Strobl, C., Augustin, T. und Daumer, M. (2008). Evaluating microarray-based classifiers: An overview, *Cancer Informatics* **6**.

Breiman, L. (1994). Bagging Predictors, *Technical Report 421*, Department of Statistics, University of California, Berkeley, California.

Breiman, L. (1996). Out-of-Bag Estimation, *Technical report*, Statistics Department, University of California, Berkeley, California.

Breiman, L. (2001). Random Forests, *Machine Learning* **45**(1): 5–32.

Breiman, L., J., J. F. C., Stone und Olshen, R. A. (1984). *Classification and Regression Trees*, The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis Group.

Bronshtein, A. (2017). A quick Introduction to k-Nearest Neighbors Algorithm, `https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7`. aufgerufen am 28.11.2018.

Brownlee, J. (2016). K-Nearest Neighbors for Machine Learning, `https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/`. aufgerufen am 27.11.2018.

Brownlee, J. (2018). How and When to Use a Calibrated Classification Model with scikit-learn, `https://machinelearningmastery.com/calibrated-classification-model-in-scikit-learn/`. aufgerufen am 18.12.2018.

Buehlmann, P. und Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting (with discussion), *Statistical Science* **22**(4): 477–505.

Bühlmann, P. und Yu, B. (2003). Boosting With the L2 Loss, *Journal of the American Statistical Association* **98**: 324–338.

Castrounis, A. (2016). Data Science and Big Data, Explained, `https://www.kdnuggets.com/2016/11/big-data-data-science-explained.html`. aufgerufen am 19.09.2018.

Chang, W. und Borges Ribeiro, B. (2018). *shinydashboard: Create Dashboards with 'Shiny'*. R package version 0.7.1.
URL: *https://CRAN.R-project.org/package=shinydashboard*

Chang, W., Cheng, J., Allaire, J., Xie, Y. und McPherson, J. (2019). *shiny: Web Application Framework for R*. R package version 1.3.2.
URL: *https://CRAN.R-project.org/package=shiny*

Chawla, N. V., Bowyer, K. W., Hall, L. O. und Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research* **16**: 321–357.

Chen, T. und Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System, *KDD*.

Cheung, A. C. (o. J.). Primary Biliary Cholangitis, `https://rarediseases.org/rare-diseases/primary-biliary-cholangitis/#references`. aufgerufen am 11.07.2019.

Coors, S. und Fendt, F. (2019). *shinyMlr: A graphical user interface for machine learning in R*. R package version 1.0.
URL: *https://github.com/mlr-org/shinyMlr*

Cristianini, N. und Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*, Cambridge University Press, New York, NY, USA.

Cule, E. und de Iorio, M. (2012). A semi-automatic Method to guide the Choice of Ridge Parameter in Ridge Regression.

de Mello, R. F. und Ponti, M. A. (2018). *Machine Learning - A Practical Approach on the Statistical Learning Theory*, Springer, Cham, Schweiz.

Eugster, M. J. A. (2011). *Benchmark Experiments - A Tool for Analyzing Statistical Learning Algorithms*, Dissertation, Ludwig-Maximilians-Universität, München.

F. E. Harrell, J. (2015). *Regression Modelling Strategies - With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*, Springer Series in Statistics, 2 Aufl., Springer.

Fahrmeir, L., Kneib, T. und Lang, S. (2009). *Regression - Modelle, Methoden und Anwendung*, 2 Aufl., Springer-Verlag, Heidelberg.

Fawcett, T. (2005). An introduction to ROC analysis, *Pattern Recogn. Lett.* **27**(8): 861–874.

Fawcett, T. und Niculescu-Mizil, A. (2007). Technical Note: PAV and the ROC Convex Hull, *Machine Learning* **68**(1): 97–106.

Ferri, C., Hernández-Orallo, J. und Modroiu, R. (2008). An experimental comparison of performance measures for classification, *Pattern Recognition Letters* **30**: 27–38.

Fisher, A., Rudin, C. und Dominici, F. (2018). Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the "Rashomon"Perspective.

Freund, Y. und Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of Computer and System Sciences* **55**(1): 119––139.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine, *The Annals of Statistics* **29**: 1189–1232.

