Vorhersagen von Komplikationen nach einer Subarachnoidalblutung anhand von Machine Learning Methoden

Viktoria Blombach

Eine Arbeit präsentiert für den Abschluss
Bachelor of Science

Algorithmische Bioinformatik
Heinrich Heine Universität Düsseldorf
Deutschland
9. August 2022

Danksagung

Zuerst gebührt mein Dank Prof. Dr. Klau, der meine Bachelorarbeit betreut und mich bei der Themenorientierung beraten hat. Vielen Dank auch an Prof. Dr. Dilthey, der sich als Zweitkorrektor für diese Bachelorarbeit bereit erklärt hat. Weiterhin möchte ich mich bei Philipp Spohr für seine hilfreichen Anregungen und Unterstützung durch die gesamte Zeit hinweg bedanken sowie Nguyen Khoa Tran, der mich in den letzten beiden Wochen als Vertretung für Philipp Spohr ebenfalls unterstützt hat. Ein besonderer Dank gilt Alberto Descalzo Garcia, der mich in der Vorbereitung und dem Verständnis der Daten intensiv unterstützt hat. Ebenfalls möchte ich mich bei meinen Freunden und Familie bedanken, dass sie mir durchgehend mit viel Geduld, technische und moralische Unterstützung geleistet haben.

Zusammenfassung

In dieser Bachelorarbeit soll untersucht werden, ob mithilfe von Machine Learning Methoden und den vom Uniklinikum Düsseldorf zur Verfügung gestellten Daten von an einer Subarachnoidalblutung Erkrankten, frühzeitig erkennen kann, ob es zu Komplikationen in Form eines Vasospasmus oder einer verzögerten zerebralen Ischämie kommen wird. Dabei beschreibt Machine Learning das Lernen von Systemen zum Erkennen von Mustern in vorhandenen Datensätzen. Aufgrund dessen, sollen Lösungen zu Problemen entwickelt oder unbekannte Daten analysiert werden. Für diese Umsetzung, muss das System mit relevanten Daten und Algorithmen ausgestattet werden. (Luber und Litzel 2016)

In der heutigen Zeit steigt die Relevanz von Machine Learning Methoden und soll vor allem im Bereich der Medizin das ärztliche Fachpersonal beim Erstellen von schnelleren und genaueren Diagnosen unterstützen. Insbesondere bei großen Datenmengen kann der Computer umfangreiche Analysen erstellen und Muster erkennen. Auf Grund des angelernten Algorithmus benötigt der Computer nur einen Bruchteil der Zeit, die ein Mensch für diese Aufgabe benötigen würde. (Mainzer 2022)

Um die Forschungsfrage zu beantworten, wurden Daten von Erkrankten einer Subarachnoidalblutung des Universitätsklinikum Düsseldorf als Grundlage genutzt. Einen GroSSteil der Zeit wurde für die Vorbereitung der Daten auf die Machine Learning Methoden verwendet. Sie mussten in ein anderes Darstellungsformat überführt werden, sowie überprüft werden, ob alle Daten von Relevanz sind. Zusätzlich sollte untersucht werden, ob der Durchschnitt der Vitalwerte oder der Maximalwert der Vitalwerte besser geeignet ist, um die Komplikationen mithilfe der Machine Learning Methoden vorherzusagen.

Hinterher wurden eine Anzahl von ausgewählten Machine Learning Algorithmen verwendet, um festzustellen welches Modell am besten geeignet ist. Jedes Modell wurde bewertet und analysiert, dabei ist ein Modell mit besonders vielversprechenden Ergebnissen aufgefallen: das Logistic Regression Modell.

Inhaltsverzeichnis

1 Einleitung **1**

2 Hintergrund **2**

3 Machine Learning Methoden **3**

3.1 Datenskalierung . . . 3

3.1.1 Datenskalen . . . 3

3.1.2 One-Hot-Encoder . . . 4

3.1.3 Standard Scaler . . . 5

3.1.4 Equal-width Binning . . . 5

3.2 Supervised Learning Methoden . . . 6

3.2.1 k-Nearest-Neighbor . . . 6

3.2.2 Logistic Regression . . . 7

3.2.3 Decision Tree Classifier . . . 8

3.2.4 Random Forest Classifier . . . 8

3.2.5 Gradient Boosting Classifier . . . 9

3.2.6 Support Vector Machine mit Kernel . . . 10

3.3 Unsupervised Learning Methoden . . . 11

3.3.1 Principal Component Analysis . . . 12

3.3.2 k-means-Clustering . . . 12

3.3.3 Agglomerative Clustering . . . 13

3.3.4 DBSCAN . . . 13

3.4 Performance Metrics . . . 14

3.4.1 Confusion Matrix . . . 14

3.4.2 F1-Score . . . 15

3.4.3 Accuracy . . . 16

3.4.4 Matthews Correlation Coefficient . . . 16

4 Aufbereitung der Medizinischen Daten **18**

4.1 Datenaufbereitung . . . 18

4.2 Skalierung . . . 20

5 Ergebnisse **24**

5.1 Auswertung Supervised . . . 24

5.2 Auswertung Unsupervised . . . 31

6 Diskussion & Fazit **37**

6.1 Diskussion . . . 37

6.2 Fazit . . . 38

A Anhang **43**

Abbildungsverzeichnis **51**

Liste für Abkürzungen **53**

1 Einleitung

ML beschreibt das Lernen von Systemen zum Erkennen von Mustern in vorhandenen Datensätzen um aufgrund dessen Lösungen zu Problemen zu entwickeln oder unbekannte Daten zu analysieren (Luber und Litzel 2016). Gerade in der heutigen Zeit steigt die Relevanz von Maschinellem Lernen und künstlicher Intelligenz enorm. Nicht nur in den Bereichen Informatik, Prozessoptimierung und Forschung gewinnt ML immer mehr an Bedeutung (IAIS o. D.).

Auch die Medizin hat die Vorteile von diesen Algorithmen erkannt. Seit einiger Zeit wird das ärztliche Fachpersonal in der Diagnostik unterstützt (Mainzer 2022). Gleichzeitig wird die menschliche Fehleranfälligkeit dezimiert. Dabei ist besonders zu beachten: „Menschliches Denken soll bei Machine Learning nicht imitiert, sondern ergänzt werden.“ (acatech 2020).

In der Vergangenheit wurde ML bereits erfolgreich bei der Vorhersage für Krankheiten eingesetzt. Im Bereich Deep Learning wurde ein selbstlernendes System optimiert, um Brustkrebs frühzeitig zu erkennen (Ribli u. a. 2018).

Die internistisch-neurologische Intensivstation des Universitätsklinikums Düsseldorf hat sich auf den Bereich neurovaskulärer Erkrankungen spezialisiert. Aus diesem Grund werden jährlich zahlreiche Personen mit Kopfverletzungen unterschiedlichster Ausprägungen auf der Intensivstation aufgenommen und behandelt (Düsseldorf o. D.). Darunter fällt auch die SAB, eine Blutung zwischen der mittleren und inneren Hirnhaut zumeist durch ein Aneurysma verursacht. Sie verursacht als alleinstehende Krankheit schon schwere Schäden (Poeck und Hacke 2016). Jedoch können Betroffene während der Behandlung auf der Intensivstation mit weiteren Komplikationen konfrontiert werden, die unverzüglich behandelt werden müssen.

Komplikationen, wie z.B. Vasospasmen und verzögerte zerebrale Ischämie, können den Gesundheitsstatus der Betroffen erheblich verschlechtern (Rieg u. a. 2017). Sie führen zu einer Verengung der Gefäße im Gehirn, die bereits in kurzer Zeit irreparable Schäden verursachen (Guide 2021; Heidelberg o. D.). Wäre es möglich, Anzeichen für Komplikationen frühzeitig zu entdecken, könnte das ärztliche Fachpersonal schneller reagieren und somit schwerwiegenden Schäden vorbeugen.

In dieser Arbeit wird der Fokus auf eben dieses Krankheitsbild der SAB und der daraus resultierenden Komplikationen gelegt. Ziel hierbei ist es, zu untersuchen, welche Schlussfolgerungen aus den Daten von aktuell behandelten Patienten einer SAB, mittels verschiedener ML Methoden, gezogen werden können. Des Weiteren soll untersucht werden, ob mögliche Komplikationen bzw. eine Verschlechterung des Gesundheitszustandes frühzeitig erkannt werden können. Dafür werden verschiedene ML Algorithmen auf Ihre Tauglichkeit geprüft und auf die vorliegenden Daten der Patienten des Universitätsklinikums Düsseldorfs angewendet. Die Ergebnisse werden miteinander verglichen, um eine Einschätzung abgeben zu können, ob eine Früherkennung möglicher Komplikationen vorhersehbar ist.

2 Hintergrund

In diesem Abschnitt wird auf den wissenschaftlichen Hintergrund dieser Bachelorarbeit eingegangen. Unter anderem wird ein Überblick über die SAB, sowie ihre Komplikationen verschafft und welche Art von ML angewendet werden soll.

Die SAB betrifft das zentrale Nervensystem. Dabei tritt freies Blut in den mit Hirnflüssigkeit gefüllten Subarachnoidalraum, der zwischen der mittleren und inneren Hirnhaut liegt, ein. In diesem Bereich verlaufen viele Blutgefäße um das Gehirn herum. Platzt ein solches Gefäß, breitet sich das Blut im Subarachnoidalraum aus und drückt von außen auf das Gehirn. Im Gehirn selber entstehen dabei keine Blutungen (Poeck und Hacke 2016). Bei etwa 85% aller Fälle wird eine SAB durch eine Ruptur eines Aneurysmas verursacht (Rieg u. a. 2017). Aneurysmen sind sackförmige Aufweitungen der arteriellen Gefäßwände und führen zu einer Wandverdünnung der Gefäße im betroffenen Bereich. Steigt der Blutdruck in besagtem Abschnitt, führt dies zu einer Ruptur des Gefäßes. Wird ein Aneurysma nicht umgehend behandelt, kann es zu schweren dauerhaften Gehirnschädigungen führen (Poeck und Hacke 2016).

Die Inzidenz einer SAB wird auf 2 - 22.5 pro 100.000 Einwohner geschätzt. Im Durchschnitt erkranken Frauen häufiger an einer SAB als Männer; das Verhältnis liegt bei 3:2 (Kundra u. a. 2014). Circa 10% aller SAB-Erkrankten sterben noch vor der Behandlung, während 35% innerhalb der ersten 30 Tage nach Behandlungsbeginn versterben. Nur etwa 50% der Betroffenen leben noch nach einem Jahr des Initialereignisses (erstes Auftreten der SAB). Die Hälfte aller Erkrankten ist dauerhaft auf Pflege angewiesen (Rieg u. a. 2017). Der Gesundheitsstatus der Betroffenen ist von verschiedenen Faktoren abhängig, wie z.B. dem Auftreten einer sogenannten verzögerten zerebralen Ischämie (D'Souza 2015) oder auch Vasospasmen (Rieg u. a. 2017).

Bei der zerebralen Ischämie kommt es zu einer Minderdurchblutung, auch Ischämie genannt, von Gewebeabschnitten des Gehirns (Guide 2021). Die verzögerte zerebrale Ischämie tritt bei etwa 30% der zu Behandelnden zwischen Tag 4 - 14 der initialen Behandlung der SAB auf (Dreier 2015). Mit einer Häufigkeit von 30 - 70% können auch Vasospasmen im gleichen Zeitraum auftreten (Rieg u. a. 2017; AMBOSS o. D.). Vasospasmen, auch Gefäßspasmen, führen zu einer Reizung der Blutgefäße und somit zu einer aktiven Verengung der betroffen Gefäße (Heidelberg o. D.). Die Folge von beiden Komplikationen ist eine Minderversorgung von Sauerstoff im Gehirn, was zu schwerwiegenden Behinderungen, Schlaganfällen oder zum Tod führen kann (Dreier 2015; Poeck und Hacke 2016).

Aufgrund der hohen Mortalitätsrate ist es für das ärztliche Fachpersonal von enormer Wichtigkeit frühst möglich zu Erfahren, ob eine der beschriebenen Komplikationen wahrscheinlich auftreten wird. Aus diesem Grund soll untersucht werden, ob man anhand von ML Methoden diese Komplikationen vorhersagen kann.

Im ML unterscheidet man in zwei Arten von Problemen. Zum einen Regressionsprobleme, die vorhersagen sollen, wie eine kontinuierliche Variable steigt oder fällt (Nguyen und Zeigermann 2017, S. 7). Ein Beispiel dafür wäre, die Immobilienpreise in Deutschland in

vier Jahren vorherzusagen. Zum anderen gibt es Klassifikationsprobleme. Es sollen kategorische Antworten gegeben werden, z.B. ob eine Situation eintreffen wird oder nicht. Hierbei wird noch einmal in binäre Klassifikationsprobleme und Klassifikationsprobleme mit mehreren Klassen unterschieden (Nguyen und Zeigermann 2017, S. 7). Unter die binäre Klassifikation fällt das Problem, welches in dieser Bachelorarbeit behandelt werden soll. Es wird der Frage nachgegangen, ob eine behandelte Person Komplikationen erleiden wird oder nicht.

3 Machine Learning Methoden

Im Folgenden werden Methoden beschrieben, die bedeutend sind um die Daten auf die ML Algorithmen vorzubereiten. Außerdem wird ein Überblick zu den verschiedenen ML Methoden zur Klassifikation gegeben. Im Zuge dieser Arbeit wird eine bestimmte Auswahl an Algorithmen genutzt, um die Daten der Betroffenen zu untersuchen. Im Weiteren wird auf Supervised und Unsupervised ML Methoden eingegangen, sowie die Algorithmen, welche auf diesen Datensatz angewendet werden. Vorab kann nicht gesagt werden, welcher ML Algorithmus besonders geeignet ist. Daher müssen verschiedene Algorithmen genutzt und gegenübergestellt werden (Gharagyozyan 2019). Hierzu kann die Anzahl der Algorithmen auf die Anforderungen reduziert werden. Es müssen Klassifikationsalgorithmen genuzt werden, die mit numerischen Werten gute Ergebnisse erzielen können. Da es keine Bilder zu den Daten der Erkrankten gibt, schließt das Algorithmen aus die insbesondere mit Graphiken zurecht kommen. Zuletzt werden Methoden zur Beurteilung eines ML Modells aufgelistet und erläutert.

3.1 Datenskalierung

Damit Daten für ML Methoden genutzt werden können, müssen diese immer in ein bestimmtes Format gebracht werden, um einheitlich zu sein (Pedregosa u. a. 2011a, vgl. Importance of Feature Scaling). Weil die Algorithmen, die verwendet werden, nur mit numerischen Werten umgehen können, wird in den nächsten Abschnitten erläutert, mit welchen Methoden Daten in ein anderes Datenformat umgewandelt werden können.

3.1.1 Datenskalen

Daten jeglicher Art unterliegen bestimmten Skalen, in welche sie eingeordnet werden können. Insgesamt gibt es vier verschiedene Skalenniveaus (Fahrmeir u. a. 2016, S. 15):

- Nominalskala
- Ordinalskala
- Intervallskala
- Verhältnisskala

Nominalskala: Die Nominalskala ist das niedrigste Skalenniveau und ordnet Merkmalsträger einer bestimmten Kategorie zu, die keiner Rangordnung folgen oder mit einer anderen Kategorie verglichen werden können. Entweder sind die Kategorien der Merkmalsträger gleich oder verschieden (Fahrmeir u. a. 2016, S. 15 f.). Darunter fallen beispielsweise aus den vorliegenden medizinischen Daten das Geschlecht oder die Position des Aneurysma.

Ordinalskala: Die nächsthöhere Stufe ist die Ordinalskala, die die Merkmalsträger nicht nur einer bestimmten Kategorie, sondern auch einer natürlichen Rangordnung zuordnet. Jedoch sind diese Abstände nicht miteinander zu vergleichen. Es kann nur gesagt werden, ob eine Kategorie „besser“oder „schlechter“ist (Fahrmeir u. a. 2016, S. 16). Im Falle der medizinischen Daten kann als Beispiel die Bewertungen der Zustände genannt werden, mit denen die erkrankten Personen eingestuft werden, sobald sie ins Krankenhaus gebracht werden.

Intervallskala: Die Intervall- und Verhältnisskala gehören dem höchsten Skalenniveau an, den Kardinalskalen. Bei der Intervallskala kommt hinzu, dass die Abstände zwischen den Merkmalsausprägungen gleich sind. Das bedeutet, dass diese Merkmalsträger nicht nur in größer und kleiner eingeordnet werden können, sondern dadurch auch ein Abstand zwischen den Merkmalsträgern herrscht. Die Intervallskala hat jedoch einen willkürlich gesetzten Nullpunkt (Fahrmeir u. a. 2016, S. 16). Als Beispiel könnte man die Temperaturanzeige in Celsius angeben oder auch die IQ-Skala. In den medizinischen Daten liegen keine Daten dieses Skalenniveaus vor.

Verhältnisskala: Im Gegensatz zur Intervallskala besitzt die Verhältnisskala einen absoluten Nullpunkt. Das bedeutet, dass nur positive Werte angenommen werden können. Bei diesem Skalenniveau dürfen Verhältnisse zwischen den Merkmalsträgern gebildet werden. Multiplikation und Division sind also erlaubt, da das Ergebnis wieder einen Merkmalsträger bildet (Fahrmeir u. a. 2016, S. 16). Hierunter fallen die Vitalwerte und das Alter der Patienten.

3.1.2 One-Hot-Encoder

Der OHE wird verwendet, um Daten umzuwandeln, die als Stringwert vorliegen und der Nominalskala zugeordnet werden können. Dazu durchläuft die Methode alle Kategorien und ordnet diese jeweils einer eigenen Spalte zu. Trifft ein Merkmal zu, wird der Wert 1 eingesetzt und 0, wenn ein Merkmal nicht zutrifft. So kann jedes Merkmal an einem bestimmten Tupel erkannt werden.(D. M. Harris und S. L. Harris 2013, S. 129)

Als Beispiel werden Farben in Abbildung 1 verwendet:

	rot	grün	blau
0	1	0	0
1	0	1	0
2	0	0	1

Abbildung 1: OHE Beispiel anhand von Farben

Für den OHE wird eine Methode in der «pandas»-Bibliothek bereitgestellt, in die die Spalte mit den verschiedenen Kategorien übergeben und umformatiert zurück gegeben wird.

3.1.3 Standard Scaler

Um einige ML Methoden richtig anwenden zu können, müssen die Daten vorher standardisiert werden, wie für Logistic Regression, Support Vector Machine sowie alle unsupervised Methoden (Pedregosa u. a. 2011a, vgl. Importance of Feature Scaling). Beim Standardisieren werden die Daten so transformiert, dass alle Merkmale den Mittelwert null und die Varianz eins haben (Fahrmeir u. a. 2016, S. 269). So wird sicher gestellt, dass alle Daten in der gleichen Größenordnung vorliegen. Die Formel zur Standardisierung sieht wie folgt aus:

$$Z = \frac{X - \mu}{\sigma}$$

- Z = standardnormalverteilte Zufallsvariable
- X = Zufallsvariable
- μ = Erwartungswert
- σ = Standardabweichung

X ist der Wert des Merkmals, welches transformiert werden soll. μ entspricht dem Mittelwert und σ der Varianz. (Fahrmeir u. a. 2016, S. 269)

Die «scikit-learn»-Bibliothek (Pedregosa u. a. 2011b) bietet ein Paket «preprocessing» mit verschiedenen Scaler-Funktionen, um Daten zu transformieren. Darunter auch den «StandardScaler()» zum standardisieren der Daten wie oben beschrieben.

3.1.4 Equal-width Binning

Beim Binning, was so viel bedeutet wie Gruppieren, werden Merkmalswerte in n Bins mit einer Größe $\frac{Wertebereich}{n}$ aufgeteilt (Runkler 2010, S. 47). Als Beispiel kann der Minimal- und Maximalwert eines Merkmales wie folgt definiert werden: $X_{min} = 100$ und $X_{max} = 200$. Dieser soll in $n = 10$ gleich große Bins unterteilt werden. Dies ergibt für den Wertebereich eine Breite von zehn Einheiten.

$$\frac{|X_{max} - X_{min}|}{n} = \frac{|200 - 100|}{10} = 10$$

(Runkler 2010, S. 48)

Diese Funktion «cut()»wird von der «pandas»-Bibliothek (Reback u. a. 2020) bereitgestellt.

