E5b Lehrstuhlversuch

Datenanalyse mit IceCube-Monte-Carlo Simulationsdaten

Philipp Hoffmann
philipp2.hoffmann@udo.edu

Alexander Drossel
alexander.drossel@udo.edu

Durchführung: 14.09.2016

Abgabe: 22.09.2016

TU Dortmund – Fakultät Physik

Inhaltsverzeichnis

1 Einleitung

1.1 Ziel des Versuchs

Ziel des Versuchs ist es anhand simulierter Daten des Ice-Cube-Experiments die relevanten Signale mittels maschinellen Lernens vom Untergrund zu trennen. Hierbei liegt der Fokus auf den einzelnen Schritten, die für die Trennung der Daten durchlaufen werden. Es sollen so Grundkenntnisse über das maschinelle Lernen am Ice-Cube-Experiment aufgezeigt werden.

1.2 Astrophysikalische Grundlagen

Im Jahre 1912 wurde erstmals von Viktor Heß die Höhenstrahlung entdeckt. Seitdem ist bekannt, dass nicht nur Photonen in die Atmosphäre der Erde eindringen, sondern auch Protonen und schwerere Kerne. Dies wird als Höhenstrahlung bzw. geladene kosmische Strahlung bezeichnet. Bei der geladenen Strahlung werden Energien von bis zu $10^{20}\,\mathrm{eV}$ beobachtet und sie folgt annähernd einem Potenzgesetz

$$\frac{\mathrm{d}\theta}{\mathrm{d}E} = \theta_0 E^{\gamma}$$

mit $\gamma \approx -2.7$ als dem spektralen Index. Geladene kosmische Strahlung besteht zum Großteil aus Protonen und geladenen Kernen und wird durch (extra)galaktische Magnetfelder abgelenkt. Deswegen lässt sich die Quelle dieser Strahlung nicht ermitteln. Jedoch sollten Quellen die Hadronen beschleunigen, auch Neutrinos und hochenergetische Photonen erzeugen. Da Neutrinos im Gegensatz zu hadronischen Teilchen ungeladen sind und einen sehr kleinen Wirkungsquerschnitt besitzen, werden sie nicht von Magnetfeldern abgelenkt und können durch die sehr selten auftretenden Wechselwirkungen zu einem deutlich größeren Anteil Materie durchdringen. Neutrinos zeigen somit nahezu direkt auf ihren Ursprungsort zurück.

1.3 Das IceCube-Experiment

Das IceCube-Experiment am geografischen Südpol dient zur Detektion hochenergetischer Neutrinos. Es besteht aus drei Stationen: IceTop, dem In-Ice-Array und DeepCore. Das In-Ice-Array und DeepCore bestehen aus 86 in Eis eingeschmolzenen Kabeln, an denen in einer Eistiefe von 1450 m bis 2450 m insgesamt 5160 Photoelektronenvervielfachern angebracht sind. Davon befinden sich sieben Kabel mit einem geringeren Abstand aneinander und bilden so den DeepCore. DeepCore besitzt eine untere Energieschwelle von 10 GeV. Die Energieschwelle des restlichen In-Ice-Arrays liegt bei 100 GeV. An der Oberfläche des Eises befindet sich das Luftschauer-Experiment IceTop. Hier werden die Teilchen in lichtdichten Eistanks über ihr Tscherenkowlicht detektiert. Da sich IceTop an der Oberfläche befindet kann diese Station als Veto für die Neutrinodetektion verwendet werden, weil hier hauptsächlich kosmische Strahlung und Teilchen aus Luftschauern in Wechselwirkung treten. So ist es beispielsweise auch möglich Neutrinos vom Südhimmel zu detektieren. Tscherenkowlicht entsteht, wenn sich geladene Teilchen schneller als