Friedman, J. H., Hastie, T. und Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting, *The Annals of Statistics* **28**(2): 337–407.

Friedman, J., Hastie, T. und Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent, *Journal of Statistical Software* **33**(1): 1–22.
URL: *http://www.jstatsoft.org/v33/i01/*

Friedman, J. und Popescu, B. (2008). Predictive Learning via Rule Ensembles, *The Annals of Applied Statistics* **2**: 916 – 954.

Gelman, A. und Hill, J. (2006). *Missing-data imputation*, Analytical Methods for Social Research, Cambridge University Press, S. 529—-544.

Granville, V. (2017a). Difference between Machine Learning, Data Science, AI, Deep Learning, and Statistics, `https://www.datasciencecentral.com/profiles/blogs/difference-between-machine-learning-data-science-ai-deep-learning`. aufgerufen am 19.09.2018.

Granville, V. (2017b). Machine Learning Summarized in One Picture, `https://www.datasciencecentral.com/profiles/blogs/machine-learning-summarized-in-one-picture`. aufgerufen am 04.10.2018.

Hallinan, J. (2014). Assessing and Comparing Classifier Performance with ROC Curves, `https://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/`. aufgerufen am 16.10.2018.

Hastie, T., Tibshirani, R. und Friedman, J. (2008). *The Elements of Statistical Learning - Data Mining, Inference and Prediction*, 2 Aufl., Springer Series in Statistics New York, NY, Stanford, California.

Hofner, B., Boccuto, L. und Goeker, M. (2015). Controlling false discoveries in high-dimensional situations: Boosting with stability selection, *BMC Bioinformatics* **16**(144).

Hofner, B., Mayr, A., Robinzonov, N. und Schmid, M. (2014). Model-based boosting in R: A hands-on tutorial using the R package mboost, *Computational Statistics* **29**: 3–35.

Hornik, K., Strobl, C. und Zeileis, A. (2019). *Package 'party' - A Laboratory for Recursive Partytioning*. R Package version 1.3-3.

Hothorn, T., Buehlmann, P., Dudoit, S., Molinaro, A. und Van Der Laan, M. (2006). Survival ensembles, *Biostatistics* **7**(3): 355–373.

Hothorn, T., Buehlmann, P., Kneib, T., Schmid, M. und Hofner, B. (2010). Model-based boosting 2.0, *Journal of Machine Learning Research* **11**: 2109–2113.

Hothorn, T., Buehlmann, P., Kneib, T., Schmid, M. und Hofner, B. (2018). *mboost: Model-Based Boosting*. R package version 2.9-1.
URL: *https://CRAN.R-project.org/package=mboost*

Hothorn, T., Hornik, K. und Zeileis, A. (2005). Unbiased Recursive Partitioning I: A Non-Parametric Conditional Inference Framework, `https://eeecon.uibk.ac.at/~zeileis/papers/StatComp-2005a.pdf`. aufgerufen am 31.10.2018.

Hothorn, T., Hornik, K. und Zeileis, A. (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework, *Journal of Computational and Graphical Statistics* **15**(3): 651–674.

J. Vieira Barbosa, J. Vionnet, A. S. C. S. V. A. D. M. M. F. C. (2018). Primary biliary cholangitis : an update, *PubMed* **14**(626): 1289–1494.

James, G., Witten, D., Hastie, T. und Tibshirani, R. (2013). *An Introduction to Statistical Learning*, Springer, New York.

Jiao, Y. und Du, P. (2016). Review: Performance Measures in Evaluating Machine Learning based Bioinformatics Predictors for Classificators, *Higher Education Press and Springer-Verlag* **4**(4): 320–330.

Jordan, J. (2017). Evaluating a Machine Learning Model, `https://www.jeremyjordan.me/evaluating-a-machine-learning-model/`. aufgerufen am 20.11.2018.

Kearns, M. und Valiant, L. (1989). Cryptographic Limitations on learning Boolean Formulae and Finite Automata, *Proc. of the 21st Symposium on Theory of Computing*, ACM Press, Seattle, WA, S. 433–444.