3.2 Supervised Learning Methoden

Supervised Learning, auch überwachtes Lernen, beschreibt das maschinelle Lernen mit Eingabedaten und den gewünschten Ausgabewerten um herauszufinden wie sich der vorgegebene Ausgabewert erstellen lässt (Müller und Guido 2017). Da die Daten der Patienten einem binären Klassifikationsproblem zugeordnet werden können, wird es die Klassen „Komplikation“ und „keine Komplikation“ geben. Es soll herausgefunden werden, mit welcher Wahrscheinlichkeit ein Patient eine Komplikation bekommen kann oder nicht. Für diese Aufgabe stehen folgende Algorithmen im Vordergrund:

- k-Nearest-Neighbor
- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- SVM mit Kernel

Um die Daten den Algorithmen zu übergeben, müssen diese zunächst in Trainings- und Test-Datensatz aufgeteilt werden. Für das Trainieren der Modelle wird immer ein Großteil der gesamten Daten verwendet. Hinterher wird mit dem kleineren Test-Datensatz der trainierte Algorithmus überprüft (Nguyen und Zeigermann 2017, S. 35). Hierzu werden die Standardeinstellungen von «scikit-learn» verwendet um die Daten aufzuteilen. Was bedeutet, dass 75% der Daten dem Trainings-Datensatz zugeordnet werden und 25% dem Test-Datensatz (Pedregosa u. a. 2011a, train_test_split).

Alle verwendeten Machine Learning Methoden werden von der «scikit-learn»-Bibliothek (Pedregosa u. a. 2011b) bereitgestellt. Für das Einteilen in die Trainings- und Test-Datensätze wird die «train_test_split()» Methode von «scikit-learn» verwendet, die die Daten zufällig aufteilt (Pedregosa u. a. 2011a, train_test_split).

3.2.1 k-Nearest-Neighbor

Der k-Nearest-Neighbor Algorithmus gehört zu den einfacheren implementierbaren ML Algorithmen. Er durchläuft das gesamte Trainings-Datenset und ordnet jeden Datenpunkt seiner zugehörigen Klasse zu. Danach wird das Test-Datenset an das zu trainierende Modell übergeben. Dieses überprüft jeden einzelnen Punkt auf den am nächsten liegenden Nachbarspunkt. Wird der Parameter $n_neighbors = 1$ gewählt, wird nur ein einziger Nachbarspunkt ausgewählt und aufgrund dessen zugeordneter Klasse die Vorhersage getroffen. Wird *n_neighbors* höher gewählt, werden auch mehr Nachbarn ausgewählt. Die Klasse mit den meisten Nachbarspunkten wird als Vorhersage genommen. Je höher der Parameter gewählt wird, desto mehr wird das Modell verallgemeinert. (Müller und Guido 2017, S. 36 f.)

3.2.2 Logistic Regression

Bei der Logistischen Regression soll vorhergesagt werden, wie wahrscheinlich es ist, dass das positive Ergebnis der Klassen zutreffen wird. Also, ob eine Komplikation auftreten wird (Team o. D., vgl. Logistische Regression). Für diese Vorhersage wird die Gleichung der Multiplen linearen Regression verwendet (Team o. D., vgl. Lineare Regression):

$$y = b_1 \cdot x_1 + b_2 \cdot x_2 + ... + b_k \cdot x_k + a$$

Die Gleichung y wird dann in die Logistische Funktion f eingesetzt:

$$f(y) = \frac{1}{1 + e^{-y}}$$

Diese Funktion sorgt dafür, dass nur Werte zwischen null und eins angenommen werden können. Dabei steht null für das negative Ergebnis und eins für das positive Ergebnis (Team o. D., vgl. Logistische Regression). Da die Wahrscheinlichkeit für das Zutreffen einer Komplikation mit $p(y = 1)$ ermittelt werden soll, ergibt sich folgende Gleichung:

$$p(y = 1) = \frac{1}{1 + e^{b_1 \cdot x_1 + b_2 \cdot x_2 + ... + b_k \cdot x_k + a}}$$

Anhand der Koeffizienten b dieser Gleichung kann visualisiert werden, welche Merkmale besonders starken Einfluss auf das Eintreten der beiden Klassen hat. Die Merkmale werden durch x_1 bis x_k definiert. Die Gewichtung definiert sich durch ihre Koeffizienten b_1 bis b_k. (Team o. D., vgl. Logistische Regression)

Werden die Merkmale betrachtet, steigt die Wahrscheinlichkeit, dass $p(y = 1)$ ist, wenn die zugehörigen Koeffizienten einen positiven Wert annehmen und je größer der Wert von x ist. Andersherum steigt die Wahrscheinlichkeit für $p(y = 0)$ je größer der Wert von x ist und die Koeffizienten einen negativen Wert annehmen (Kopp und Lois 2012, S. 169). Die Werte der Regressionskoeffizienten werden durch die Maximum-Likelihood-Schätzung ermittelt. (Team o. D., siehe Logistische Regression)

Die Konstante a in der Gleichung für Multiple lineare Regression, beschreibt den Schnittpunkt der Gleichung mit der y-Achse (Team o. D., siehe Lineare Regression).

Für das Logistic Regression Modell gibt es verschiedene Parameter, die eingestellt werden können, um das Ergebnis zu beeinflussen. Unter anderem kann der Solver für die Fehlerkorrektur eingestellt werden. Zudem kann der Parameter C genutzt werden, um die Stärke der Regularisierung der Koeffizienten zu beeinflussen. In dieser Arbeit werden die Solver *lbfgs* und *saga* genutzt. *lbfgs* unterstützt nur die *L2*-Regulation oder keine. Er läuft besser auf kleineren Datensätzen (Pedregosa u. a. 2011a, vgl. Logistic regression). Der *saga*-Solver hingegen unterstützt die *L1*- und *L2*-Regulation, welcher schneller für größere Datensätze läuft (Pedregosa u. a. 2011a, vgl. Logistic regression). *L1*-Regulation, auch Lasso Regression genannt, sorgt unter anderem dafür, dass einige Koeffizienten den Wert null annehmen und somit komplett aus der Wertung fallen. Dies hat den Vorteil, dass die wichtigsten Merkmale hervorgerufen werden und das Modell leichter zu interpretieren ist (Müller und Guido 2017, S. 52 f.). Die *L2*-Regulation, auch Ridge Regression genannt, sorgt dafür dass die Koeffizienten Richtung null

gedrängt werden aber nicht den Wert null annehmen können (Müller und Guido 2017, S. 49). Durch diese Einstellungen wird das Modell eingeschränkt um Overfitting zu reduzieren (Müller und Guido 2017, S. 49). Diese Regulation wird unter dem Parameter *penalty* eingestellt. Die *C*-Regulation gibt an, wie stark die Daten Regularisiert werden sollen. Je kleiner der Wert für *C*, desto stärker die Regulation, und desto näher sind die Werte für die Koeffizienten nahe null. Diese Einstellung führt leichter zu Overfitting. Je höher der Wert für *C*, desto schwächer die Regulation und desto wichtiger ist die korrekte Klassifikation der einzelnen Datenpunkte (Müller und Guido 2017, S. 57 f.).

3.2.3 Decision Tree Classifier

Decision Trees erstellen im wesentlichen eine Folge von Tests, die Ja/Nein-Fragen ähneln, und zu einer Entscheidung führen sollen. Dabei wird der Datensatz per Rekursion immer mehr aufgeteilt bis jeder Datenpunkt eindeutig einer Klasse zugeordnet werden kann. Die Wurzel des Baumes enthält den gesamten Datensatz und nimmt hier die erste Teilung vor, die auch als Test bezeichnet werden kann. Die Kanten verbinden den Test mit dem nächsten Knoten, die Antworten. Die Blätter des Baumes teilen die Antwort einer Klasse zu, indem sie die Klasse wählen, die am häufigsten in den Daten vorkommt. Je tiefer der Decision Tree geht, desto komplexer wird auch das Modell und neigt sehr schnell zum Overfitting. Das liegt daran, dass das Modell versucht jeden einzelnen Datenpunkt der richtigen Klasse zuzuordnen (Müller und Guido 2017, S. 68 ff.).

Unter dem Parameter *max_depth* kann die Tiefe des Baumes eingestellt werden. Je höher, desto komplexer das Modell, je niedriger desto mehr Verallgemeinert es (Müller und Guido 2017, S. 71 ff.).

Wird die Gewichtung der Merkmale visualisiert, so haben diese im Gegensatz zu dem Logistic Regression Modell immer positive Werte. Je stärker ein Merkmal gewichtet wird, desto früher kommt dieses Merkmal infrage, um die erste Teilung des Datensatzes vorzunehmen (Müller und Guido 2017, S. 75 f.).

3.2.4 Random Forest Classifier

Der Random Forest Klassifikator ist eine Methode, bei der mehrere ML Algorithmen miteinander verknüpft werden (Breiman 2001, S. 5). Deshalb wird er auch Ensemble Methode genannt, was zu sehr mächtigen Modellen führt (Müller und Guido 2017, S. 80). Dazu werden mehrere Decision Trees erstellt, wobei sich jeder Decision Tree etwas von anderen unterscheidet (Breiman 2001, S. 5 f.). Jeder der erstellten Decision Trees overfittet mit hoher Wahrscheinlichkeit, da er möglichst gute Vorhersagen treffen soll. Jedoch ist dies nicht weiter relevant, da der große Vorteil von dem Random Forest Algorithmus ist, dass er das Overfitting der einzelnen Decision Trees reduzieren kann. Es werden viele gut funktionierende Decision Trees erstellt, die auf unterschiedliche Weise overfitten. Durch mitteln ihrer Ergebnisse kann das Overfitting reduziert werden (Müller und Guido 2017, S. 80). Das heißt, jeder erstellte Decision Tree trifft

eine Entscheidung und die Klasse mit den meisten „Stimmen“ entscheidet über das endgültige Ergebnis der Klassifikation (Breiman 2001, S. 6).

Damit die einzelnen Decision Trees sich jeweils unterscheiden, werden zwei verschiedene Arten des Zufalls genutzt. Zum einem durch die Auswahl der Datenpunkte beim Aufbau des Decision Trees und zum anderen durch die Auswahl der zu testenden Merkmale bei jeder Teilung des Decision Trees (Müller und Guido 2017, S. 80). Wie auch bei den Modellen der einzelnen Decision Trees, kann bei Random Forest die Wichtigkeit der Merkmale angezeigt werden. Durch das Zusammenführen der Merkmalswichtigkeiten aller Decision Trees berechnet sich die des Random Forest. Zumeist ist diese genauer als die von einzelnen Decision Trees. Zudem weist er auch viel mehr Merkmalen eine Wichtigkeit ungleich null zu, als es ein einzelner Decision Tree tun würde (Müller und Guido 2017, S. 83 f.).

Unter dem Parameter *n_estimators* kann die Anzahl der Bäume eingestellt werden, unter *max_depth* die Tiefe der einzelnen Bäume (Pedregosa u. a. 2011a, RandomForestClassifier).

3.2.5 Gradient Boosting Classifier

Auch der Gradient Boosting Classifier gehört zu den Ensemble Methoden und setzt sich aus mehreren Decision Trees zusammen. Der Unterschied zum Random Forest Algorithmus liegt hier beim Aufbau des Modells. Anstatt alle Decision Trees gleichzeitig zu erstellen, wird beim Gradient Boosting Classifier jeder Decision Tree hintereinander aufgebaut und versucht die Fehler des vorherigen Decision Trees auszubessern. (Chen 2014)

Zudem gibt es kein Zufallselement wie beim Random Forest Algorithmus, stattdessen wird Pre-Pruning genutzt. Pre-Pruning bedeutet, dass der Decision Tree vereinfacht, gekürzt oder optimiert werden soll (Breslow und Aha 1997, S. 10). Pre-Pruning reduziert zudem das Overfitting der einzelnen Decision Trees. Genauer bedeutet es, dass die Decision Trees nicht zu komplex und tief werden. (Müller und Guido 2017, S. 88)

Die Idee hinter Gradient Boosting ist, dass möglichst viele einfache Modelle miteinander kombiniert werden. Jeder Decision Tree ist auf einen bestimmten Teil des Datensatzes spezialisiert und kann für diesen eine gute Vorhersage liefern. So soll die Leistung des Algorithmus mit jedem hinzugefügten Decision Tree erhöht werden. (Chen 2014; Müller und Guido 2017, S. 85)

Wie bei den Decision Trees und Random Forest Algorithmus können auch beim Gradient Boosting Classifier die Wichtigkeit der einzelnen Merkmale angezeigt werden (Müller und Guido 2017, S. 87).

Unter den Parametern kann die *learning_rate* eingestellt werden. Je höher der Wert, desto stärker wird der vorherige Decision Tree korrigiert. Dies führt zu komplexeren Modellen. Auch *n_estimators* erhöht die Komplexität des Modells. Hier kann angegeben werden, wie viele Decision Trees genutzt werden sollen. *max_depth*, gibt wie bei den anderen Tree-Algorithmen die Tiefe der einzelnen Decision Trees an (Pedregosa u. a. 2011a, GradientBoostingClassifier).

3.2.6 Support Vector Machine mit Kernel

Um die SVM mit Kernel richtig erklären zu können, muss zuerst auf die lineare SVM eingegangen werden. Zudem wird der Algorithmus der Einfachheit halber nur für binäre Klassifikationsprobleme erläutert.

Bei der linearen SVM, versucht der Algorithmus den Datensatz mit einer Trennlinie in beide Klassen zu separieren. Diese Trennungslinie wird Hyperebene genannt. Dabei soll diese Hyperebene den maximalen Abstand zum nächstgelegenen Punkt beider Klassen finden. (Grus 2019, S. 209)

Idealerweise dürfen sich beide Klassen nicht überlappen, damit eine optimale Hyperebene platziert werden kann. Abbildung 2 zeigt ein solches Beispiel. Zwischen den gestrichelten Linien verläuft eine Region, die als Trennungsband (Margin) bezeichnet werden kann. Wird eine perfekte Trennung erreicht, wird sie Hard-Margin genannt. (Nguyen und Zeigermann 2017, S. 982 f.)

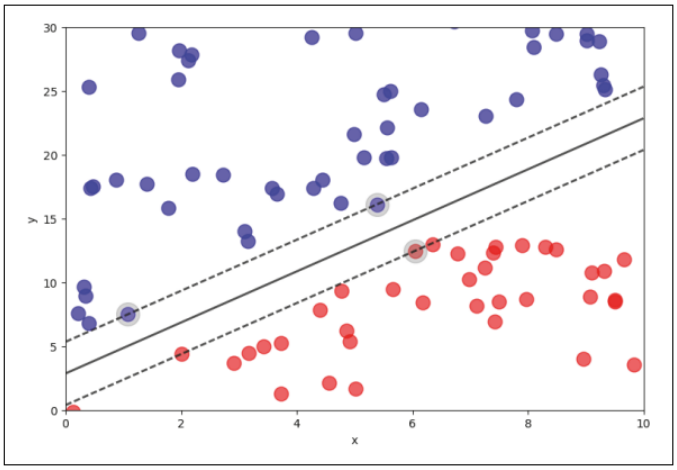


Abbildung 2: Beispiel einer optimalen Hyperebene - Hard Margin - bei einer SVM. Quelle: (Nguyen und Zeigermann 2017, S. 83)

Im Normalfall, ist es aber nicht möglich eine Hyperebene zu platzieren, die beide Klassen optimal trennt (Grus 2019, vgl. Abbildung 16-5). Für dieses Problem, gibt es die Möglichkeit der nicht-linearen Trennung (Nguyen und Zeigermann 2017, S. 86). Dazu wird der so genannte Kernel-Trick verwendet, der die Daten in einen höher dimensionalen Raum überführt (Grus 2019, S. 210). Die Idee bei der Dimensionserhöhung ist, dass die Struktur der Daten vereinfacht wird und es somit einfacher ist eine Hyperebene zu platzieren, die beide Klassen voneinander trennt (Runkler 2010, S. 94 f.; Müller und Guido 2017, vgl. Abb. 2-39). Bei der Rücktransformation der Dimensionen können auch nicht zusammenhängende Trennungsflächen entstehen (Müller und Guido 2017, vgl. Abb 2-40).

In den Standardeinstellungen der Parameter, in der «scikit-learn»-Bibliothek, ist als Kernel der RBF-Kernel, auch Gaußscher Kernel genannt, eingestellt (Pedregosa u. a. 2011a, vgl. SVC).

Der Gaußsche Kernel berechnet die Entfernung zwischen den Datenpunkten folgendermaßen:

$$k_{rbf}(x_1, x_2) = exp(-\gamma||x_1 - x_2||^2)$$

$||x_1 - x_2||$ ist der euklidische Abstand zwischen zwei Datenpunkte x_1 und x_2. γ legt die Breite des Gaußschen Kernels fest. (Müller und Guido 2017, S. 94)

Die Parameter *C* und *gamma*, können die Ergebnisse des Modells beeinflussen. Dabei reguliert der Parameter *C* die Wichtigkeit der Merkmale, ähnlich wie bei der Logistic Regression. Er sorgt bei einem kleinen Wert dafür, dass die einzelnen Datenpunkte nur einen begrenzten Einfluss auf die Veränderung der Entscheidungsgrenzen haben. Bei größeren Werten sorgt *C* dafür, dass die Entscheidungsgrenze stärker verändert wird, so dass die Datenpunkte korrekt klassifiziert werden. Die Größenordnung, ob zwei Punkte als nahe beieinanderliegend gelten, bestimmt der Parameter *gamma*. Je kleiner dieser Wert ist, desto größer ist der Radius des Gaußschen Kernels und nimmt mehr Punkte in die Wertung. Ein niedriger Wert führt zu einem komplexeren Modell. (Müller und Guido 2017, S. 95)

3.3 Unsupervised Learning Methoden

Unsupervised Learning, oder auch unüberwachtes Lernen, bedeutet, dass das System einen Datensatz ohne festgelegte Zielwerte übergeben bekommt und in diesen Eingabedaten Muster erkennen soll (Nguyen und Zeigermann 2017, S. 23). Unsupervised Learning ist schwieriger auszuwerten als Supervised Learning Methoden, da in den meisten Fällen nicht bekannt ist, wonach genau gesucht wird. Somit ist auch unklar ob, dass was der Algorithmus lernt, sinnvoll ist (Müller und Guido 2017, S. 3). Hierbei wird zwischen der Transformation des Datensatzes und dem Clusterverfahren unterschieden. Die Transformation sorgt dafür, dass die Repräsentation der Daten für Menschen oder andere Lernalgorithmen besser verständlich ist (Nguyen und Zeigermann 2017, S. 23). Das verbreitetste Anwendungsbeispiel ist die Dimensionsreduktion auf Daten, die besonders viele Merkmale besitzen. Das Endergebnis besitzt weniger Dimensionen und somit eine bessere Übersicht der Merkmale die wichtig sind (Grus 2019, S. 143). Das Clusterverfahren teilt Datensätze mit ähnlichen Elementen oder Mustern in separate Gruppen auf (Müller und Guido 2017, S. 123).

In dieser Bachelorarbeit wird ein Transformations-Algorithmus und verschiedene Clusterverfahren genutzt, um zu erkennen, ob die Algorithmen in der Lage sind, die Daten in die vorgegebenen Klassen zu unterteilen. Dafür werden folgende Algorithmen verwendet:

- PCA (Hauptkomponentenzerlegung)
- k-Means-Clustering
- Aggloremative Clustering
- DBSCAN

3.3.1 Principal Component Analysis

PCA auch Hauptkomponentenzerlegung wird verwendet, um die Dimensionen von Merkmalen zu reduzieren (Nguyen und Zeigermann 2017, S. 23). So werden die wichtigsten Zusammenhänge erfasst und besser dargestellt (Grus 2019, S. 143). Um PCA anwenden zu können, muss im vornherein der Standard Scaler verwendet werden, damit jedes Merkmal eine einheitliche Varianz besitzt (Müller und Guido 2017, S. 136).