Lichtgeschwindigkeit in einem Medium bewegen. Im Eis werden die Neutrinos der Flavour ℓ mittels Sekundärteilchen detektiert. Diese entstehen durch die zwei folgenden Wechselwirkungen mit Kernen A im Eis und restlichen Reaktionsprodukten X:

$$\nu_\ell(\bar{\nu}_\ell) + A \to \ell^\pm + X$$
$$\nu_\ell + A \to \nu_\ell + X$$

Da Myonen eine Vergleichsweise lange Lebensdauer besitzen werden diese und ihre Erzeuger, die Myon-Neutrinos, am häufigsten detektiert. Myonen aus Neutrinointeraktionen sind im Folgenden das Signal. Jedoch werden Myonen auch in großer Zahl durch Wechselwirkung geladener kosmischer Strahlung in der Atmosphäre erzeugt; diese sind der Untergrund.

$$p + A \to \pi^+/K^+ + X,$$
$$\pi^+/K^+ \to \mu + \nu_\mu.$$

Myonen aus Luftschauern treten ca. 10^6-mal so häufig auf wie Myonen aus Myon-Neutrinos. Da Myonen im Gegensatz zu Neutrinos nicht die Erde durchqueren können, muss es sich bei Ereignissen deren rekonstruierte Richtung nicht vom Südhimmel kommt um Neutrinos oder fehlrekonstruierte kosmische Myonen handeln. Wird so ein Schnitt am Zenitwinkel der rekonstruierten Richtung durchgeführt bleiben noch etwa 10^3-mal so viele Untergrund- wie Signalereignisse. Um diese vom Signal zu trennen werden Methoden des maschinellen Lernens verwendet.

2 Methoden des maschinellen Lernens

Um die relevanten Signale von den Untergrunddaten zu trennen werden in diesem Versuch Methoden des maschinellen Lernens verwendet. Der typische Ablauf für die Prozessierung simulierter Daten sieht wie folgt aus:

Vorbereitung der Daten Zu Beginn werden die Daten für die weiteren Schritte vorbereitet.

Attributselektion Es werden die für die Lerner relevanten Attribute mit einem Algorithmus herausgesucht. Dies verringet die Rechenzeit in den folgenden Schritten drastisch. Die Attributselektion wird hiernach mit dem Jaccard-Index auf die Stabilität gegen statistische Schwankungen im Lerner untersucht.

Multivariate Selektion Die Lernen werden auf die vorbereiteten Datensätze trainiert. Der trainierte Lerner lässt sich danach auf andere Datensätze anwenden.

Überprüfen der prozessierten Daten Zum Schluss werden die bearbeiteten Daten auf ihre Genauigkeit geprüft. Hierzu werden Qualitätsparameter eingeführt und die Lerner mittels Kreuzvalidierung getestet.

Im Folgenden werden diese Schritte und einige der in diesem Versuch verwendeten Methoden näher erläutert.

2.1 Attributsselektion mittels mRMR

Bei der Attributsauswahl mittels minimum Redundancy Maximum Relevance (mRMR) wird die Wahrscheinlichkeitsverteilung der einzelnen Attribute betrachtet und ist somit nicht von einem spezifischen Lerner abhängig. Hierzu wird der wechselseitige Informationsgehalt zweier Attribute x, y benutzt:

$$I(x,y) = \int p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \mathrm{d}x\mathrm{d}y$$

Hierbei bezeichnen $p(x)$, $p(y)$ und $p(x,y)$ die Wahrscheinlichkeitsdichtefunktionen der betreffenden Attribute. Die Attribute werden hierbei vom mRMR-Algorithmus so ausgewählt, dass sie möglichst stark mit dem Zielattribut, aber untereinander möglichst wenig korreliert sind.