Koehrson, W. (2018). Hyperparameter Tuning the Random Forest in Python, https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74. aufgerufen am 05.02.2019.

Krech, D. und Crutchfield, R. (1992). *Grundlagen der Psychologie - Studienausgabe*, 3, Weinheim.

Li, W. und Chen, H. (o. J.). Logistic Regression Elastic Net, `https://webcache.googleusercontent.com/search?q=cache:vFCVAFyFL7wJ:https://ai.arizona.edu/sites/ai/files/resources/logistic_regression_and_elastic_net.pptx+&cd=6&hl=de&ct=clnk&gl=de`. aufgerufen am 11.02.2019.

Lohninger, H. (2012). Neuronale Netze: Klassifizierung und Diskriminierung, `http://www.statistics4u.info/fundstat_germ/wrapnt536557_klassifizierung_und_diskriminierung.html`. aufgerufen am 11.10.2018.

Lopez-Ibanez, M., Dubois-Lacoste, J., Stützle, T. und Birattari, M. (2011). The irace Package: Iterated Racing for Automatic Algorithm Configuration, *Technical Report 4*, IRIDIA, Brüssel.

Machart, P. (2012). *Coping with the Computational and Statistical Bipolar Nature of Machine Learning*, Dissertation, Aix-Marseille Université.

Mayr, A., Binder, H., Gefeller, O. und Schmid, M. (2014). The Evolution of Boosting Algorithms From Machine Learning to Statistical Modelling, *Methods of Information in Medicine* **53**: 1–32.

McClish, D. K. (1989). Analyzing a Portion of the ROC Curve, *Medical Decision Making* **9**(3): 190–195.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. und Leisch, F. (2019a). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-2.
URL: *https://CRAN.R-project.org/package=e1071*

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. und Leisch, F. (2019b). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-2.
URL: *https://CRAN.R-project.org/package=e1071*

Mierswa, I. (2017a). What is Artificial Intelligence, Machine Learning, and Deep Learning?, `https://ingomierswa.com/2017/04/19/what-is-artificial-intelligence-machine-learning-and-deep-learning/`. aufgerufen am 13.09.2018.

Mierswa, I. (2017b). What is Data Science?, `https://rapidminer.com/blog/what-is-data-science/`. aufgerufen am 13.09.2018.

Miller, T. (2017). Explanation in Artificial Intelligence: Insights from the Social Sciences, *CoRR* **abs/1706.07269**.
URL: *http://arxiv.org/abs/1706.07269*

Moeser, J. (2017). Starke KI, schwache KI - Was kann künstliche Intelligenz?, `https://jaai.de/starke-ki-schwache-ki-was-kann-kuenstliche-intelligenz-261/`. aufgerufen am 13.09.2018.

Molinaro, A. M., Simon, R. und Pfeiffer, R. M. (2005). Prediction Error Estimation: A Comparison of Resampling Methods, *Bioinformatics* **21**(15): 3301–3307.

Molnar, C. (2018). Interpretable Machine Learning - A Guide for Making Black Box Models Explainable, `https://christophm.github.io/interpretable-ml-book/`. aufgerufen am 18.02.2019.

Morik, K. (2013). Maschinelles Lernen, *Vorlesungsskript*, Technische Universität Dortmund.

Morik, K. und Ligges, U. (2013). Wissensentdeckung in Datenbanken, *Vorlesungsskript*, Technische Universität Dortmund.

Mosler, P. (2011). Diagnosis and management of acute cholangitis, *Current Gastroenterology Reports* **13**(2): 166–172.

Natekin, A. und Knoll, A. (2013). Gradient boosting machines, a tutorial, *Frontiers in Neurorobotics* **7**: 21.
URL: *https://www.frontiersin.org/article/10.3389/fnbot.2013.00021*

Pekhimenko, G. (2006). Penalizied Logistic Regression for Classification.

Pereira, J. M., Basto, M. und da Silva, A. F. (2016). The logistic lasso and ridge regression in predicting corporate failure, *Procedia Economics and Finance* **39**: 634–641.