Es existieren m Datenpunkte in in einem P-dimensionalen Raum. Damit die Hauptkomponenten gebildet werden können, wird zunächst der Mittelpunkt der Daten als Koordinatenursprung genutzt. Danach wird das Koordinatensystem so gedreht, dass die erste Koordinate in Richtung der größten Varianz der Datenpunkte zeigt (Broelemann 2006, S. 3 f.). Das ist die Richtung, in der die Datenpunkte die meisten Informationen enthalten. Diese bilden die erste Hauptkomponente. Der Vorgang kann so oft wiederholt werden wie es Merkmale gibt (Müller und Guido 2017, S. 132 f.). PCA versucht hier ursprüngliche Merkmale mit hoher Korrelation zusammenzufassen (Broelemann 2006, S. 3 f.).

Diese Methode wird auch benutzt, um Daten auf Supervised Learning Methoden vorzubereiten. In manchen Fällen führt die Reduktion der Dimensionen zu besseren Ergebnissen. (Müller und Guido 2017, S. 124)

3.3.2 k-means-Clustering

Der k-means-Clustering Algorithmus ist ein Clustering Algorithmus, der Mittelpunkte von Clustern ermittelt, die bestimmte Regionen innerhalb der Daten repräsentieren (Müller und Guido 2017, S. 158).

Dazu wird per Zufall eine entsprechende Anzahl von Mittelpunkten (Centroids) auf den Datensatz gelegt. Dafür muss vorher angegeben werden, wie viele Cluster der Algorithmus finden soll. Dementsprechend viele Centroids generiert er. Diese Punkte entsprechen mit einer sehr hohen Wahrscheinlichkeit nicht annähernd den gesuchten Mittelpunkten. Dennoch werden sie als solche behandelt. Hiernach geht der Algorithmus iterativ vor um die richtigen Mittelpunkte zu finden. Als nächstes misst der Algorithmus den Abstand jedes Datenpunktes zu den Mittelpunkten. Dem Mittelpunkt, dem der Datenpunkt am nächsten ist, wird er zugeordnet. Dies passiert mit jedem Datenpunkt, bis alle einem Mittelpunkt zugeordnet werden konnten. Anschließend werden die Mittelpunkte für beide Gruppen neu berechnet. Diese verschieben sich, da beide aus einer neu zusammengestellten Gruppe von Datenpunkten berechnet werden. Hiernach bestimmt der Algorithmus wieder den Abstand zu allen Datenpunkten und ordnet sie dem nächsten Mittelpunkt zu. Das wird so oft wiederholt, bis sich die Cluster in einer Iteration nicht mehr verändern. (Team o. D., vgl. k-Means Clusteranalyse)

In dem Parameter *n_clusters* kann angegeben werdens, wie viele Cluster gefunden werden sollen (Pedregosa u. a. 2011a, vgl. KMeans).

3.3.3 Agglomerative Clustering

Das Agglomerative Clustering ist ein Algorithmus, der damit beginnt jeden Datenpunkt als eigenes Cluster zu betrachten. Er vereinigt sukzessive die zwei ähnlichsten Cluster miteinander, bis eine Abbruchbedingung eintritt. (Müller und Guido 2017, S. 169 f.)

Der Parameter *n_clusters* ist diese Abbruchbedingung, an welche die Anzahl der Cluster angegeben wird, die erreicht werden soll. Zudem ist ein Parameter *linkage* implementiert, der drei verschiedene Ähnlichkeitsmaße, mit denen zwei Cluster miteinander verglichen werden, bereitstellt:

- ward
- average
- complete

(Pedregosa u. a. 2011a, vgl. AgglomerativeClustering)

ward ist das standardmäßig eingestellte Ähnlichkeitsmaß. Um zwei Cluster miteinander zu verbinden, wählt es die Cluster aus, bei welchen die Varianz innerhalb des neu geformten Clusters möglichst wenig ansteigt. Meistens führt diese Methode zu etwa gleich großen Clustern. Bei *average* werden immer zwei Cluster gewählt, bei denen die jeweiligen Punkte den geringsten durchschnittlichen Abstand untereinander haben. In *complete* werden die Cluster mit der kleinsten Maximaldistanz zwischen zwei Punkten miteinander verbunden. (Müller und Guido 2017, S. 170)

3.3.4 DBSCAN

DBSCAN bedeutet „density-based spatial clustering of applications with noise“ (Müller und Guido 2017, S. 174). Dieser Cluster Algorithmus ermittelt Regionen im Datensatz, wo Datenpunkte dicht beieinander liegen. Die Grundidee ist, dass dichte Regionen Cluster bilden und relativ leere Bereiche diese Cluster voneinander trennen. Punkte in einer dichten Region nennt man Kernobjekte. (Pedregosa u. a. 2011a, vgl. 2.3.7.DBSCAN)

Dazu werden zwei Parameter (*min_samples* und *eps*) bereitgestellt, um diese zu definieren. *min_samples* gibt die Mindestanzahl der Punkte an, die innerhalb eines definierten Abstandes *eps* zu einem gegebenen Punkt liegen. Liegt dieser Punkt in dem Bereich, wird er als Kernobjekt eingestuft. Wenn mehrere Kernobjekte näher als der Abstand *eps* zueinander liegen, werden diese dem gleichen Cluster zugeordnet. (Pedregosa u. a. 2011a, vgl. DBSCAN)

Zuerst wählt der Algorithmus einen beliebigen Startpunkt aus, von welchem aus er alle Punkte im Abstand *eps* oder weniger sucht. Findet er im Abstand *eps* weniger Punkte als in *min_samples* angegeben, wird dieser Punkt als Rauschen eingestuft und gehört somit keinem Cluster an. Gibt es jedoch mehr oder gleich viele Punkte wie *min_samples* im Abstand *eps*, so wird der Punkt als Kernobjekt eingestuft und erhält eine neue Clusterbezeichnung. Hiernach

werden alle Nachbarn des Punktes im Abstand *eps* abgearbeitet und werden dem Cluster zugeordnet. Ist einer der Nachbarn auch ein Kernobjekt, werden auch dessen Nachbarn auf gleiche Weise abgearbeitet. Das Cluster wächst solange, bis keine Kernobjekte mehr im Abstand *eps* gefunden werden. Danach wird der nächste Punkt ausgewählt, welcher noch nicht abgearbeitet wurde und der Prozess beginnt von vorn. (Müller und Guido 2017, S. 174 f.)

Sobald der Algorithmus durchgelaufen ist, bleiben drei Arten von Punkte übrig:

- Kernobjekte
- Punkte im Abstand *eps* zu Kernobjekten
- Rauschen

(Müller und Guido 2017, S. 175)

3.4 Performance Metrics

Um ein ML Modell zu bewerten, gibt es viele verschiedene Verfahren. Die «scikit-learn»-Bibliothek stellt in dem «metrics» Package einige Methoden zur Verfügung, um diese Berechnungen durchzuführen (Pedregosa u. a. 2011a, vgl. 3.3.2. Classification metrics).

Die bekanntesten Metrics sind der F1-Score und die Accuracy. Zusätzlich sollen die Modelle auch mit dem MCC bewertet werden, der einige Schwachstellen des F1-Score ausgleicht. (Schulte 2019)

Im Folgenden werden diese Metrics beschrieben und deren Aufbau erklärt.

3.4.1 Confusion Matrix

Die Confusion Matrix wird erstellt, um einen Überblick darüber zu erhalten, welche Ergebnisse des ML Modells richtig oder falsch vorhergesagt wurden. Dies setzt sich aus folgenden Variablen zusammen:

- richtig positiv (TP)
- falsch positiv (FP)
- falsch negativ (FN)
- richtig negativ (TN)

(Fawcett 2006, S. 862)

Aus diesen Variablen setzt sich dann die Confusion Matrix zusammen.

Confusion Matrix

	Hat Komplikation	Hat keine Komplikation	
positiv (Komplikation)	TP	FP	$\sum$ pos. Ergebnisse
negativ (keine Komplikation)	FN	TN	$\sum$ neg. Ergebnisse
	$\sum$ Komplikationen	$\sum$ keine Komplikationen	

(vgl. Fig.1. Fawcett 2006, S. 862)

Bei der Confusion Matrix sind hohe Werte auf der Diagonalen wünschenswert. Das Modell soll möglichst viele Fälle richtig klassifzieren (Schulte 2019).

3.4.2 F1-Score

Der F1-Score ist eine Testmethode, um zu bewerten, wie gut das ML Modell den positiven Fall des Problems vorhersagen kann. Um diesen Score ermitteln zu können, müssen die gesammelten Werte der Confusion Matrix genutzt werden. Aus diesen Zahlen wird zuerst Precision und Recall ermittelt:

Precision

$$P(\text{tatsächlich krank}|\text{positives Testergebnis}) = \frac{TP}{TP + FP}$$

Precision gibt von der Gesamtheit aller positiv klassifizierten Ergebnisse den Anteil der als korrekt positiv klassifizierten Ergebnisse an. (Schulte 2019)

Recall

$$P(\text{positives Testergebnis}|\text{tatsächlich krank}) = \frac{TP}{TP + FN}$$

Recall gibt die Wahrscheinlichkeit an, mit der ein positives Ergebnis korrekt als positiv klassifiziert wurde. (Schulte 2019)

Nun setzt sich der F1-Score aus Precision und Recall mithilfe des gewichteten harmonischen Mittels zusammen. Die Formel des gewichteten harmonischen Mittels sieht folgendermaßen aus:

$$x_{harm} = 2 \cdot \frac{a \cdot b}{a + b}$$

Daraus leitet sich der F1-Score ab:

F1-Score

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

(Broelemann 2006, S. 1 f.) Der F1-Score gewichtet Precision und Recall in dieser Formel gleich. Die allgemeine Formel des F-Scores sieht wie folgt aus:

$$F_\alpha = (1 + \alpha^2) \cdot \frac{Precision \cdot Recall}{\alpha^2 \cdot Precision + Recall}$$

(Broelemann 2006, S. 3)

Je nachdem, wie α gewählt wird, verschiebt sich die Gewichtung zwischen den beiden Variablen. Beispielsweise wird bei dem F_2-Score Recall viermal so hoch gewichtet wie Precision. Beim $F_{0.5}$-Score hingegen umgekehrt Precision viermal so hoch wie Recall. Im Normalfall wird der F1-Score genutzt, um ML Modelle zu bewerten. (Schulte 2019)

Der F1-Score kann Werte zwischen 0.0 und 1.0 annehmen, wobei hohe Werte für ein besseres Modell sprechen. Je niedriger der Wert desto schlechter ist das Modell. (Schulte 2019)

Nachteile: Der F1-Score kann verfälschte Ergebnisse liefern, wenn beide Klassen, die vorhergesagt werden können, ungleich verteilt sind. Zudem wird der *richtig negativ* Wert (TN) gar nicht in die Wertung aufgenommen. (Schulte 2019)

3.4.3 Accuracy

Der Accuracy-Score gibt an, wie viele Ergebnisse des ML Modells *richtig positiv* und *richtig negativ* vorausgesagt wurden. Das lässt sich mit folgender Formel ermitteln:

$$P(\text{richtig klassifiziert}) = \frac{\text{richtig klassifiziert}}{\text{alle Fälle}} = \frac{TP + TN}{TP + FP + TN + FN}$$

(vgl. Fig.1. Fawcett 2006, S. 862)

Aus dem Accuracy-Score kann abgeleitet werden, ob ein Modell zum Over- bzw. Underfitting tendiert. Dazu wird der Accuracy-Score des Trainings-Datensatzes und dem Test-Datensatz gegenüber gelegt. Diese werden in einer so genannten Lernkurve verglichen. (Müller und Guido 2017, S. 30 f.)

Nachteile: Der Accuracy-Score kann zuweilen zu falschen Schlüssen leiten, wenn die beiden Klassen, die vorhergesagt werden können, ungleich verteilt sind. Besteht der Datensatz z.B. überwiegend aus Daten, die nur einer Klasse zugeordnet werden, so ist die Accuracy bereits sehr hoch, wenn immer nur auf die häufigere Klasse getippt wird. Die andere Klasse kann unter Umständen gar nicht von dem Modell vorhergesagt werden. (Schulte 2019)

3.4.4 Matthews Correlation Coefficient

Alternativ zum F1-Score kann auch der MCC genutzt werden. Im Gegensatz zum F1-Score zieht der MCC alle Fälle aus der Confusion Matrix in Betracht und ist auch auf Datensätze

mit unausgeglichen Klassen anwendbar (Schulte 2019). Der MCC lässt sich wie folgt aus der Confusion Matrix berechnen:

MCC

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}$$

(Boughorbel, Jarray und El-Anbari 2017, S. 2)

Der MCC kann einen Wert zwischen -1.0 bis 1.0 annehmen, wobei -1.0 der schlechteste Wert ist und aussagt, dass die Ergebnisse immer falsch vorhergesagt werden. 1.0 hingegen ist der beste Wert und bedeutet, dass die Ergebnisse immer perfekt vorausgesagt werden. Nimmt er den Wert 0.0 an, so sind die Ergebnisse des Modells zufällig und nicht aussagekräftig. (Boughorbel, Jarray und El-Anbari 2017, S. 6)

Nachteile: Der MCC ist nicht für den Fall definiert, falls der Nenner den Wert 0 annimmt. Er kann bei solchen Fällen nicht angewendet werden. (Boughorbel, Jarray und El-Anbari 2017, S. 6)

4 Aufbereitung der Medizinischen Daten

Um Aussagen zu treffen, ob es zu einer der Komplikationen der SAB kommen wird, muss der Algorithmus erst angelernt werden. Dafür sind die Daten der Erkrankten Voraussetzung und müssen in ein entsprechendes Format gebracht werden. Wichtig ist, dass die Daten korrekt aufbereitet werden, damit aussagekräftige Ergebnisse erzielt werden können (Nguyen und Zeigermann 2017, S. 45). Wie der Aufbau der Daten ist und in welches Format sie gebracht werden, wird nachfolgend besprochen.

Die Daten der Erkrankten liegen als numerische Textwerte in mehreren CSV-Dateien in Tabellenform dar. Diese sind auf die Jahre 2014 bis 2019 aufgeteilt und enthalten alle gemessenen Vital-, Labor- und Bilanzwerte. Unter die Vitalwerte fallen Puls, Atemfrequenz, SpO2, Temperatur, Blutdruck und ICP (Intrakranieller Druck). Alle gemessenen Vitalwerte werden mit Uhrzeit angegeben. Die Laborwerte enthalten alle Labortest-Werte der Erkrankten während des Aufenthaltes im Krankenhaus. Eine weitere CSV-Datei beinhaltet die Patienten-ID mit der zugehörigen Fall-Nummer, sowie das Aufnahmedatum auf die Intensivstation der Erkrankten. Die Daten, ob bestimmte Erkrankte einer SAB, Vasospasmen oder eine verzögerte zerebrale Ischämie erlitten haben, sind in einer separaten CSV-Datei festgehalten. Hier sind zusätzliche Daten des Erkrankten zum Geschlecht, Alter, Lokalisation des Aneurysma, Angiographischer Vasospasmus, sowie schwere der Symptome der SAB vor Krankenhausaufnahme angegeben. Die Schwere des Zustandes des Erkrankten wird in mehreren Skalen festgehalten. Darunter fallen Glasgow Coma Scale (*gcs*) (Lazarovici 2005), World Federation of Neurosurgical Societies (*wfns*) (Lazarovici 2005), Fisher-Graduierung (*fisher*) (Souza, Vieira und Azevedo-Filho 2017) und Modified Rankin Scale (*mrs*) (Sano u. a. 2016) .

Da die Laborwerte einen sehr großen Umfang aufweisen, nicht regelmäßig bei allen Behandelten gemessen wurden und daher viele Lücken haben, werden diese nicht mit in die Wertung genommen. In dieser Bachelorarbeit wird der Fokus auf die Vitalwerte gelegt, weil diese mit großer Verlässlichkeit täglich alle halbe Stunde gemessen wurden. Zudem können nur die Daten aus dem Jahr 2019 benutzt werden, weil nur bei ihnen die Datei mit den Labels, ob die Erkrankten eine Komplikation erlitten hatten, vorliegt. Insgesamt gibt es Einträge für 59 Erkrankte, die benutzt werden können.

4.1 Datenaufbereitung

Damit die medizinischen Daten für die ML Algorithmen verwendet werden können, müssen diese vorher entsprechend aufbereitet werden. Dies ist Voraussetzung um aussagekräftige Ergebnisse erhalten zu können (Nguyen und Zeigermann 2017, S. 45).

Zur Vorbereitung werden die Vitalwerte aus der CSV-Datei, welche noch die Labor- und Bilanzwerte enthält, herausgefiltert. Die Vitalwerte sollen dann mit den Daten der anderen beiden CSV-Dateien in eine Tabelle integriert werden. Somit sind die Patienten-ID, Fall-Nummer, Aufnahmedatum, Vitalwerte, Uhrzeit der Messungen, die allgemeinen Angaben zum Erkrankten und die Klassen ob eine Komplikation auftrat in einer Tabelle. Anschließend sollen die

Daten bereinigt werden.

Außreisser Insbesondere Ausreißer sollen entfernt werden, die die Ergebnisse der ML Modelle verfälschen können (Grus 2019, S. 137). Hierfür hat das ärztliche Fachpersonal des UKD eine CSV-Datei angefertigt, um die obere und untere Grenze der Vitalwerte anzugeben. Diese Tabelle wird genutzt, damit jegliche Werte die unter oder oberhalb dieser Grenze liegen, herausgefiltert werden.

Format Um die Daten für die ML Methoden zu verwenden, soll jeder Betroffene eine Zeile mit allen Informationen in der Tabelle haben. Hierbei wird sich am Format des Brustkrebsdatensatzes orientiert (Pedregosa u. a. 2011a, vgl. load_breast_cancer), der als Datensatz zur Verfügung steht in der «scikit-learn»-Bibliothek. Er nimmt für jede Person alle Merkmale des Tumors in einer Zeile auf. Um das Umsetzen zu können, muss das Format der Vitalwerte aus den Daten verändert werden. Dafür soll nach und nach eine neue Tabelle erstellt werden, die am Ende an die ML Methoden übergeben werden kann. Hierzu sollen als erster Ansatz immer die Durchschnittswerte der gemessenen Vitalwerte von einem Tag als neue Spalte zu der neuen Tabelle hinzugefügt werden. Das bedeutet, dass z.B. eine Spalte zur durchschnittlichen Herzfrequenz des ersten Tages als Merkmal zum jeweiligen Betroffenen hinzugefügt wird. Das wird mit allen anderen Vitalwerten genauso wiederholt für die gesamte Zeitspanne, die der Behandelte auf der Intensivstation verbracht hat. Ein anderer Ansatz soll immer den maximal Wert eines Vitalwertes des Tages angeben. Hinterher soll untersucht werden, welche Methode bessere Ergebnisse liefert.

Transformation der Stringwerte Einige Daten liegen als Textform vor, wie das Geschlecht oder die Position des Aneurysma. Diese müssen umgeändert werden da alle Daten und Informationen als numerische Werte vorliegen müssen, damit die ML Algorithmen diese verwenden können (Grus 2019, S. 135). Da es sich bei Geschlecht und dem Aneurysma um Nominalskalen handelt, können diese mit dem OHE, wie in Abschnitt 3.1.2 erklärt, umgewandelt werden. Für das Geschlecht wird nur eine Spalte verwendet. Ist der Erkrankte männlich, so wird eine 0 eingefügt, handelt es sich um eine weibliche Erkrankte so wird eine 1 eingefügt. In diesem Fall wurde auf eine zweite Spalte verzichtet, da dies redundant wäre und keine neuen Informationen enthalten würde.