2.2 Stabilitätsanalyse mit dem Jaccard-Index

Der Jaccard-Index in ein Maß für die Ähnlichkeit zweier Mengen. Um ihn zu berechnen teilt man die Anzahl der gemeinsamen Elemente (Schnittmenge) durch die Größe der Vereinigungsmenge:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Dies wird l-mal auf l Teilmengen des Datensatzes angewendet. Somit lässt sich die Ähnlichkeit der verschiedenen Selektionen beurteilen. Hierbei beschreibt ein Wert um 1.0 eine maximal stabile Attributauswahl gegen statistische Schwankungen.

$$\hat{J} = \frac{2}{l(l-1)} \sum_{i=1}^{l} \sum_{j=i+1}^{l} J(F_i, F_j)$$

2.3 Multivariate Lerner

Die Verwendung von multivariaten Lernern bietet den Vorteil, dass diese auch Korrelationen zwischen den Variablen beachten.

2.3.1 Random Forest

Der Random Forest Lerner basiert auf binären Entscheidungsbäumen. Jeder Entscheidungsbaum erhält nur eine Teilmenge der vorhandenen Attribute oder Ereignisse, womit die Korrelation der Bäume untereinander verkleinert wird. Beim erstellen eines solchen Baums werden für jedes Ereignis an den Knoten die Werte eines Attributs betrachtet. Es wird durch Minimieren oder Maximieren eines Optimierungskriteriums ein Schnitt in diesem Attribut gesetzt. Diese Schnitte werden wiederholt, bis die maximale Tiefe des Baums erreicht ist. Der letzte Attributsschnitt bestimmt zu welcher Klasse ein Ereignis zugeordnet wird. Beim vorliegenden Zweiklassenproblem dieses Versuchs ist dies entweder Signal oder Untergrund, was als 1 oder 0 ausgedrückt werden kann. Um den

Effekt des Übertrainings zu minimieren, werden N Bäume angelegt und folgend über diese zu einer Konfidenz gemittelt. Die Konfidenz (c) des Random Forest-Algorithmus wird definiert als das Mittel über der Entscheidungen P_i der N Teilbäume:

$$c = \frac{1}{N} \sum_{i=1}^{N} P_i, P_i \in \{0, 1\}$$

2.3.2 Naive Bayes-Algorithmus

Der Naive-Bayes-Algorithmus basiert auf dem Satz von Bayes. Dies ist ein mathematischer Satz der Wahrscheinlichkeitstheorie, der die Berechnung bedingter Wahrscheinlichkeiten beschreibt. Die bedingte Wahrscheinlichkeit $P(A|B)$ ist die (bedingte) Wahrscheinlichkeit, dass Ereignis A eintritt, unter der Bedingung, dass B bereits eingetreten ist. Das Bayes'sche Theorem sagt aus:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

In diesem Fall beschreibt A die Klassenzugehörigkeit (Signal A oder Untergrund $\bar{A}$) und B ein Attribut. Somit lässt sich folgender Ausdruck Formulieren:

$$Q = \frac{p(A|B)}{p(\bar{A}|B)} = \frac{p(B|A)p(A)}{p(B|\bar{A})p(\bar{A})}$$

Dieser Ausdruck nimmt einen Wert von 1 an, wenn das Ereignis mit höherer Wahrscheinlichkeit ein Signal ist. Für mehrere Attribute wird der Ausdruck zu:

$$Q = \frac{p(A|B_1, ..., B_n)}{p(\bar{A}|B_1, ..., B_n)} = \prod_{i=1}^{n} \frac{p(B_i|A)}{p(B_i|\bar{A})}$$

2.4 Gradient Boosting

Gradient Boosting ist ein iterativer Lerner der auf Entscheidungsbäumen basiert. Er erzeugt in jedem Iterationsschritt einen neuen Lerner, der eine möglichst geringe Abweichung zum idealen Modell besitzen soll. Dies geschieht durch Minimierung einer Kostenfunktion, welche anhand der Start-/ und Zielwerte, sowie dem vorangegangenen Lerner, ein Maß für die Abweichung der durch den Lerner errechneten Zielwerte zu den realen Zielwerten beschreibt. Diese Lerner können alleine betrachtet als schwach bezeichnet werden und stellen noch kein gutes Modell dar. Beim Gradient Boosting wird mit den in den Iterationsschritten berechneten Gewichten und zugehörigen schwachen Lernern eine gewichtete Summe gebildet. Das Ergebnis dieser kann somit als ein starker Lerner bezeichnet werden.