Probst, P., Boulesteix, A.-L. und Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms, *Journal of Machine Learning Research* **20**(53): 1–32.
URL: *http://jmlr.org/papers/v20/18-444.html*

Probst, P., Wright, M. und Boulesteix, A.-L. (2018). Hyperparameters and Tuning Strategies for Random Forest, *arXiv e-prints* .

Provost, F. und Fawcett, T. (2001). Robust Classification for Imprecise Environments, *Machine Learning* **42**: 203–231.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Rezac, M. und Rezac, F. (2011). How to measure the Quality of Credit Scoring Models, *Journal of Economics and Finance* **61**(5): 486–507.

Riberio, M. T., Singh, S. und Guestrin, C. (2016). Why should I trust you?: Explaining the predictions of any classifier, *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, ACM, New York, NY, USA, S. 1135–1144.

Richter, J. (2017). mlrHyperopt, `http://jakob-r.de/mlrHyperopt/articles/mlrHyperopt.html`. aufgerufen am 04.02.2019.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C. und Müller, M. (2011). proc: an open-source package for r and s+ to analyze and compare roc curves, *BMC Bioinformatics* **12**: 77.

Roocks, P. (2016). Computing Pareto Frontiers and Database Preferences with the rPref Package, *The R Journal* **8**(2): 393–404.

Sayad, S. (o. J.). K Nearest Neighbors - Classification, `https://www.saedsayad.com/k_nearest_neighbors.htm`. aufgerufen am 28.11.2018.

Schapire, R. E. (1990). The Strength of Weak Learnability, *Machine Learning* **5**(2): 197–227.

Schapire, R. E. und Freund, Y. (2012). *Boosting: Foundations and Algorithms*.

Scheipl, F., Große-Wentrup, M. und Bischl, B. (2018). FCIM: Fortgeschrittene Computerintensive Methoden, *Vorlesungsfolien*, Ludwig-Maximilians-Universität, München.

Schilling, J. (1997). *Soziale Arbeit*, Leuterhand, Berlin.

Schimek, M. G. (2003). Penalized Logistic Regression in Gene Expression Analysis.

Schliep, K. und Hechenbichler, K. (2016). *kknn: Weighted k-Nearest Neighbors*. R package version 1.3.1.
URL: *https://CRAN.R-project.org/package=kknn*

Schneider, J., Hapfelmeier, A., Thöres, S., Obermeier, A., Schulz, C., Pförringer, D., Nennstiel, S., Spinner, C., Schmid, R. M., Algül, H., Huber, W. und Weber, A. (2016). Mortality risk for acute cholangitis (mac): a risk prediction model for in-hospital mortality in patients with acute cholangitis, *BMC Gastronentrology* **16**(15): 1–8.

Schonberg, M., B Davis, R., Mccarthy, E. und R Marcantonio, E. (2011). External validation of an index to predict up to 9-year mortality of community-dwelling adults aged 65 and older, *Journal of the American Geriatrics Society* **59**: 1444–51.

Shah, A. (2016). Machine Learning vs. Statistics, `https://www.kdnuggets.com/2016/11/machine-learning-vs-statistics.html`. aufgerufen am 19.09.2018.

Simon, N., Friedman, J., Hastie, T. und Tibshirani, R. (2011). Regularization paths for cox's proportional hazards model via coordinate descent, *Journal of Statistical Software* **39**(5): 1–13.
URL: *http://www.jstatsoft.org/v39/i05/*

Soni, D. (2018). Supervised vs. Unsupervised Learning - Understanding the differences between the two main types of machine learning methods, `https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d`. aufgerufen am 04.10.2018.

Stekhoven, D. J. und Bühlmann, P. (2011). MissForest—non-parametric missing value imputation for mixed-type data, *Bioinformatics* **28**(1): 112–118.

Strasser, H. und Weber, C. (1999). On the asymptotic theory of permutation statistics, *Mathematical Methods of Statistics* **8**: 220–250.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T. und Zeileis, A. (2008). Conditional variable importance for random forests, *BMC Bioinformatics* **9**(307).
URL: *http://www.biomedcentral.com/1471-2105/9/307*

Strobl, C., Boulesteix, A.-L., Zeileis, A. und Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution, *BMC Bioinformatics* **8**(25).
URL: *http://www.biomedcentral.com/1471-2105/8/25*

Stöcker, M. (2007). *Untersuchung von Optimierungsverfahren für rechenzeitaufwendige technische Anwendungen in der Motorenentwicklung*, Dissertation, Fbakultät für Mathematik der Technischen Universität Chemnitz, Chemnitz.