Zeitraum Die endgültige Tabelle soll einen bestimmten Zeitraum der Behandelten enthalten. Da die Komplikationen mit hoher Wahrscheinlichkeit bereits am vierten Tag der initialen Behandlung auftreten können (Rieg u. a. 2017; AMBOSS o. D.; Dreier 2015), wurde ein Zeitraum der ersten 5 Tage der Behandelten auf der Intensivstation gewählt. In den Daten der Erkrankten wird nicht erwähnt an welchem Tag die Komplikationen auftraten, somit konnte kein geeigneterer Zeitraum ermittelt werden. Jegliche Veränderungen der Daten der Erkrankten sollen somit von Anfang an beobachtet werden. Viele Behandelte bleiben jedoch weniger

als fünf Tage auf der Intensivstation, dementsprechend liegen weniger Daten für diese Personen vor und können nicht mit in die Wertung aufgenommen werden. Das liegt daran, dass keine Daten zu den Vitalwerten mehr vorliegen, sobald die behandelten Personen von der Intensivstation entlassen werden. Aus diesem Grund werden alle Datensätze zu Erkrankten die weniger als 5 Tage auf der Intensivstation lagen herausgefiltert und nicht mit in die Tabelle aufgenommen.

Format der Klassen Für die Betroffenen liegen in der originalen CSV-Datei zwei Spalten vor, ob dieser einen Vasospasmus und/oder eine verzögerte zerebrale Ischämie erlitten hatte. Da diese Arbeit einen groben Überblick schaffen soll, ob ML Methoden diese Komplikationen vorhersagen können und nicht zwischen diesen unterscheiden soll, wird eine Spalte für die Klassen erstellt. Hat der Betroffene eine oder beide Komplikationen erlitten, bekommt dieser eine 1 als Eintrag, da es die positive Klasse darstellt. Tauchte hingegen keine Komplikation auf, so wird -1 eingesetzt, das die negative Klasse darstellt.

Letzte Bereinigungen Jetzt erstellt sich eine Tabelle mit insgesamt 52 Merkmalen und zwei Kategorien. Hier stößt man jedoch auf einige Unstimmigkeiten, die noch näher erläutert werden müssen. Einige Werte können nicht aufgefüllt werden, da diese über einen Zeitraum von mehreren Tagen nicht gemessen wurden. Darunter fallen insbesondere die Atemfrequenz oder der Intrakranielle Druck die zuweilen über mehrere Tage nicht gemessen wurden. Diese Daten sollen jedoch im Datensatz bleiben, da nur wenige Daten vorliegen und nicht noch mehr gelöscht werden sollen. Wenn genug Daten vorliegen, könnte und sollte man diese aus dem Datensatz entfernen. (Burkov 2019, vgl. 5.1.5 Handhabung fehlender Merkmale)

Zudem sollte noch das Merkmal *Angiographischer Vasospasmus* mit Vorsicht behandelt werden. Er wird detektiert, wenn der Erkrankte bereits einen Vasospasmus hat. Dies passiert über das Verfahren der Digitalen Subtraktionsangiographie, die jegliche Anomalien der zerebralen Gefäße erfasst (Müller-Wille o. D.). Da diese Komplikation jedoch vorhergesagt werden soll, würde diese Information dem ML Algorithmus nicht weiter helfen. Die verschiedenen ML Methoden sollen anhand der anderen Merkmale vorhersagen können, ob die behandelte Person eine der Komplikationen erleiden wird. Zusätzlich werden Tabellenspalten entfernt die keine Informationen zu den Ergebnissen beisteuern können, wie etwa die Patienten-ID, Fall-Nummer, Aufnahmedatum und die Uhrzeiten der Messungen von den Vitalwerten.

Am Ende liegt eine Tabelle mit 51 Merkmalen und zwei Klassen vor, die für die ML Methoden verwendet werden kann.

4.2 Skalierung

Im allgemeinen wird empfohlen Daten zum Alter zu Gruppieren (Burkov 2019, vgl. 5.1.2 Binning). Aus diesem Grund wird das Alter der Erkrankten in fünf Gruppen aufgeteilt. Dies passiert per Binning, was in Abschnitt 3.1.4 erläutert wurde. Da das Alter der behandelten

Personen einer Verhältnisskala entspricht, kann diese Spalte in verschiedene Alterskategorien aufgeteilt werden.

Da ein Teil der Merkmale der Daten bereits skaliert ist, soll untersucht werden, ob die gesamten Merkmale der Daten skaliert werden sollte oder nicht. Unter die skalierten Merkmale fallen das Geschlecht und die Position des Aneurysma. Sie bewegen sich in einem Wertebereich von [0,1]. Die Merkmale, die den Ordinalskalen zugeordnet werden können, werden untersucht, ob diese zusätzlich skaliert werden sollten. Darunter fallen das Alter und die Zustände der Erkrankten, bei Einlieferung ins Krankenhaus. Aus diesem Gründen werden vier Datensätze erstellt, die alle eine eigene Benennung erhalten:

- Daten mit skalierten Vitalwerten = vitals-Datensatz
- Daten mit skalierten Vitalwerten, Alter, Zustände der Erkrankten = vca-Datensatz
- Daten mit allen skalierten Merkmalen = all-Datensatz
- unskalierter Datensatz = unscaled-Datensatz

Anschließend soll untersucht werden, welcher Datensatz am besten für welchen Algorithmus geeignet ist.

Die Wertebereiche der Merkmale unterscheiden sich stark und verteilen sich unterschiedlich. Im nachfolgenden werden aufgrund der Übersicht nur die Vitalwerte des ersten Tages visualisiert mit den restlichen Merkmalen. Die Verteilung in Abbildung 3 kann zu Verfälschungen

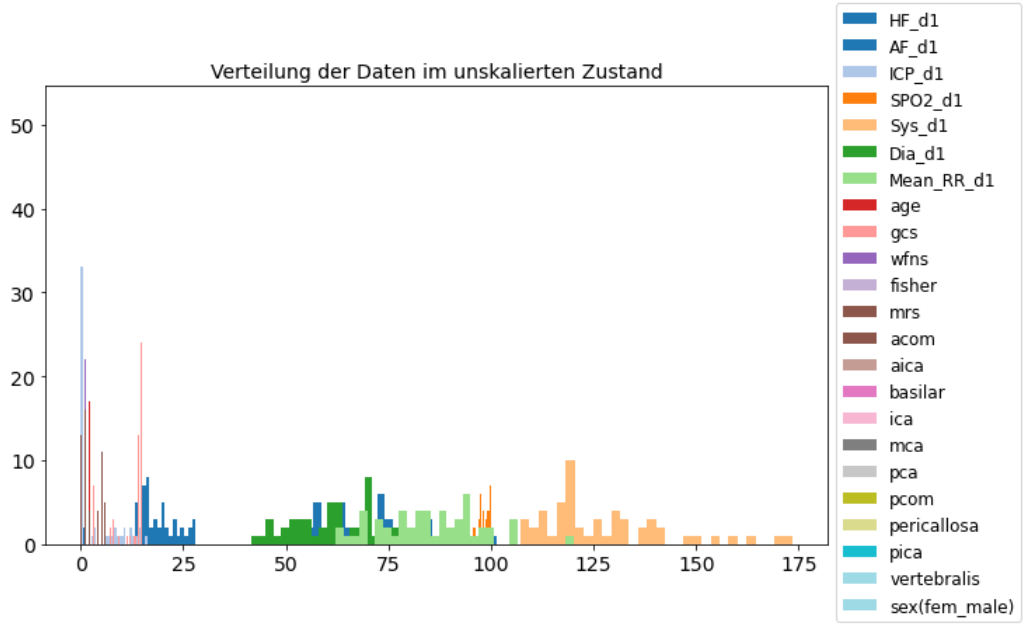


Abbildung 3: Die Verteilung der Daten mit dem unscaled-Datensatz

der Ergebnissen führen. Gerade die Vitalwerte verteilen sich auf einem höheren Wertebereich als die anderen Merkmale. (Runkler 2010, S. 31)

Bei den skalierten Vitalwerten in Abbildung 4 ist zu erkennen, wie sich die Merkmale im Wertebereich [-3, 3] um die null zentrieren. Unter den Merkmalen die nicht skaliert wurden,

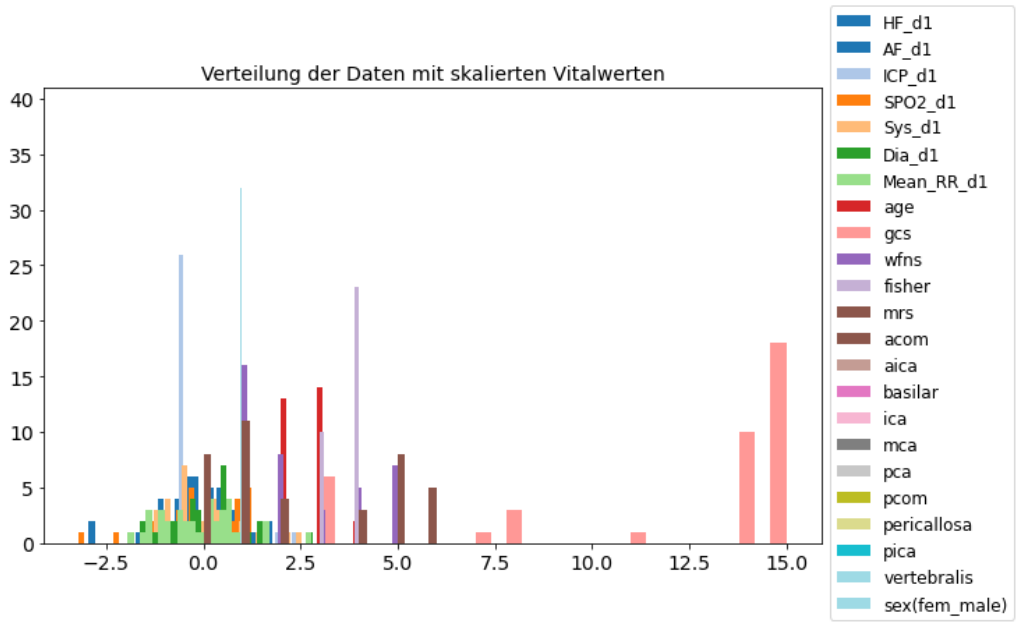


Abbildung 4: Die Verteilung der Daten mit dem vitals-Datensatz

fällt besonders *gcs* mit dem höchsten Wert 15, auf. Dies spricht dafür, dass auch die Zustände skaliert werden sollten.

In Abbildung 5 nimmt die Verteilung Werte zwischen [-3, 3] an und scheint die bisher beste Verteilung sein, da sie der Gauß-Verteilung am ähnlichsten sieht (Runkler 2010, S. 33). In Abbildung 6 ist zu erkennen, dass die Daten sich plötzlich verschieben und sich etwa im Wertebereich [-3, 6] bewegen. Insbesondere die Merkmale zur Position der Aneurysmen schlagen aus und nehmen hohe Werte an. Das spricht wiederum dagegen, die bereits skalierten Merkmale erneut zu skalieren.

Aufgrund dieser Verteilungen ist anzunehmen, dass die Skalierung für den vca-Datensatz der Betroffenen am besten für ML Algorithmen geeignet sind. Dies wird untersucht, wenn die verschiedenen Datensätze den ML Algorithmen übergeben werden.

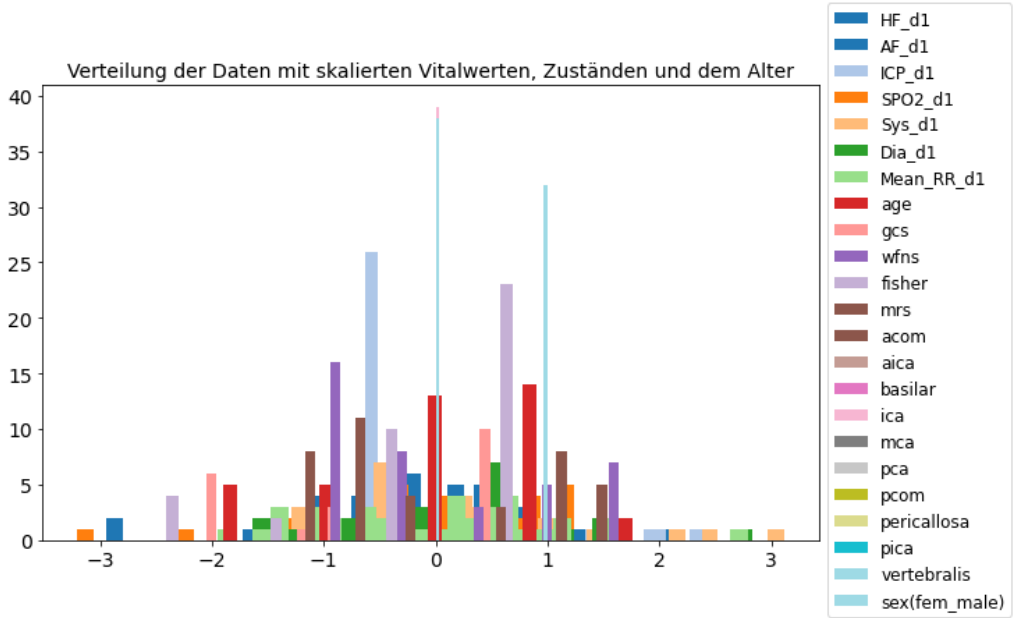


Abbildung 5: Die Verteilung der Daten mit dem vca-Datensatz

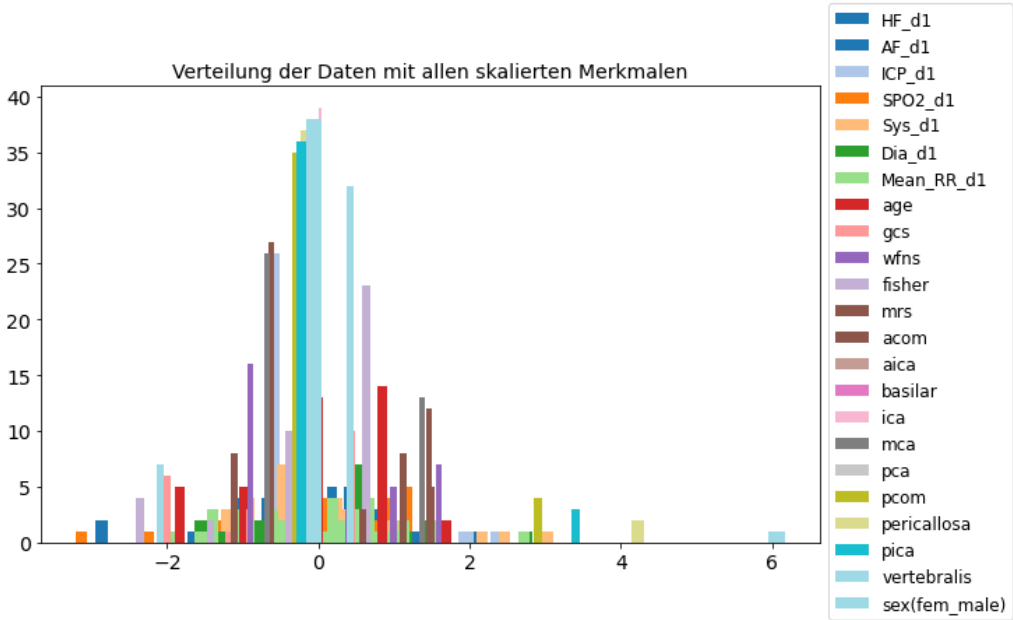


Abbildung 6: Die Verteilung der Daten mit dem all-Datensatz

5 Ergebnisse

Im folgenden Abschnitt werden die Ergebnisse der verwendeten Algorithmen analysiert und erläutert. Dabei wird anhand des F1-Score, der Accuracy und dem MCC bestimmt, wie gut die Performance eines Modells ist. Diese Metrics wurden im Abschnitt 3.4 näher erläutert. Für alle Algorithmen, wurden die Daten mit den Durchschnitts- und Maximalwerten der Vitalwerte genutzt, sowie die vier Ausführungen der skalierten Datensätze. Bei den allen ML Methoden musste festgestellt werden, dass bessere Ergebnisse mit den Durchschnittswerten der Vitalwerte der behandelten Personen erreicht werden konnte. Zudem stach Logistic Regression in ihrer Funktion besonders hervor. Sie lieferte vielversprechende Ergebnisse. Da die Modelle mit den Maximalwerten schlechter abschnitten, werden die Scores nur im Anhang einsehbar sein. Bei Unsupervised ML Algorithmen wurden insgesamt keine zufriedenstellenden Ergebnisse erreicht. Es wurden Clustering Algorithmen genutzt, um zu erkennen, ob diese die Daten gut trennen können. Zudem wurde der Transformationsalgorithmus PCA genutzt, ob durch Dimensionsreduktion der Daten bessere Ergebnisse bei den Supervised ML Methoden erzielt werden konnten. Die Daten besitzen 51 Merkmale und könnten für die Supervised Methoden besser zu verarbeiten sein, wenn deren Dimension reduziert wird (Müller und Guido 2017, S. 132). Auch hier kamen nur schwache Modelle zustande, weshalb die Ergebnisse zu den Supervised Algorithmen mit PCA auch im Anhang einzusehen sind.

5.1 Auswertung Supervised

k-nächste-Nachbarn Diese Methode wurde für insgesamt zehn Nachbarn ausgetestet. Dazu wurde eine Schleife programmiert, die von eins bis zehn hoch iteriert ist und immer den neuen Wert für den Parameter *n_neigbors* eingesetzt hat.

Für alle vier Versionen der skalierten Durchschnittswerte-Daten, waren die Ergebnisse recht ähnlich. Allgemein kann gesagt werden, dass alle Modelle bei einem gewählten Nachbarn zum Overfitting tendieren. Die Lernkurven der verschiedenen Verfahren werden in Abbildung 7 gegenübergestellt.

Im Folgenden wird nur die Lernkurve für das Modell mit den vitals-Datensatz erläutert. Bei einem Nachbarn hat der Trainings-Datensatz eine 100% Accuracy, während der Test-Datensatz einen sehr niedrigen Wert hat, was klar für Overfitting spricht. Das Modell ist zu komplex und kann die Daten des Test-Datensatzes nicht mehr richtig zuordnen, weshalb es eine Accuracy von ca. 65% erreicht. Erst ab zwei Nachbarn sinkt die Accuracy für den Trainings-Datensatz erheblich. Hier hingegen kommt es deutlich zum Underfitting, da die Accuracy für Trainings- und Test-Datensatz sehr nah beieinander liegen. Danach haben beide Accuracy Werte sich eingependelt und verändern sich kaum noch. (Müller und Guido 2017, S. 30 f.)

Beim Betrachten der Bewertungen in Abbildung 8 wird auffällig, dass alle Modelle eine schwache Performance hatten.

Ab sechs Nachbarn erreicht der F1-Score bei allen Modellen den Wert 0.0, außer bei dem unscaled-Datensatz. Dort erreicht der F1-Score den Wert 0.33 bei sieben Nachbarn. Jedoch

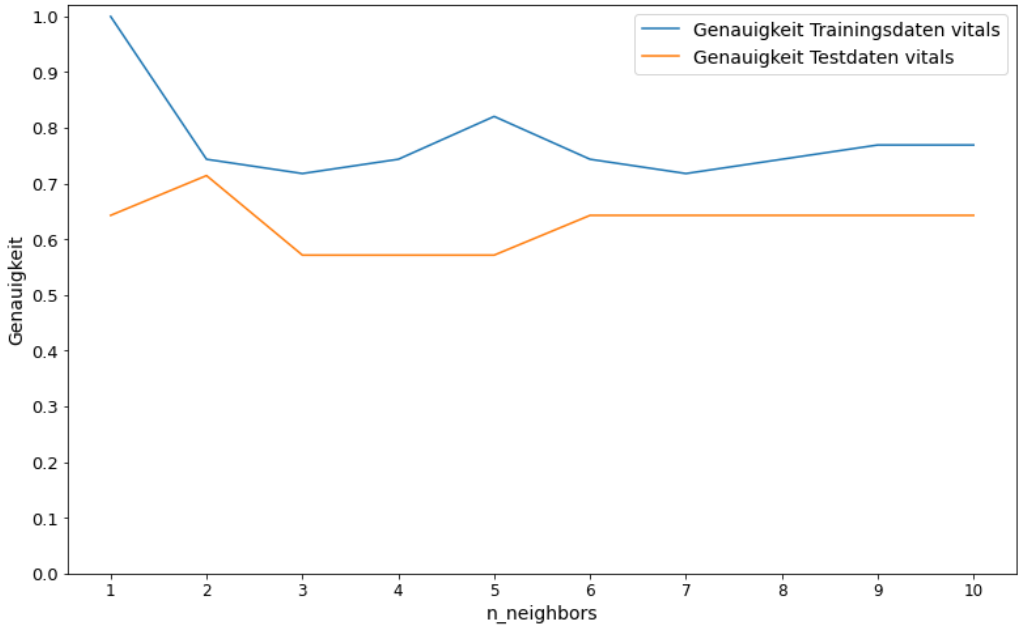


Abbildung 7: vitals-Datensatz mit 1 bis 10 Nachbarn

ist auch das kein herausragender Wert. Der MCC kann auch keine nennenswerte Ergebnisse liefern, da sich alle Werte nahe der 0.0 einpendeln.

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.615385	0.333333	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.642857	0.714286	0.571429	0.571429	0.571429	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	0.344265	0.372104	-0.025950	-0.206725	-0.206725	0.000000	0.000000	0.000000	0.000000	0.000000

(a) Scores des vitals-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.600000	0.000000	0.285714	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.714286	0.571429	0.642857	0.571429	0.571429	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	0.377778	-0.206725	0.121716	-0.206725	-0.025950	0.000000	0.000000	0.000000	0.000000	0.000000

(b) Scores des vca-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.222222	0.000000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.500000	0.571429	0.50000	0.571429	0.50000	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	-0.141421	-0.206725	-0.30429	-0.206725	-0.30429	0.000000	0.000000	0.000000	0.000000	0.000000

(c) Scores des all-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.444444	0.000000	0.250000	0.000000	0.000000	0.000000	0.333333	0.000000	0.000000	0.000000
Accuracy	0.642857	0.571429	0.571429	0.642857	0.571429	0.642857	0.714286	0.642857	0.642857	0.642857
MCC	0.188562	-0.206725	-0.025950	0.000000	-0.206725	0.000000	0.372104	0.000000	0.000000	0.000000

(d) Scores des unscaled-Datensatz

Abbildung 8: Scores der vier Modelle für den k-Nearest-Neighbor Algorithmus. Der Kopf der Tabellen zeigt die Anzahl der Nachbarn an.

Logistic Regression

Der Logistic Regression Algorithmus schnitt am besten in der Wertung ab. Es werden nur die skalierten Datensätze genutzt, da der Logistic Regression Algoritmus eine Skalierung der Daten voraussetzt. Hier werden Modelle mit dem *lbfgs*- und *saga*-Solver gegenübergestellt und einmal ein Modell mit *L1*-Regulation. Für alle drei Modelle werden die Werte für die C-Regulation auf 0.01, 1 und 100 gesetzt. In Abbildung 9 werden die Scores für den *lbfgs*-Solver aufgelistet.

Die Modelle mit den vitals- und vca-Datensatz schneiden identisch ab. Werden alle Daten skaliert, so verschlechtert sich das Ergebnis erheblich. Für eine $C = 1$ Regulation erreicht das Modell für alle Performance Metrics sehr gute Werte. Die Lernkurve in Abbildung 10 zeigt, dass für dieses Modell ein guter Punkt zwischen Over und Underfitting gefunden wurde. Mit einer Accuracy von über 95% auf den Trainings-Datensatz und einer hohen Accuracy von 85% auf den Test-Datensatz. Da die Accuracy wieder fällt, sobald C höher eingestellt wird, spricht es dafür das ein guter Punkt gefunden wurde (Müller und Guido 2017, S. 30 f.).

Die besten Modelle für den *saga*-Solver und die *L1*-Regulation haben mit dem vca-Datensatz identische Werte erreicht wie für die beiden besten Modelle des *lbfgs*-Solver. Alle diese Modelle haben einen F1-Score von 0.75 und einen MCC mit 0.7 erreicht, was sehr gut ist. Daher kann davon ausgegangen werden, dass das Modell der Logistic Regression wohl am besten mit dem vca-Datensatz zurecht kommt. Des weiteren wird noch eine Übersicht in Abbildung 11 geschaffen, auf welche Merkmale die Modelle besonders Gewicht gelegt haben. Da sich

	0.01	1.00	100.00
F1-Score	0.000000	0.750000	0.727273
Accuracy	0.642857	0.857143	0.785714
MCC	0.000000	0.700649	0.559431

(a) Scores des vitals-Datensatz

	0.01	1.00	100.00
F1-Score	0.000000	0.750000	0.727273
Accuracy	0.642857	0.857143	0.785714
MCC	0.000000	0.700649	0.559431

(b) Scores des vca-Datensatz

	0.01	1.00	100.00
F1-Score	0.000000	0.222222	0.444444
Accuracy	0.642857	0.500000	0.642857
MCC	0.000000	-0.141421	0.188562

(c) Scores des all-Datensatz

Abbildung 9: Scores der drei Logistic Regression Modelle mit *lbfgs*-Solver. Der Kopf der Tabellen zeigt die *C*-Regulation an

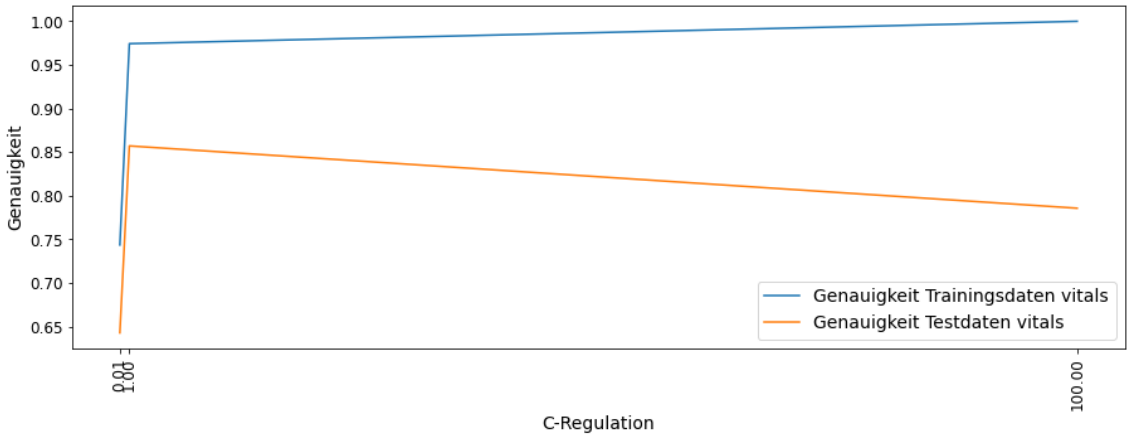
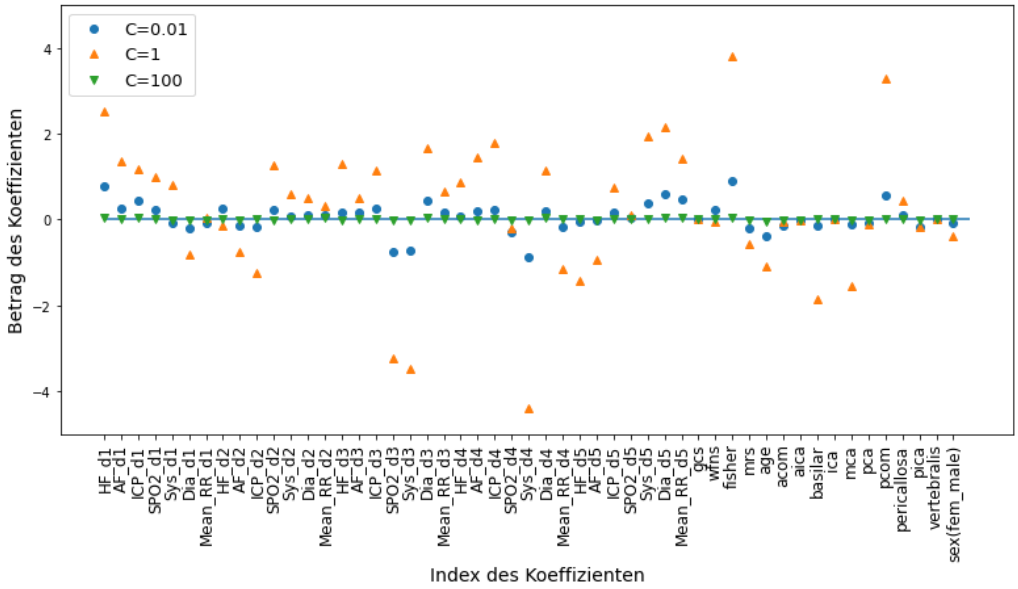


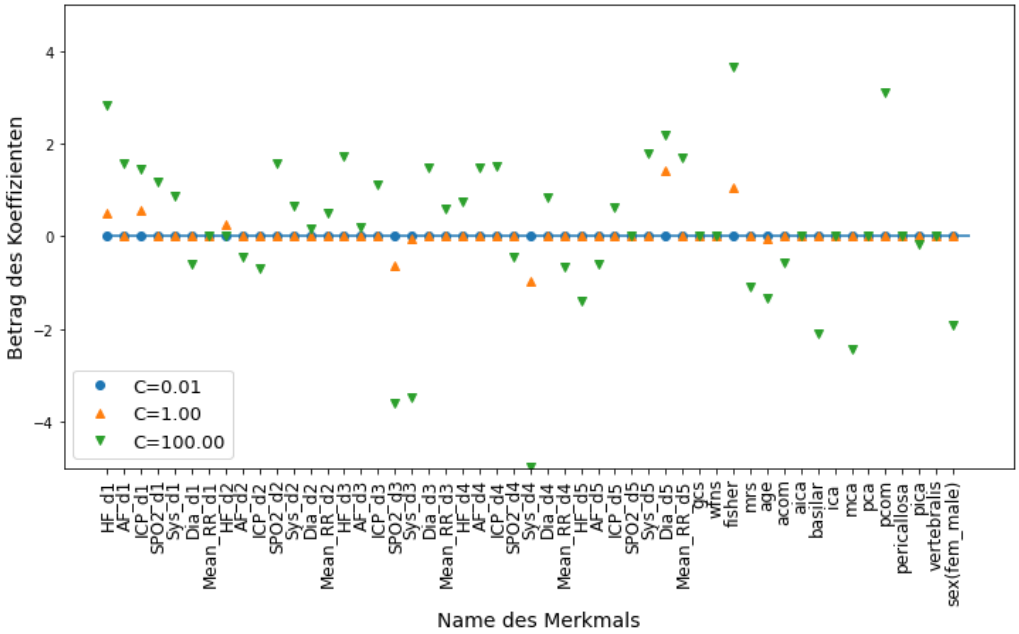
Abbildung 10: Lernkurve von Logistic Regression mit lbfgs mit dem vitals- und vca-Datensatz

die Koeffizientenverteilung des *lbfgs*- und *saga*-Solver gleichen, wird nur das Modell für den *lbfgs*-Solver gezeigt. Zudem wird des weiteren nur die Koeffizienten für C=1 betrachtet, wo die Modelle die besten Scores erzielt hatten.

In Bild (a) wird deutlich, dass die am stärksten gewichteten Merkmale (Auf der y-Achse, ab einen Wert von zwei) *HF_d1*, *Dia_d5*, *fisher*, *pcom* mit dem positiven Zutreffen einer „Komplikation“ assoziiert werden können. *Sys_d4*, *Sys_d3*, *SPO2_d3* sprechen dagegen stark für den negativen Ausgang „keine Komplikationen“ zu bekommen. Bei der *L1*-Regulation in Bild (b) weicht die Gewichtung etwas ab. Hier sprechen *HF_d1 Dia_d5*, *fisher*, *ICP_d1* sowie *HF_d2*, für einen positiven Ausgang. *Sys_d4*, *Sys_d3* und *SPO2_d3* für einen negativen Ausgang. Interessant zu beobachten ist, dass *HF_d2* bei der *L1*-Regulation für einen positiven Ausgang sprechen, während es bei der *L2*-Regulation eher dagegen spricht.



(a) Koeffizientengewichtung für vca-Datensatz mit L2-Regulation



(b) Koeffizientengewichtung für vca-Datensatz mit L1-Regulation

Abbildung 11: Koeffizientengewichtung für die besten Modelle von dem Logistic Regression Algorithmus

Random Forest

Random Forest war ein weiterer Algorithmus, welcher bedingt gut für die Daten lief. Er benötigt keine Skalierung der Daten (Nguyen und Zeigermann 2017, S. 91), was man an den Ergebnissen in Abbildung 12 gut erkennen kann. Alle Modelle liefen gleich, egal welcher skalierte Datensatz genommen wurde. Er erzielte für das beste Modell einen F1-Score von 0.57 mit einem MCC von 0.55.

Läuft Random Forest bis zur maximalen Tiefe, so overfittet das Modell, wie an dem Trainings-Accuracy Wert mit 100% zu erkennen ist. Für eine maximale Tiefe von drei, overfittet das Modell nicht mehr so stark, erreicht aber immer noch einen sehr hohen Wert für den Trainings-

	vitals	vca	all	raw
F1-Score	0.571429	0.571429	0.571429	0.571429
Train Accuracy	1.000000	1.000000	1.000000	1.000000
Test Accuracy	0.785714	0.785714	0.785714	0.785714
MCC	0.547723	0.547723	0.547723	0.547723

(a) Scores für alle Datensätze und der maximalen Tiefe mit Random Forest

	vitals	vca	all	raw
F1-Score	0.571429	0.571429	0.571429	0.571429
Train Accuracy	0.974359	0.974359	0.974359	0.974359
Test Accuracy	0.785714	0.785714	0.785714	0.785714
MCC	0.547723	0.547723	0.547723	0.547723

(b) Scores für alle Datensätze und einer Tiefe von drei mit Random Forest

	vitals	vca	all	raw
F1-Score	0.000000	0.000000	0.000000	0.000000
Train Accuracy	0.871795	0.871795	0.871795	0.871795
Test Accuracy	0.642857	0.642857	0.642857	0.642857
MCC	0.000000	0.000000	0.000000	0.000000

(c) Scores für alle Datensätze und einer Tiefe von zwei mit Random Forest

Abbildung 12: Scores der drei Modelle von Random-Forest mit unterschiedlichen Tiefen. Der Kopf der Tabellen zeigt die Datensätze an, die benutzt wurden.

Accuracy Wert. Allerdings verändert sich die Test-Accuracy nicht. Würde *max_depth*=2 gewählt, so verschlechtert sich das Modell sehr stark. Die Test-Accuracy fällt außerdem was darauf hinweist, dass eine Tiefe von drei das beste Modell ergibt (Müller und Guido 2017, S. 30 f.).

Bei Betrachtung der Gewichtung der Merkmale in Abbildung 13 wird deutlich, dass ähnliche Merkmale wie bei der Logistic Regression gewählt wurden. Die Merkmale *Mean_RR_d5*, *Dia_d5*, *Sys_d5*, *Sys_d4*, *Dia_d3* sowie *HF_d3* liegen alle über einer Gewichtung von 0.4. Besonders das Merkmal *Dia_d5* fällt auf. Ihm wird die höchste Wichtigkeit zugeschrieben mit einem Wert von 0.10. Das bedeutet, dass der Datensatz anhand diesen Merkmals schon gut aufgeteilt werden kann (Müller und Guido 2017, S. 75).

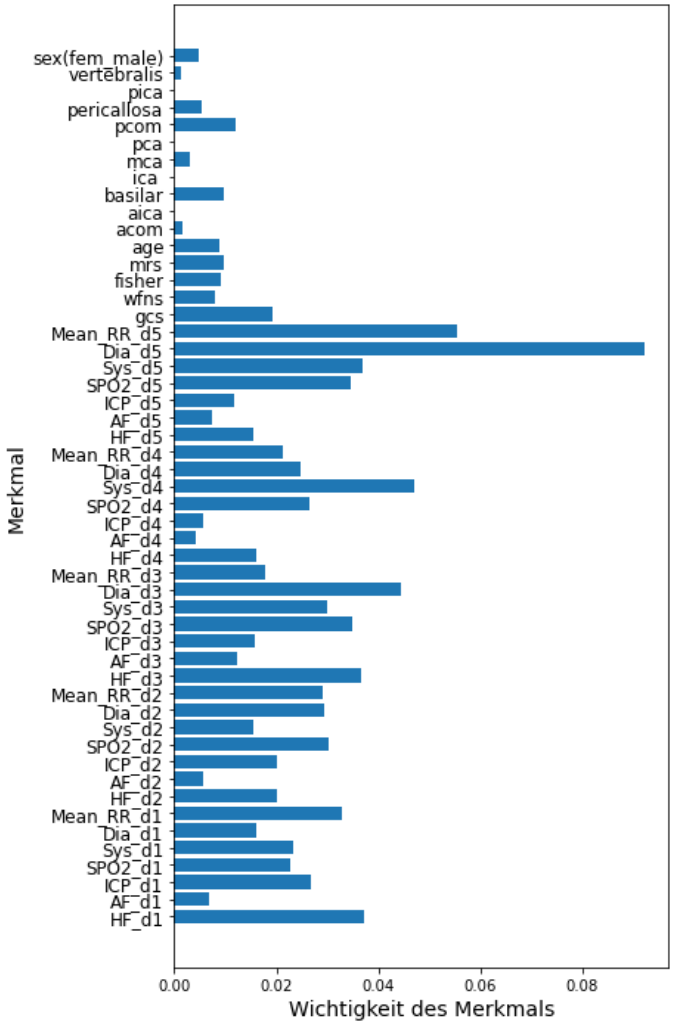


Abbildung 13: Gewichtung der Merkmale beim Random Forest Algorithmus für alle Datensätze

Gradient Boosting Classifier

Der Gradient Boosting Classifier erhält nur für die gewählten Parameter *learning_rate*=0.01 und *max_depth* = 1 gute Ergebnisse. Auch hier ist es nicht von Belang, welcher Datensatz benutzt wurde, da alle die gleichen Ergebnisse erreichten. Dies hängt wieder mit der Tatsache zusammen, dass Decision Trees keine Skalierung benötigen (Nguyen und Zeigermann 2017, S. 91). Die Anzahl der Decision Trees in *n_estimators* mit 100 oder 150 beeinflussen das Ergebnis nicht. Genau wie der Random Forest Algorithmus erhält auch das beste Modell für den Gradient Boosting Classifier einen F1-Score von 0.57 und einen MCC von 0.55. Das ist in Abbildung 14 einsehbar.

	m=3, l=0.1	m=1, l=0.1	m=3, l=0.01	m=1, l=0.01
F1-Score	0.400000	0.333333	0.400000	0.571429
Train Accuracy	1.000000	1.000000	1.000000	0.871795
Test Accuracy	0.571429	0.714286	0.571429	0.785714
MCC	0.066667	0.372104	0.066667	0.547723

Abbildung 14: Scores des Gradient Boosting Classifier Algorithmus für alle Datensätze

Wird die Gewichtung der Merkmale für das beste Modell in Abbildung 15 betrachtet, ist zu erkennen, dass vier Merkmale ausgewählt wurden. Wieder ist das *Dia_d5* Merkmal besonders wichtig.