Swalin, A. (2018). Choosing the right Metric for Evaluating Machine Learning Models - Part I, `https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4`. aufgerufen am 20.11.2018.

Therneau, T. M., Atkinson, E. J. und Foundation, M. (2017). An Introduction to Recurve Partitioning Using the RPART Routines, `https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf`. aufgerufen am 29.10.2018.

Tibshirani, R. (2012). *Glossary of Machine Learning and Statistical Terms*, Stanford University. `http://statweb.stanford.edu/~tibs/stat315a/glossary.pdf`. aufgerufen am 22.09.2018.

Timischl, W. (2013). *Angewandte Statistik - Eine Einführung für Mediziner und Biologen*, 3 Aufl., Springer-Verlag, Wien.

Vastmans, J. (2019). Privates Interview. BG Unfallklinik Murnau.

Wasserman, L. (2012). Statistics versus Machine Learning, `https://normaldeviate.wordpress.com/2012/06/12/statistics-versus-machine-learning-5-2/`. aufgerufen am 14.09.2018.

Wehberg, S., Sauerbrei, W. und Schumacher, M. (2007). Diagnosestudien: Wertigkeit der Sonographie bei der Differenzierung von gut- und bösartigen Brusttumoren bei Patientinnen mit klinischen Symptomen, *Methodik klinischer Studien - Methodische Grundlagen der Planung, Durchführung und Auswertung*, Vol. 2, Springer, Berlin, S. 319–340.

Wehner, J. (o. J.). Gallenblase und Gallenwege, `http://www.medizinfo.de/leber/anatomie/galle.shtml`. aufgerufen am 05.09.2018.

Witten, I. H., Frank, E. und Hall, M. A. (2011). *Data Mining: Partical Machine Learning Tools and Techniques*, 3 Aufl., Elsevier Science.

Wübbenhorst, K. (2018). Benchmarking, `https://wirtschaftslexikon.gabler.de/definition/benchmarking-29988`. aufgerufen am 10.10.2018.

Y. Yue, X. Zuo, X. L. (2013). Confidence Level Based on Ridge Estimator in Process Measurement and Its Application, *Chinese Journal of Chemical Engineering* **21**(10): 1144 – 1154.

Yokoe, M., Takada, T., Strasberg, S. M., Solomkin, J. S., Mayumi, T., Gomi, H., Pitt, H. A., Garden, O. J., Kiriyama, S., Hata, J., Gabata, T., Yoshida, M., Miura, F., Okamoto, K., Tsuyuguchi, T., Itoi, T., Yamashita, Y., Dervenis, C., Chan, A. C. W., Lau, W.-Y., Supe, A. N., Belli, G., Hilvano, S. C., Liau, K.-H., Kim, M.-H., Kim, S.-W. und Ker, C.-G. (2013). Tg13 diagnostic criteria and severity grading of acute cholecystitis (with videos), *Journal of Hepato-Biliary-Pancreatic Sciences* **20**(1): 35–46.

Zhao, Q. und Hastie, T. (2018). Causal Interpretations of Black-Box Models, *Journal of Business Economic Statistics* .

Zheng, A. (2015). *Evaluating Machine Learning Models*, 1 Aufl., O'Reilly, Sebastopol, California.

Zhou, Z.-H. (2012). *Ensemble Methods - Foundations and Algorithms*, Chapman and Hall/-CRC Press.

Zimbardo, P. G. und Gerrig, R. J. (1999). *Psychologie*, Springer, Heidelberg.

Zou, H. und Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net, *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **67**(2): 301–320.

Eidesstattliche Erklärung

Hiermit versichere ich, Anna Theresa Stüber, die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Die verwendete Literatur und sonstige Hilfsmittel sind vollständig angegeben.
Die Arbeit wurde bisher noch keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

München, den 14. Juli 2019

.....
Unterschrift