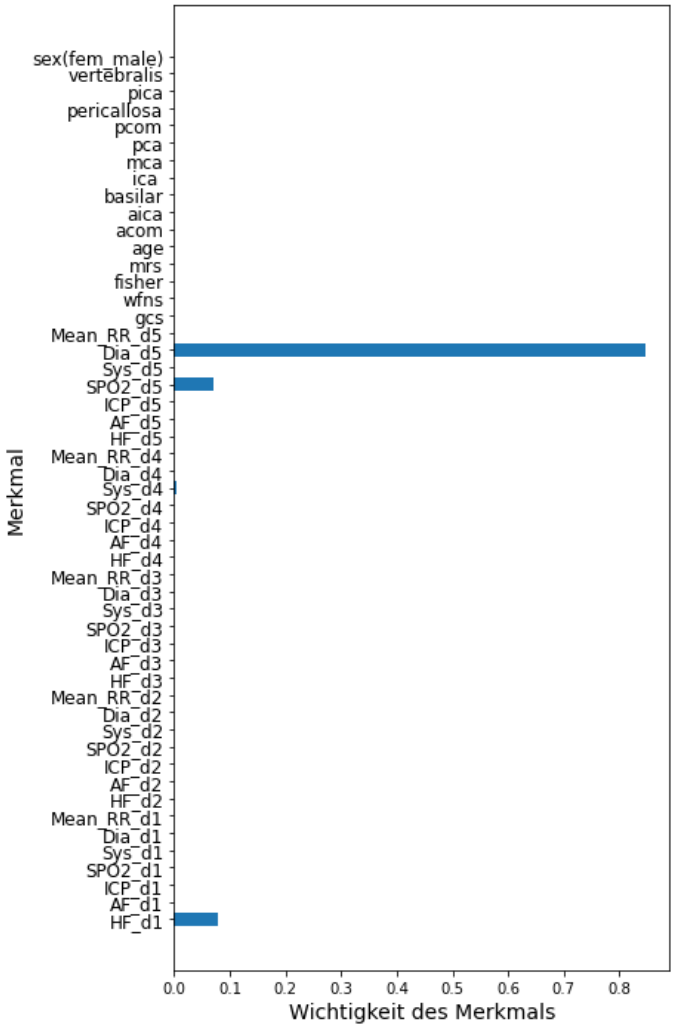


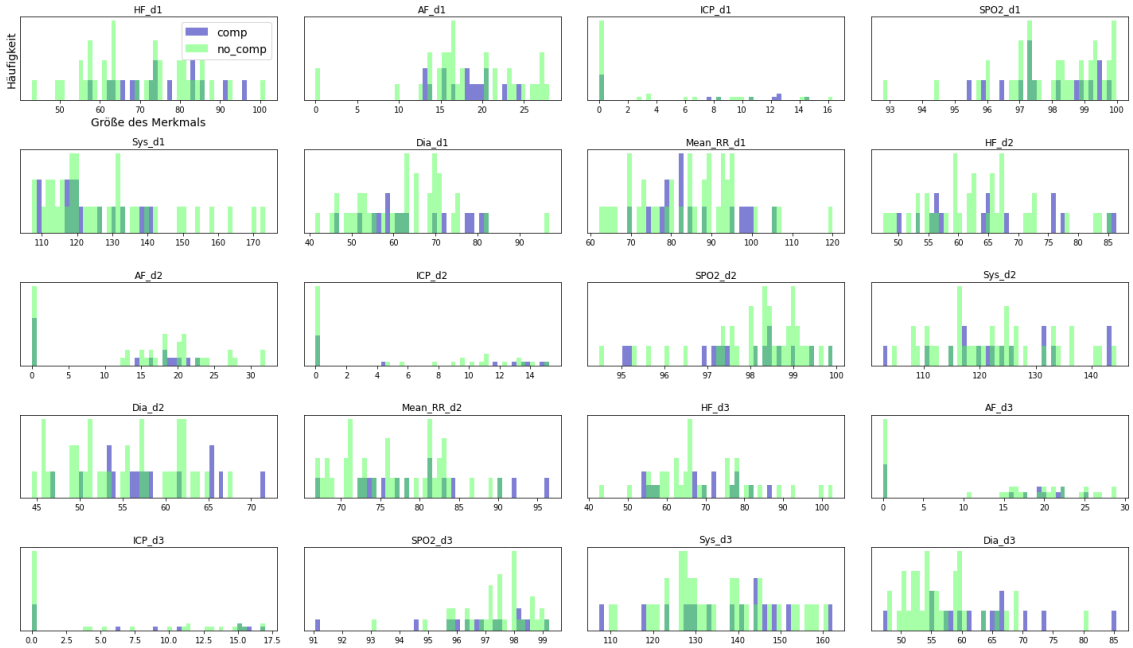
Abbildung 15: Gewichtung der Merkmale beim Gradient Boosting Classifier Algorithmus für alle Datensätze

Es sollte noch erwähnt werden, dass der Decision Tree Algorithmus für eine *max_depth*=2, die gleichen Ergebnisse erreichte wie Random Forest oder der Gradient Boosting Classifier. Die Scores sind im Anhang einzusehen, da die Ergebnisse keine neuen Informationen liefern können. Für die SVM liefen die Modelle nicht gut, weshalb hier nicht weiter auf die Ergebnisse eingegangen wird. Auch diese Scores werden im Anhang einzusehen sein. Abschließend kann festgehalten werden, dass das Logistic Regression am besten abschnitt. Die Ergebnisse der Tree Algorithmen waren zwar nicht so vielversprechend wie das der Logistic Regression, konnte jedoch einen guten Vergleich ziehen, welche Merkmale besonders wichtig sind.

5.2 Auswertung Unsupervised

Zuerst wurde eine Übersicht der Merkmale mit dem unscaled-Datensatz erstellt, um zu beobachten, ob sich die einzelnen Merkmale in beide Klassen voneinander differenzieren. Die

Übersicht findet sich in Abbildung 16, die in drei Teile aufgeteilt ist. An dieser Übersicht ist zu erkennen, dass sich keines der Merkmale richtig in beide Klassen voneinander gruppieren lassen.



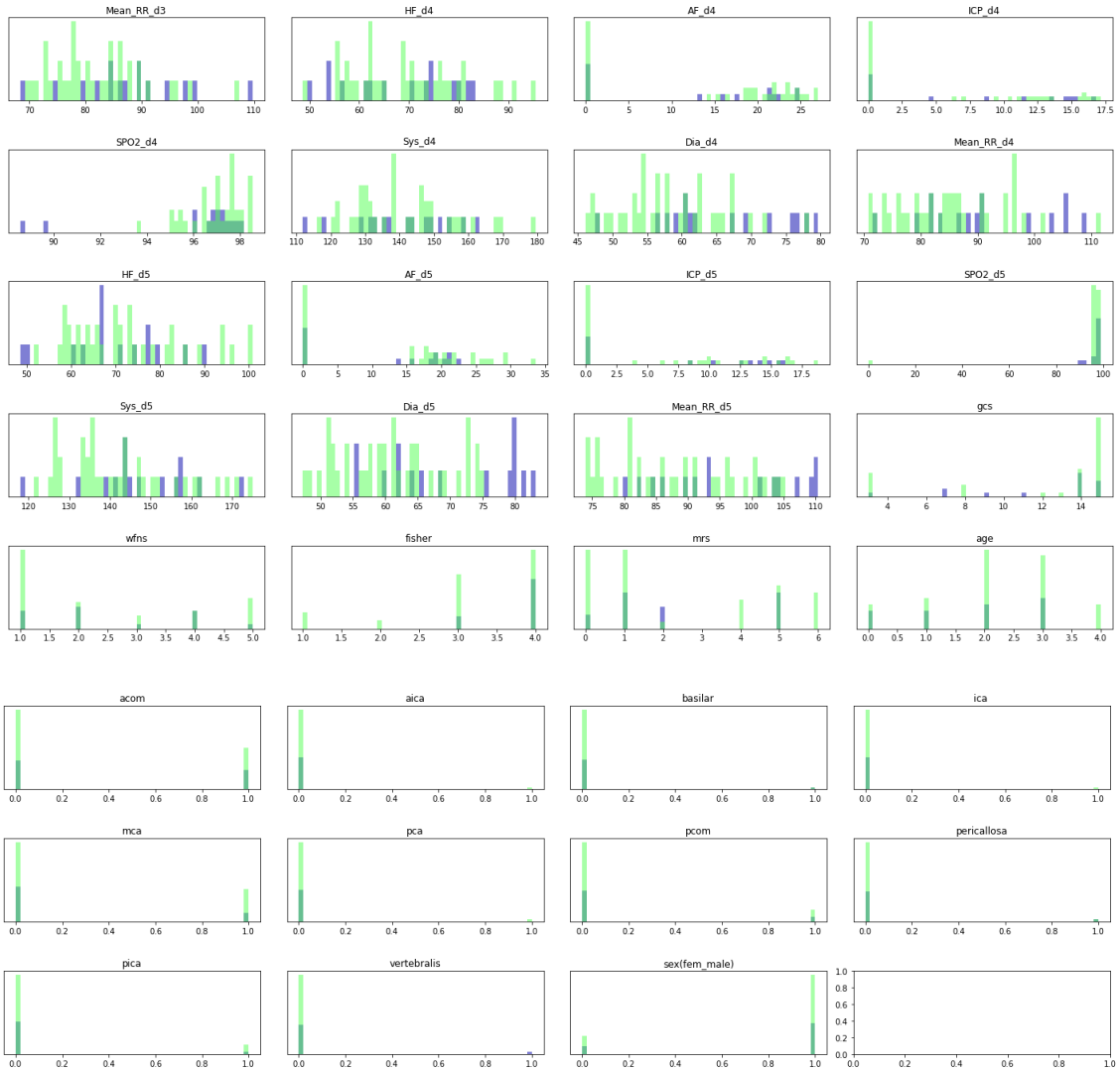
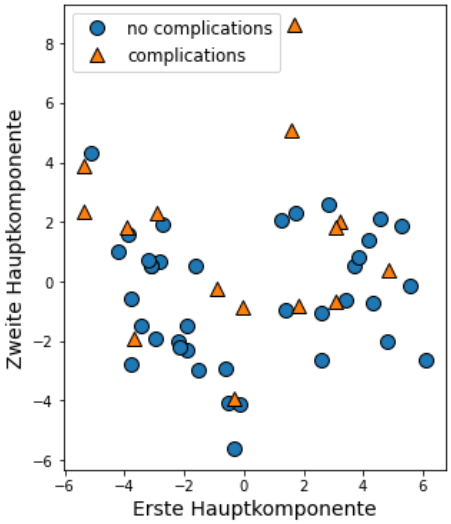


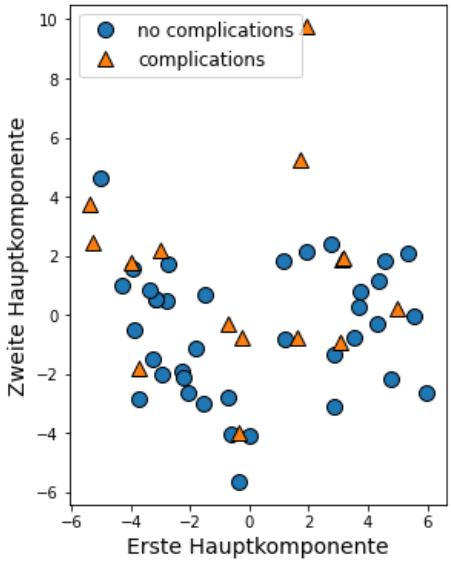
Abbildung 16: Übersicht der einzelnen Merkmale vom unscaled-Datensatz. Die beiden Bezeichnungen in der Legende stehen für Komplikation (comp) und keine Komplikation (no_comp).

PCA

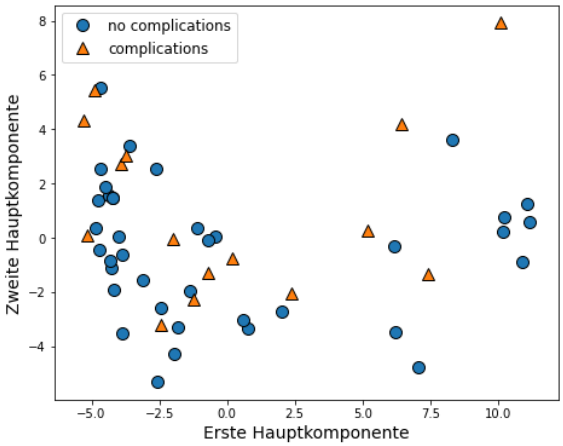
Da es nur zwei Klassen gibt, wurde $n_components = 2$ angegeben, um zu erkennen, ob sich die Hauptkomponenten dementsprechend auseinander differenzieren. Nachdem PCA die Hauptkomponenten gebildet hatte, wurde dies anhand der Klassenbezeichnungen der Datenpunkte in Abbildung 17 visualisiert. Wie gut zu erkennen ist, sind die Datenpunkte zerstreut und bilden bei keinem der skalierten Datensätze richtige Gruppen. Das würde auch erklären, warum PCA vor den Supervised ML Methoden eingesetzt, keine besseren Ergebnisse liefern konnte.



(a) Hauptkomponenten für vca-Datensatz mit PCA



(b) Hauptkomponenten für all-Datensatz mit PCA



(c) Hauptkomponenten für vitals-Datensatz mit PCA

Abbildung 17: Hauptkomponenten für die verschieden skalierten Datensätze mit PCA

k-means-Clustering

Dieser Algorithmus sollte die Merkmale der Daten in $n_clusters = 2$ aufteilen, da auch nur zwei Klassen zur Vorhersage existieren. Auf Abbildung 18 ist zu erkennen, dass die Cluster sich nicht richtig voneinander trennen und auch die Klassenzuteilung eher willkürlich ist. Links sind die Cluster abgebildet, die der k-means-Clustering Algorithmus aus den Daten erstellt hat. Die Dreiecke markieren die Mittelpunkte, die der Algorithmus gefunden hat. Auf den rechten Bildern, ist immer die eigentliche Klassenzugehörigkeit der Punkte gezeigt. Auf den linken Bildern ist zu erkennen, dass der Algorithmus die positive Klasse nach links unten zentriert. Die negative Klasse eher mittig oben des Koordinatensystems. Bei der eigentlichen Klassenverteilung auf den rechten Bildern zentriert sich die positive Klasse eher mittig und die negative Klasse ist im oberen Teil des Koordinatensystems verstreut. Hieraus lässt sich ableiten, dass der k-means-Clustering Algorithmus nicht gut auf die Daten anzuwenden ist.

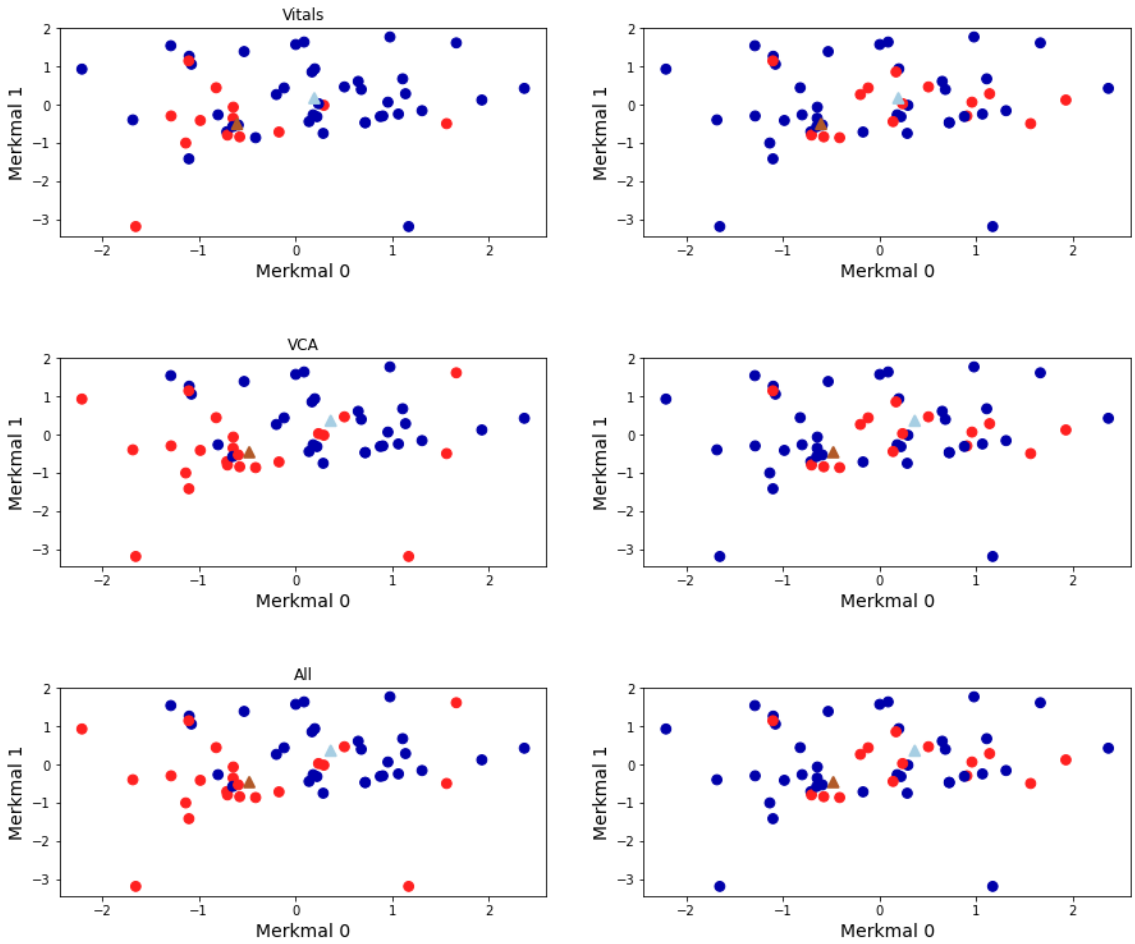


Abbildung 18: Plots der verschieden skalierten Datensätze für den k-means-Clustering Algorithmus. Die roten Punkte stellen die Merkmale dar, die der positiven Klasse angehören. Die blauen Punkte gehören der negativen Klasse an.

Agglomerative Clustering

Auch bei diesem Algorithmus wurden keine guten Differenzierungen erzielt. Es wurden *n_clusters* = 2 gewählt, wie beim k-means-Clustering Algorithmus. Die Abbildung 19 zeigt, dass auch hier keine klare Differenzierung erreicht wurde. Wie beim k-means-Clustering Algorithmus sind links die Bilder wie der Algorithmus die Cluster erstellt hat und rechts der Plot mit den eigentlichen Klassenzugehörigkeiten. Ähnlich wie der k-means-Clustering ordnet der Agglomerative Clustering Algorithmus die positive Klasse linksseitig des Koordinatensystems und die negative Klasse nach rechts oben.

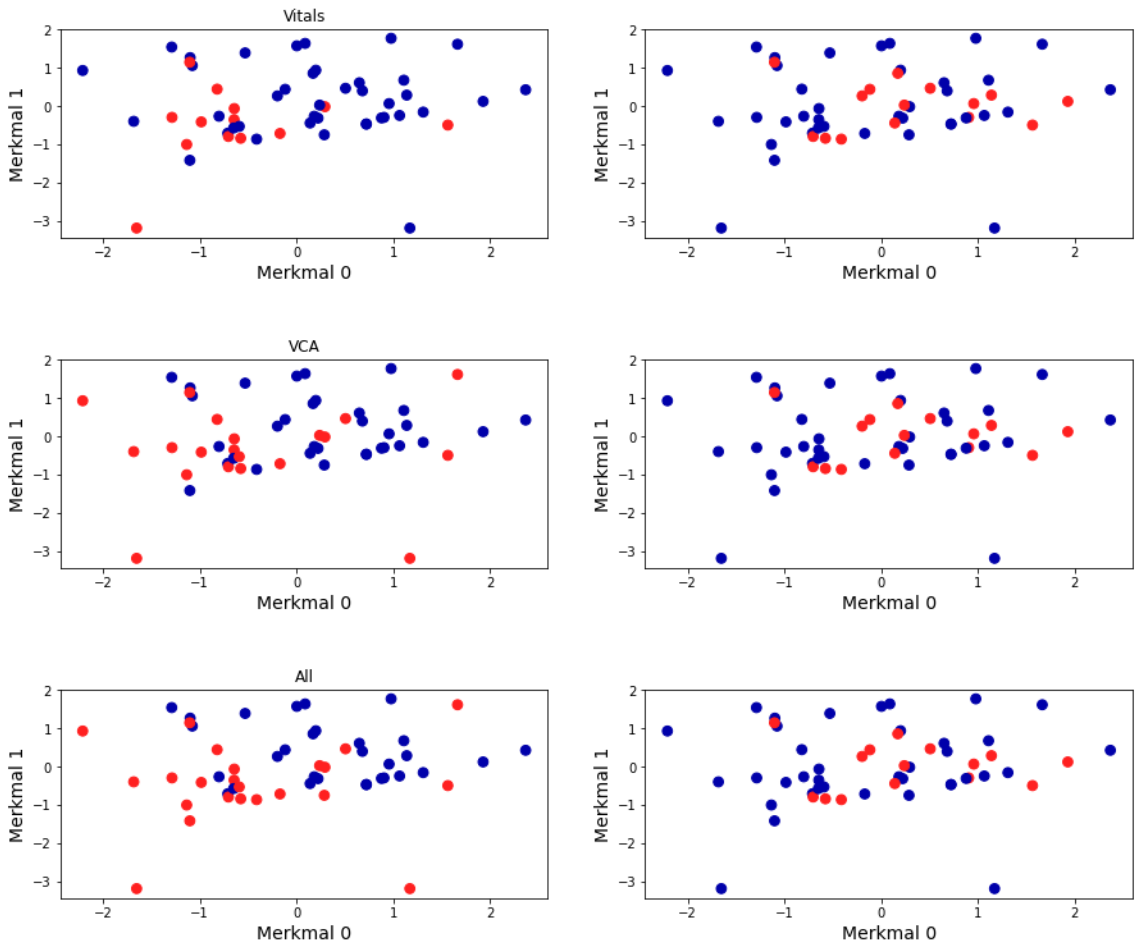


Abbildung 19: Plots der verschieden skalierten Datensätze für den Agglomerative Clustering Algorithmus. Die roten Punkte stellen die Merkmale dar, die der positiven Klasse angehören. Die blauen Punkte gehören der negativen Klasse an.

DBSCAN

Für den DBSCAN Algorithmus wurden die Parameter $eps = 0.5$ und $min_samples = 2$ gewählt. Für die Standardeinstellungen der Parameter wäre nur ein Cluster entstanden. Mit den erwähnten Parameter Einstellungen wurden zwei Cluster erstellt. Für alle skalierten Datensätze, sehen die Cluster identisch aus. Jedoch, wenn die Labels für die einzelnen Cluster ausgegeben werden, ist zu erkennen, dass der Algorithmus alle Merkmale bis auf zwei als Rauschen einstuft. Das bedeutet, dass nur zwei Datenpunkte einem Cluster zugeordnet wurden, während die anderen keinem Cluster angehören. Aus diesem Grund hat dieser Algorithmus am schlechtesten abgeschnitten. In Abbildung 20 ist zu sehen, das diese Aufteilung wenig Aussagekraft hat, da nur zwei Datenpunkte einem Cluster angehören.

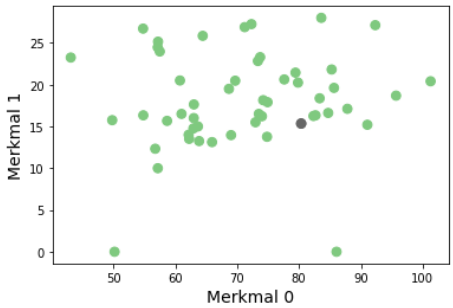


Abbildung 20: Plot des DBSCAN Algorithmus für alle skalierten Datensätze

6 Diskussion & Fazit

6.1 Diskussion

Im Zuge dieser Bachelorarbeit sollte ein Überblick erstellt werden, wie gut ML Methoden mit den Daten der Betroffenen arbeiten. In diesem Abschnitt soll diskutiert werden, welche Verbesserungen möglich wären und wie die ML Methoden weiterentwickelt werden könnten.

Von Nachteil war, dass nur Daten von 59 Erkrankten benutzt werden konnten. Wenn mehr Daten zu behandelten Personen vorliegen und den Datensatz vergrößern, würde es die Wahrscheinlichkeit für ein Overfitting reduzieren und die Vorhersagen verbessern (Müller und Guido 2017, S. 31). Sobald genug Daten gesammelt werden, könnten Datensätze von behandelten Personen mit fehlenden Messwerten über mehrere Tage entfernt werden. Diese Werte sind nicht von Vorteil, da sie keine Informationen für die ML Methoden enthalten mit einem Wert von 0. Weil allerdings nur so wenige Daten benutzt werden konnten, wurde darauf verzichtet den Datensatz weiter zu dezimieren.

Ein anderer Ansatz wäre noch, die Zeitspanne zu verschieben. Anstatt von Tag eins bis fünf der Behandlung auf der Intensivstation zu verwenden, von Tag zwei oder drei starten. Da die Komplikationen im Durchschnitt bereits an Tag vier einsetzen können (Rieg u. a. 2017; AMBOSS o. D.; Dreier 2015), könnte ein Zeitraum angeschaut werden, der zentrierter über Tag vier liegt. Bei den Daten war nicht angegeben, wann einer der behandelten Personen eine Komplikation erlitten hatte, somit konnte kein individuelles Zeitfenster ermittelt werden. Auch hier könnten die Ergebnisse verbessert werden, wenn immer beispielsweise die fünf Tage vor Eintreten des Ereignisses in die Wertung genommen werden würden.

Beim Standardisieren der Daten wurde nur der Standard Scaler genutzt, um die Daten auf bestimmte ML Methoden vorzubereiten. Jedoch gibt es noch andere Funktionen, die Daten auf unterschiedliche Weise skalieren. Unter anderem der «MinMax Scaler()», «Robust Scaler()»und «Normalizer()» (Müller und Guido 2017, S. 124 ff.). Bei der Recherche, welcher Scaler am besten benutzt werden sollte, wurde durch mehrere Quellen ersichtlich, dass neben dem Standard Scaler auch der MinMax Scaler oft zur Transformation der Daten genutzt wird (Pedregosa u. a. 2011a, vgl. 6.3. Preprocessing data; Müller und Guido 2017, S. 125 f.; Runkler 2010, S. 31 f.). Insbesondere bei der SVM könnte dieser Scaler die Ergebnisse verbessern

(Müller und Guido 2017, vgl. 130 ff.). Aber auch auf das Logistic Regression Modell, welches bereits sehr gut läuft, könnte dieser Scaler nochmals angewendet werden.

Das Kreuzvalidierungsverfahren, auch k-fold Cross Validation genannt, ist eine Methode, um den benutzten Datensatz wiederholt in verschiedene Gruppen aufzuteilen und mehrere Modelle zu trainieren. Somit ist diese Methode effektiver, als den Datensatz einmal in Trainings- und Test-Datensatz aufzuteilen. Dabei werden die Daten in eine gewünschte Anzahl von gleich großen Teilen, auch Folds genannt, aufgeteilt. Wird der Datensatz beispielsweise in fünf Folds aufgeteilt, wird zuerst Fold eins als Test-Datensatz verwendet und die restlichen vier zum trainieren des Modells. Beim nächsten Durchlauf wird Fold zwei als Test-Datensatz verwendet und die restlichen als Trainings-Datensatz. Dies wird dann mit allen Folds des Datensatzes wiederholt. Am Ende kommen stabilere und aussagekräftigere Ergebnisse zustande. Deshalb sollte, diese Methode zusätzlich implementiert werden. (Müller und Guido 2017, S. 236 f.)

Bei der Gewichtung der Merkmale ist bei Logistic Regression und allen Tree Algorithmen aufgefallen, dass insbesondere der diastolische Wert des fünften Tages stark in die Wertung fiel, ob jemand eine Komplikation erleiden wird. Auch die Herzfrequenz des ersten Tages hat bei den Modellen eine hohe Wertung erhalten, sowie der systolische Wert über mehrere Tage. Eventuell könnten daraus Rückschlüsse gezogen werden, ob die Komplikationen mit dem Herzen im Zusammenhang stehen. Da die Symptome von Vasospasmen und der verzögerten zerebralen Ischämie in den Gefäßen des Gehirns geschehen (Guide 2021; Heidelberg o. D.), könnte es durchaus Zusammenhänge geben. Jedoch müsste dies mit dem ärztlichem Fachpersonal abgeglichen werden.

Es gäbe noch die Möglichkeit die Modelle zu erweitern, um zu untersuchen, ob die einzelnen Komplikationen in der Vorhersage unterschieden werden können. Es könnte also ein Modell mit vier Möglichen Ausgängen erstellt werden: „Vasospasmus“ , „verzögerte zerebrale Ischämie“ , „Vasospasmus und verzögerte zerebrale Ischämie“ und „keine Komplikationen“. Vielleicht lassen sich so neue Muster erkennen, welche Merkmale für welche Komplikation sprechen.

6.2 Fazit

Insgesamt wurden bessere Ergebnisse mit den wenigen Daten erzielt, als zuerst angenommen. Zudem wurde ein Modell gefunden, welches ausbaufähig ist und das medizinische Fachpersonal in der Zukunft ernsthaft unterstützen könnte. Mit weiteren Anpassungen und mehr Daten wäre es möglich, frühzeitig eine Komplikation vorherzusagen. Zudem konnte festgestellt werden, dass kategorische Merkmale die einer Nominalskala angehören nicht mehr standardisiert werden sollten. Insbesondere das Modell der Logistischen Regression zeigte, dass die Performance fällt, sobald alle Merkmale skaliert werden (siehe Abschnitt 5.1).

Bereitstellung des Codes:
Der Code zu dieser Arbeit ist auf https://gitlab.cs.uni-duesseldorf.de/albi/albi-students/neuro_ai_viz einzusehen.

Literatur

[1] S. Luber und N. Litzel. „Was ist Machine Learning?“ In: (Sep. 2016). URL: https://www.bigdata-insider.de/was-ist-machine-learning-a-592092/.

[2] K. Mainzer. „Machine Learning in der Medizin: Was können Lernalgorithmen und wie sicher sind sie?“ In: *Künstliche Intelligenz im Gesundheitswesen: Entwicklungen, Beispiele und Perspektiven*. Hrsg. von M. A. Pfannstiel. Wiesbaden: Springer Fachmedien Wiesbaden, 2022. ISBN: 978-3-658-33597-7. DOI: 10.1007/978-3-658-33597-7_31. URL: https://doi.org/10.1007/978-3-658-33597-7_31.

[3] IAIS. *Machine Learning*. Abgerufen am: 2022-08-02, voller Name: Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme. URL: https://www.iais.fraunhofer.de/de/forschung/machine-learning.html.

[4] acatech. *Machine Learning in der Medizintechnik. Analyse und Handlungsempfehlungen*. Positionspapier. Mai 2020. URL: https://www.acatech.de/publikation/machine-learning-in-der-medizintechnik/.

[5] D. Ribli u. a. „Detecting and classifying lesions in mammograms with Deep Learning“. In: *Scientific Reports* 8 (März 2018). DOI: https://doi.org/10.1038/s41598-018-22437-z. URL: https://www.nature.com/articles/s41598-018-22437-z.

[6] Uniklinik Düsseldorf. *Uniklinik Düsseldorf Station MI01/02: Intensivstation*. Abgerufen am: 2022-04-09. URL: https://www.uniklinik-duesseldorf.de/patienten-besucher/klinikeninstitutezentren/klinik-fuer-neurologie/klinik/stationen/%5Callowbreak%20mi0102-intensivstation.

[7] K. Poeck und W. Hacke. *Neurologie*. 14. Springer Publishing Company, Incorporated, 2016. ISBN: 978-3-662-46891-3.

[8] A.D. Rieg u. a. „Die aneurysmatisch bedingte Subarachnoidalblutung: Update Teil 1“. In: *AI - Anästhesiologie und Intensivmedizin* (Sep. 2017). URL: https://www.ai-online.info/archiv/2017/09-2017/die-aneurysmatisch-bedingte-subarachnoidalblutung-update-teil-1.html.

[9] Leading Medicine Guide. *Hirndurchblutungsstörung - Medizinische Experten*. Abgerufen am: 2022-04-13. Jan. 2021. URL: https://www.leading-medicine-guide.de/erkrankungen/kreislauf/zerebrale-ischaemie#shortText.

[10] Universitäts Klinikum Heidelberg. *Medikamentöse und mechanische Behandlung eines Vasospasmus*. Abgerufen am: 2022-04-13. URL: https://www.klinikum.uni-heidelberg.de/verfahren/medikamentoese-und-mechanische-behandlung-eines-vasospasmus-212084.

[11] S. Kundra u. a. „Principles of neuroanesthesia in aneurysmal subarachnoid hemorrhage“. In: *Journal of Anaesthesiology Clinical Pharmacology* 30 (Juli 2014). URL: https://journals.lww.com/joacp/Fulltext/2014/30030/Principles_of_neuroanesthesia_in_aneurysmal.4.aspx.

[12] S. D'Souza. „Aneurysmal Subarachnoid Hemorrhage“. In: *Journal of Neurosurgical Anesthesiology* 27 (Juli 2015). URL: https://journals.lww.com/jnsa/fulltext/2015/07000/aneurysmal_subarachnoid_hemorrhage.6.aspx.

[13] J. P. Dreier. „Komplikationen und Folgeerkrankungen nach Schlaganfall“. In: Kapitel 9: Verzögerte ischämische Schlaganfälle nach aneurysmatischer Subarachnoidalblutung. Georg Thieme Verlag KG, Stuttgart, 2015. ISBN: 978-3-13-174021-2. DOI: 10.1055/b-003-124634. URL: https://www.thieme-connect.de/products/ebooks/lookinside/10.1055/b-0035-124187.

[14] AMBOSS. *Subarachnoidalblutung*. Abgerufen am: 2022-04-13, Zuletzt aktualisiert: 2021-12-22. URL: https://www.amboss.com/de/wissen/Subarachnoidalblutung/.

[15] C. H. Nguyen und O. Zeigermann. „Machine Learning - kurz & gut“. In: O'Reilly, Apr. 2017. ISBN: 978-3-96010-192-5.

[16] H. Gharagyozyan. *A Practical Application of Machine Learning in Medicine*. Abgerufen am: 2022-04-10. Feb. 2019. URL: https://www.macadamian.com/learn/a-practical-application-of-machine-learning-in-medicine/.

[17] F. Pedregosa u. a. *scikit-learn Machine Learning in Python*. Abgerufen am: 2022-08-05. 2011. URL: https://scikit-learn.org/stable/.

[18] L. Fahrmeir u. a. „Statistik - Der Weg zur Datenanalyse“. In: 8. Springer Spektrum Berlin, 2016. ISBN: 978-3-662-50371-3.

[19] D. M. Harris und S. L. Harris. „Digital Design and Computer Architecture“. In: 2. Elsevier, 2013. ISBN: 978-0-12-394424-5.

[20] F. Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[21] T. A. Runkler. *Data Mining - Methoden und Algorithmen intelligenter Datenanalyse*. 1. Vieweg+Teubner Verlag Wiesbaden, 2010. ISBN: 978-3-8348-9353-6.

[22] J. Reback u. a. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[23] A. C. Müller und S. Guido. *Einführung in Machine Learning mit Python - Praxiswissen Data Science*. O'Reilly Verlag, Juli 2017. ISBN: 978-3-96009-049-6.

[24] DATAtab Team. *Erste Schritte mit DATAtab*. Abgerufen am: 2022-08-02. URL: https://datatab.de/tutorial/erste-schritte-mit-datatab.

[25] J. Kopp und D. Lois. „Logistische Regressionen“. In: *Sozialwissenschaftliche Datenanalyse: Eine Einführung*. Wiesbaden: VS Verlag für Sozialwissenschaften, 2012. ISBN: 978-3-531-93258-3. URL: https://doi.org/10.1007/978-3-531-93258-3_8.

[26] L. Breiman. „Random Forests“. In: *Machine Learning* 45 (Okt. 2001). DOI: https://doi.org/10.1023/A:1010933404324. URL: https://link.springer.com/article/10.1023/A:1010933404324.

[27] T. Chen. *Introduction to Boosted Trees*. Okt. 2014. URL: https://web.njit.edu/~usman/courses/cs675_fall16/BoostedTree.pdf.

[28] L. A. Breslow und D. W. Aha. „Simplifying decision trees: A survey“. In: *The Knowledge Engineering Review* 12 (Jan. 1997). DOI: https://doi.org/10.1017.

[29] J. Grus. „Einführung in Data Science“. In: 2. O'Reilly, Nov. 2019. ISBN: 978-3-96009-123-3.

[30] Claudia Broelemann. *Proseminar: Machine Learning*. Juli 2006. URL: http://www.ra.cs.uni-tuebingen.de/lehre/ss06/pro_learning/ClaudiaBroelemann.pdf.

[31] Berthold Schulte. *Machine-Learning-Modelle bewerten-die Crux mit der Metrik*. Abgerufen am: 2022-08-06. Juli 2019. URL: https://blog.codecentric.de/2019/07/machine-learning-modelle-bewerten-die-crux-mit-der-metrik/.

[32] T. Fawcett. „An introduction to ROC analysis“. In: *Pattern Recognition Letters* 27.8 (2006), S. 861–874. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. URL: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[33] S. Boughorbel, F. Jarray und M. El-Anbari. *Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric*. Juni 2017. DOI: https://doi.org/10.1371/journal.pone.0177678. URL: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0177678&type=printable.

[34] M. Lazarovici. *Anwenderperformanz und- variabilität der Glasgow-Koma-Skala*. Dissertation. 2005. URL: https://edoc.ub.uni-muenchen.de/4524/1/lazarovici_marc.pdf.

[35] M. L. P. Souza, A. C. Vieira und H. R.C. Azevedo-Filho. *Fisher Grading Scale and Cognitive Deficits*. Sep. 2017. DOI: https://doi.org/10.1055/s-0037-1606632. URL: https://www.thieme-connect.com/products/ejournals/pdf/10.1055/s-0037-1606632.pdf.

[36] H. Sano u. a. *Modified world federation of neurosurgical societies subarachnoid hemorrhage grading system*. Aug. 2016. DOI: 10.4103/2152-7806.187491. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4982344/pdf/SNI-7-502.pdf.

[37] A. Burkov. *Machine Learning kompakt*. mitp Verlag, Juli 2019. ISBN: 978-3-95845-995-3.

[38] R. Müller-Wille. *Digitale Subtraktionsangiographie*. Zuletzt aktualisiert am: 2020-02-05. URL: https://www.compgefa.de/wissen/digitale-subtraktionsangiographie.

A Anhang

	max	3	2
F1-Score	0.400000	0.363636	0.571429
Train Accuracy	1.000000	0.974359	0.871795
Test Accuracy	0.571429	0.500000	0.785714
MCC	0.066667	-0.043033	0.547723

Abbildung 21: Scores des Decision Trees

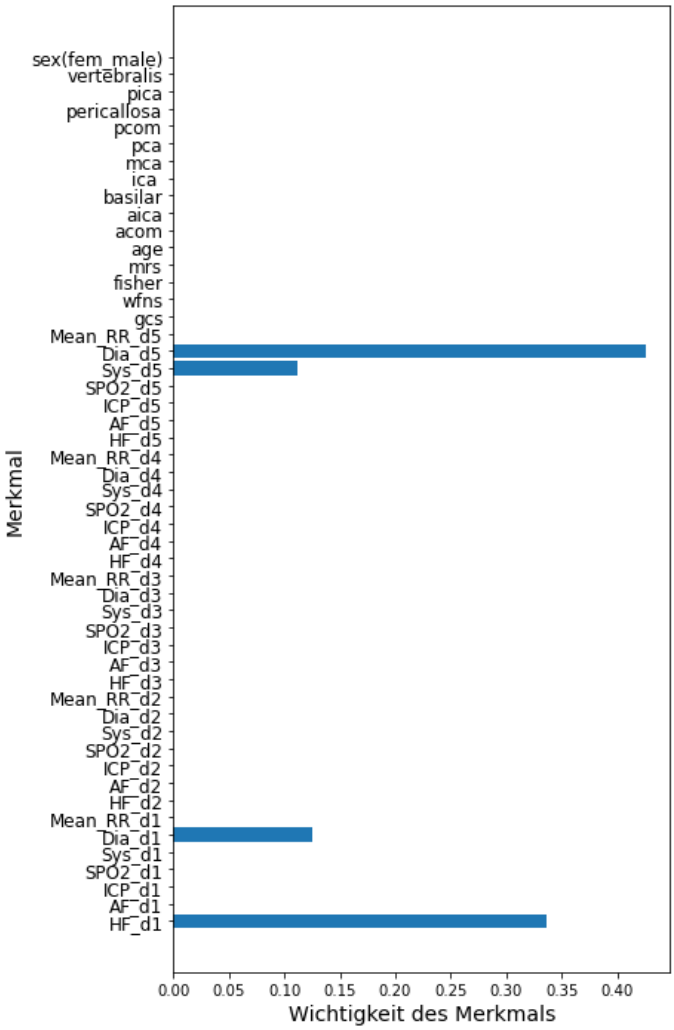


Abbildung 22: Merkmalsgewichtung von Decision Tree mit maximaler Tiefe

	C=1, g=1	C=0.01, g=1	C=100, g=1	C=1, g=100	C=1, g=1000	C=100, g=100
F1-Score	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Train Accuracy	1.000000	0.743590	1.000000	1.000000	1.000000	1.000000
Test Accuracy	0.642857	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

(a) Scores des vitals-Datensatz

	C=1, g=1	C=0.01, g=1	C=100, g=1	C=1, g=100	C=1, g=1000	C=100, g=100
F1-Score	0.000000	0.000000	0.400000	0.000000	0.000000	0.000000
Train Accuracy	1.000000	0.743590	1.000000	1.000000	1.000000	1.000000
Test Accuracy	0.642857	0.642857	0.571429	0.642857	0.642857	0.571429
MCC	0.000000	0.000000	0.066667	0.000000	0.000000	-0.206725

(b) Scores des vca-Datensatz

	C=1, g=1	C=0.01, g=1	C=100, g=1	C=1, g=100	C=1, g=1000	C=100, g=100
F1-Score	0.000000	0.000000	0.222222	0.000000	0.000000	0.000000
Train Accuracy	1.000000	0.743590	1.000000	1.000000	1.000000	1.000000
Test Accuracy	0.642857	0.642857	0.500000	0.642857	0.642857	0.571429
MCC	0.000000	0.000000	-0.141421	0.000000	0.000000	-0.206725

(c) Scores des all-Datensatz

Abbildung 23: Scores der drei Modelle für den SVM Algorithmus. Der Kopf der Tabellen zeigt die Einstellungen der Parameter an

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.333333	0.000000
Accuracy	0.50000	0.642857	0.571429	0.642857	0.428571	0.642857	0.571429	0.642857	0.714286	0.642857
MCC	-0.30429	0.000000	-0.206725	0.000000	-0.389249	0.000000	-0.206725	0.000000	0.372104	0.000000

(a) Scores des vitals-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.222222	0.000000	0.333333	0.000000	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.500000	0.642857	0.714286	0.642857	0.714286	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	-0.141421	0.000000	0.372104	0.000000	0.372104	0.000000	0.000000	0.000000	0.000000	0.000000

(b) Scores des vca-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.400000	0.000000	0.333333	0.333333	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.571429	0.642857	0.714286	0.714286	0.714286	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	0.066667	0.000000	0.372104	0.372104	0.372104	0.000000	0.000000	0.000000	0.000000	0.000000

(c) Scores des all-Datensatz

Abbildung 24: Scores der drei Modelle für den k-Nearest-Neighbor Algorithmus mit PCA. Der Kopf der Tabellen zeigt die Anzahl der Nachbarn an

	0.01	1.00	100.00
F1-Score	0.000000	0.333333	0.333333
Accuracy	0.642857	0.714286	0.714286
MCC	0.000000	0.372104	0.372104

Abbildung 25: Scores des vca- und all-Datensatz mit lbfgs- und saga-Solver für Logistic Regression

	vitals	vca	all
F1-Score	0.285714	0.600000	0.250000
Train Accuracy	0.948718	0.923077	0.948718
Test Accuracy	0.642857	0.714286	0.571429
MCC	0.121716	0.377778	-0.025950

(a) Scores des Decision Trees

	vitals	vca	all
F1-Score	0.250000	0.000000	0.285714
Train Accuracy	0.974359	0.948718	0.948718
Test Accuracy	0.571429	0.642857	0.642857
MCC	-0.025950	0.000000	0.121716

(b) Scores des Random Forest Algorithmus

	vitals	vca	all
F1-Score	0.444444	0.333333	0.444444
Train Accuracy	1.000000	1.000000	1.000000
Test Accuracy	0.642857	0.714286	0.642857
MCC	0.188562	0.372104	0.188562

(c) Scores des Gradient Boosting Algorithmus

Abbildung 26: Scores der drei Modelle für die verschiedenen Tree Algorithmen. Der Kopf der Tabellen zeigt die Parameter Einstellungen an.

	C=1, g=1	C=0.01, g=1	C=100, g=1	C=1, g=100	C=1, g=1000	C=100, g=100
F1-Score	0.000000	0.000000	0.285714	0.000000	0.000000	0.00000
Train Accuracy	0.769231	0.743590	0.897436	0.974359	0.974359	1.00000
Test Accuracy	0.642857	0.642857	0.642857	0.642857	0.642857	0.50000
MCC	0.000000	0.000000	0.121716	0.000000	0.000000	-0.30429

Abbildung 27: Scores von SVM mit vitals-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.666667	0.000000	0.250000	0.285714	0.222222	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.785714	0.642857	0.571429	0.642857	0.500000	0.571429	0.571429	0.571429	0.571429	0.571429
MCC	0.518545	0.000000	-0.025950	0.121716	-0.141421	-0.206725	-0.206725	-0.206725	-0.206725	-0.206725

(a) Scores des vitals-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.363636	0.000000	0.000000	0.000000	0.285714	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.500000	0.642857	0.571429	0.571429	0.642857	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	-0.043033	0.000000	-0.206725	-0.206725	0.121716	0.000000	0.000000	0.000000	0.000000	0.000000

(b) Scores des vca-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.428571	0.642857	0.428571	0.571429	0.50000	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	-0.389249	0.000000	-0.389249	-0.206725	-0.30429	0.000000	0.000000	0.000000	0.000000	0.000000

(c) Scores des all-Datensatz

	1	2	3	4	5	6	7	8	9	10
F1-Score	0.444444	0.333333	0.333333	0.333333	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000
Accuracy	0.642857	0.714286	0.714286	0.714286	0.714286	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	0.188562	0.372104	0.372104	0.372104	0.372104	0.000000	0.000000	0.000000	0.000000	0.000000

(d) Scores des unscaled-Datensatz

Abbildung 28: Scores der vier Modelle für den k-Nearest-Neighbor Algorithmus mit Maximal Datensatz. Der Kopf der Tabellen zeigt die Anzahl der Nachbarn an

	C=1, g=1	C=0.01, g=1	C=100, g=1	C=1, g=100	C=1, g=1000	C=100, g=100
F1-Score	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Train Accuracy	1.000000	0.743590	1.000000	1.000000	1.000000	1.000000
Test Accuracy	0.642857	0.642857	0.642857	0.642857	0.642857	0.642857
MCC	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Abbildung 29: Scores von SVM mit vitals-Datensatz mit Maximal Datensatz

	0.01	1.00	100.00
F1-Score	0.000000	0.400000	0.363636
Accuracy	0.642857	0.571429	0.500000
MCC	0.000000	0.066667	-0.043033

Abbildung 30: Scores aller Datensätze mit lbfgs-, saga-Solver und L1-Regulation für Logistic Regression mit Maximal Datensatz

	max	3	2
F1-Score	0.00000	0.000000	0.000000
Train Accuracy	1.00000	0.974359	0.923077
Test Accuracy	0.50000	0.500000	0.500000
MCC	-0.30429	-0.304290	-0.304290

(a) Scores des Decision Trees

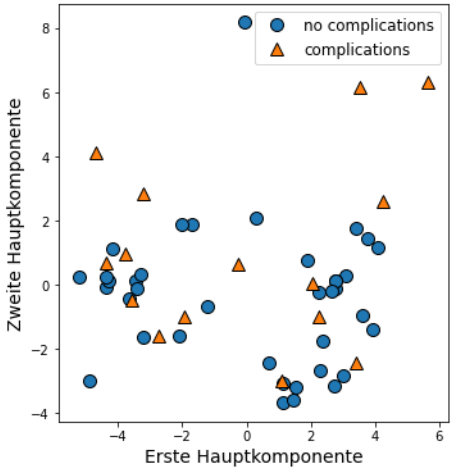
	vitals	vca	all	raw
F1-Score	0.000000	0.000000	0.000000	0.000000
Train Accuracy	1.000000	1.000000	1.000000	1.000000
Test Accuracy	0.642857	0.642857	0.642857	0.642857
MCC	0.000000	0.000000	0.000000	0.000000

(b) Scores des Random Forest Algorithmus

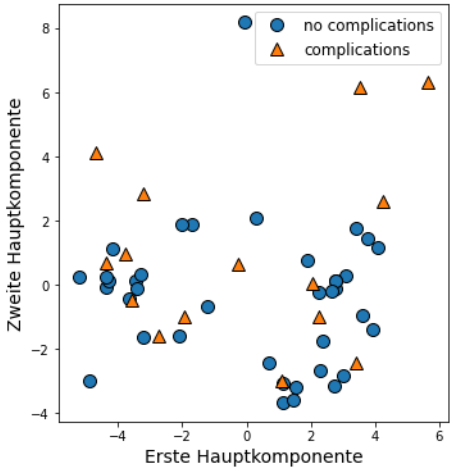
	m=3, l=0.1	m=1, l=0.1	m=3, l=0.01	m=1, l=0.01
F1-Score	0.250000	0.444444	0.000000	0.000000
Train Accuracy	1.000000	1.000000	0.974359	0.820513
Test Accuracy	0.571429	0.642857	0.500000	0.571429
MCC	-0.025950	0.188562	-0.304290	-0.206725

(c) Scores des Gradient Boosting Algorithmus

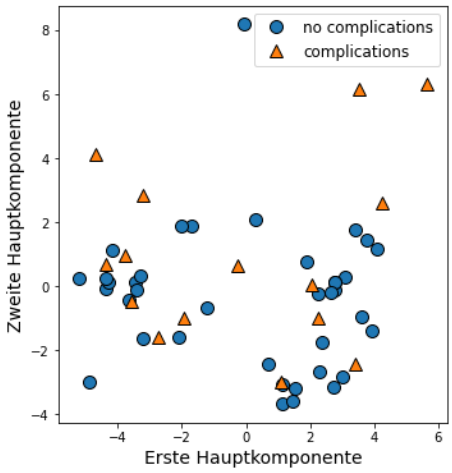
Abbildung 31: Scores der drei Modelle für die verschiedenen Tree Algorithmen mit Maximal Datensatz. Der Kopf der Tabellen zeigt die Parameter Einstellungen an.



(a) Scores des vitals-Datensatz



(b) Scores des vca-Datensatz



(c) Scores des all-Datensatz

Abbildung 32: Scores der drei Modelle für die verschiedenen PCA Algorithmen mit Maximal Datensatz.

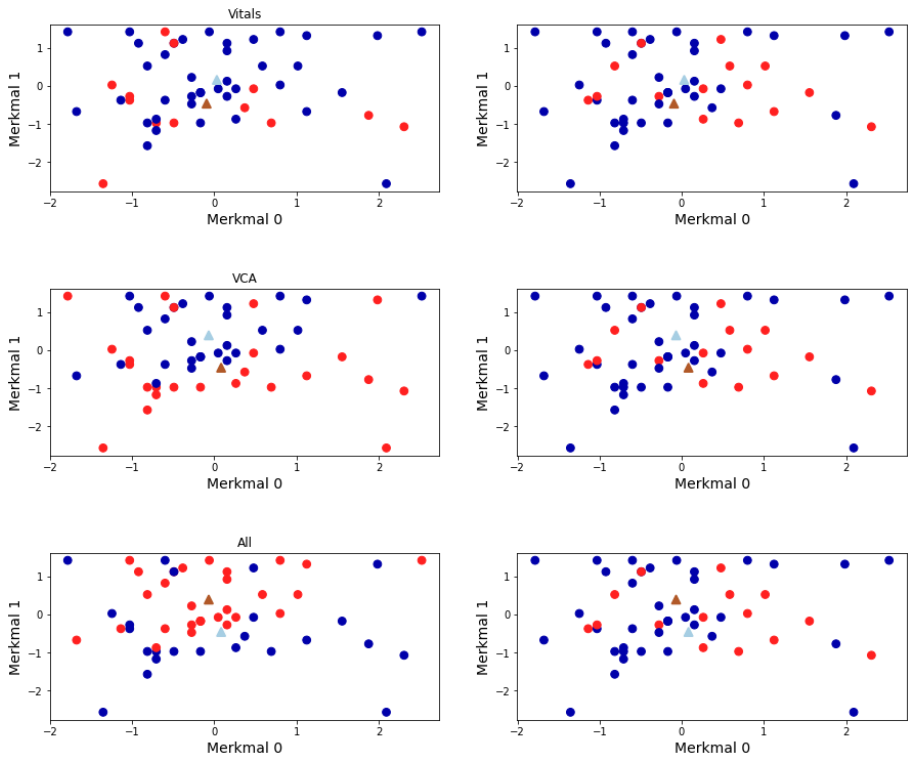


Abbildung 33: Scores des k-means Clustering Algorithmus mit Maximal Datensatz

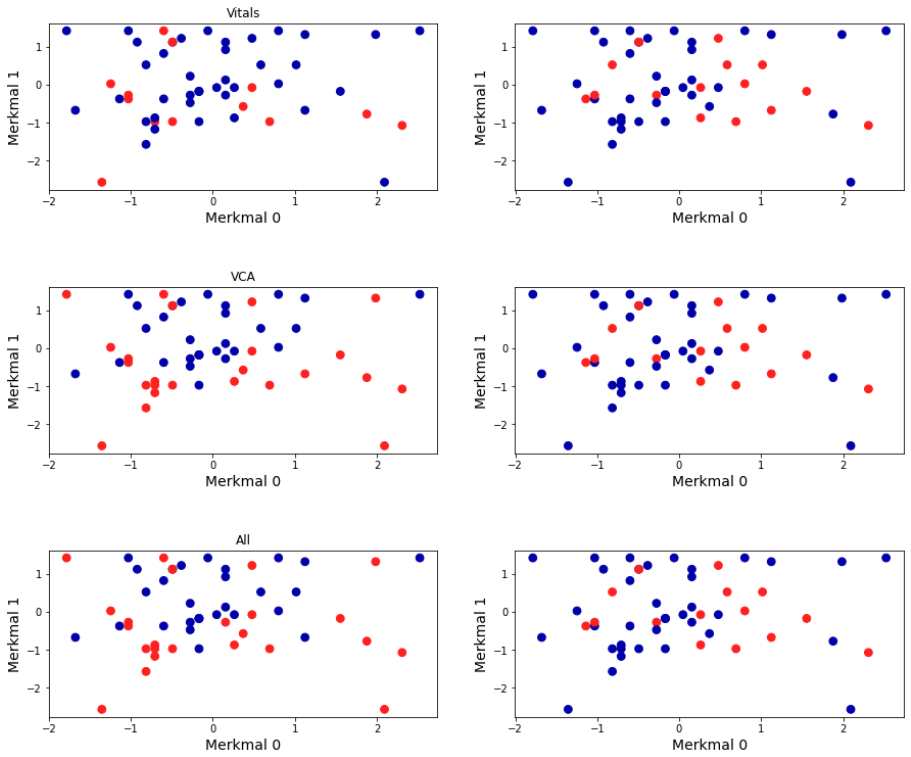


Abbildung 34: Scores des Agglomerative Clustering Algorithmus mit Maximal Datensatz

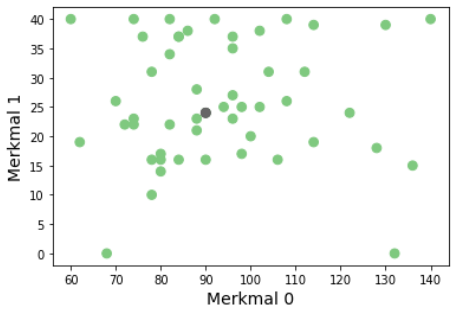


Abbildung 35: Scores des DBSCAN Algorithmus mit Maximal Datensatz

Abbildungsverzeichnis

1 OHE Beispiel anhand von Farben 5
2 Beispiel einer optimalen Hyperebene - Hard Margin - bei einer SVM. Quelle: (Nguyen und Zeigermann 2017, S. 83) 10
3 Die Verteilung der Daten mit dem unscaled-Datensatz 21
4 Die Verteilung der Daten mit dem vitals-Datensatz 22
5 Die Verteilung der Daten mit dem vca-Datensatz 23
6 Die Verteilung der Daten mit dem all-Datensatz 23
7 vitals-Datensatz mit 1 bis 10 Nachbarn 25
8 Scores der vier Modelle für den k-Nearest-Neighbor Algorithmus. Der Kopf der Tabellen zeigt die Anzahl der Nachbarn an. 26
9 Scores der drei Logistic Regression Modelle mit *lbfgs*-Solver. Der Kopf der Tabellen zeigt die *C*-Regulation an 27
10 Lernkurve von Logistic Regression mit lbfgs mit dem vitals- und vca-Datensatz . . 27
11 Koeffizientengewichtung für die besten Modelle von dem Logistic Regression Algorithmus 28
12 Scores der drei Modelle von Random-Forest mit unterschiedlichen Tiefen. Der Kopf der Tabellen zeigt die Datensätze an, die benutzt wurden. 29
13 Gewichtung der Merkmale beim Random Forest Algorithmus für alle Datensätze 30
14 Scores des Gradient Boosting Classifier Algorithmus für alle Datensätze 30
15 Gewichtung der Merkmale beim Gradient Boosting Classifier Algorithmus für alle Datensätze 31
16 Übersicht der einzelnen Merkmale vom unscaled-Datensatz. Die beiden Bezeichnungen in der Legende stehen für Komplikation (comp) und keine Komplikation (no_comp). 33
17 Hauptkomponenten für die verschieden skalierten Datensätze mit PCA 34
18 Plots der verschieden skalierten Datensätze für den k-means-Clustering Algorithmus. Die roten Punkte stellen die Merkmale dar, die der positiven Klasse angehören. Die blauen Punkte gehören der negativen Klasse an. 35

19 Plots der verschieden skalierten Datensätze für den Agglomerative Clustering Algorithmus. Die roten Punkte stellen die Merkmale dar, die der positiven Klasse angehören. Die blauen Punkte gehören der negativen Klasse an. 36
20 Plot des DBSCAN Algorithmus für alle skalierten Datensätze 37
21 Scores des Decision Trees 43
22 Merkmalsgewichtung von Decision Tree mit maximaler Tiefe 44
23 Scores der drei Modelle für den SVM Algorithmus. Der Kopf der Tabellen zeigt die Einstellungen der Parameter an 45
24 Scores der drei Modelle für den k-Nearest-Neighbor Algorithmus mit PCA. Der Kopf der Tabellen zeigt die Anzahl der Nachbarn an 45
25 Scores des vca- und all-Datensatz mit lbfgs- und saga-Solver für Logistic Regression 46
26 Scores der drei Modelle für die verschiedenen Tree Algorithmen. Der Kopf der Tabellen zeigt die Parameter Einstellungen an. 46
27 Scores von SVM mit vitals-Datensatz 46
28 Scores der vier Modelle für den k-Nearest-Neighbor Algorithmus mit Maximal Datensatz. Der Kopf der Tabellen zeigt die Anzahl der Nachbarn an 47
29 Scores von SVM mit vitals-Datensatz mit Maximal Datensatz 47
30 Scores aller Datensätze mit lbfgs-, saga-Solver und L1-Regulation für Logistic Regression mit Maximal Datensatz 47
31 Scores der drei Modelle für die verschiedenen Tree Algorithmen mit Maximal Datensatz. Der Kopf der Tabellen zeigt die Parameter Einstellungen an. 48
32 Scores der drei Modelle für die verschiedenen PCA Algorithmen mit Maximal Datensatz. 49
33 Scores des k-means Clustering Algorithmus mit Maximal Datensatz 50
34 Scores des Agglomerative Clustering Algorithmus mit Maximal Datensatz 50
35 Scores des DBSCAN Algorithmus mit Maximal Datensatz 51

Liste für Abkürzungen

MCC Matthews Correlation Coefficient

ML Machine Learning

OHE One-Hot-Encoder

PCA Principle Component Analysis

SAB Subarachnoidalblutung

SVM Support Vector Machine

UKD Universitätsklinikum Düsseldorf