2.5 Qualitätsparameter

Um die Güte der Entscheidungen eines Lerners zu berechnen werden Qualitätsparameter eingeführt.

$$\text{Reinheit } p = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}} \tag{1}$$

$$\text{Effizienz } r = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}} \tag{2}$$

Mit dem richtig als Signal klassifizierten Ereignissen (tp, true positive), den richtig als Untergrund klassifizierten Ereignissen (tn, true negative) und der Anzahl der fälschlich als Signal (fp, false positive) bzw. fälschlich als Untergrund (fn, false negative) klassifizierten Ereignisse.

3 Durchführung

In diesem Teil wird auf die verwendeten Methoden des maschinellen Lernens eingegangen sowie die Vorprozessierung der Daten erläutert. Für diesen Versuch werden simulierte Signal- und Untergrundereignisse verwendet. Für den Vergleich zwischen den verschiedenen Lernen werden drei verschiedene Lerner verwendet.

3.1 Vorprozessierung

Der Zweck der Vorprozessierung ist es die Signal- und Untergrunddaten in einen gemeinsamen Datensatz zu bringen und nicht sinnvoll verwendbare Attribute zu entfernen. Hierzu müssen alle Attribute entfernt werden die nicht in beiden Datensätzen vorkommen. Weiterhin werden alle Monte-Carlo-Wahrheiten und Werte die nicht als Zahl identifiziert werden können (NaN, Inf) entfernt oder ersetzt. Als letztes werden die Daten mit einem binären Label versehen, das bestimmt ob ein Ereignis zu Signal und Untergrund gehört und das als Zielattribut für die Lerner verwendet wird.

3.2 Attributselektion

Um die Rechenzeit zu verringern ist es sinnvoll zuerst die Attribute auszuwählen, die dem Lerner eine möglichst gute Unterscheidung zwischen Signal und Untergrund ermöglichen. Für diesen Prozess gibt es mehrere Verfahren, darunter die Vorwärtsauswahl (Forward Selection) und die mRMR-Auswahl (minimum Redundancy, Maximum Relevance). Die Vorwärtsauswahl gibt das bestmögliche Ergebnis, besonders weil es Attribute auswählt die für einen bestimmten Lerner am besten sind, was dafür sorgt, dass die Attributsauswahl nicht verallgemeinerbar ist. Die Vorwärtsauswahl ist außerdem mit einer sehr hohen Rechenzeit verbunden und erhöht durch das oft wiederholte Trainieren des Lerners die Gefahr der Überanpassung des Modells an die vorhandenen Daten.

Aus diesem Grund wird in diesem Versuch die mRMR-Auswahl verwendet. Siehe hierzu Kapitel 2.1. Um die Stabilität der Attributauswahl gegen statistische Schwankungen zu testen wird hierbei der Jaccard-Index verwendet, siehe Kapitel 2.2.

3.3 Lernalgorithmen

In diesem Versuch wird als Lerner der RandomForest-Algorithmus (Kapitel 2.3.1), Gradient Boosting (Kapitel 2.4) und der Naive Bayes-Algorithmus (Kapitel 2.3.2), verwendet. Sie werden in einer Kreuzvalidierung trainiert , um zu zeigen, dass der Lerner auch auf Datensätze angewendet werden kann auf die er nicht trainiert wurde. Hierzu wird der Datensatz in n Teile aufgeteilt und auf $n-1$ von diesen trainiert. Daraufhin wird er auf den letzten Datensatz angewendet um den Lerner zu validieren. Das Vorgehen wird n-mal wiederholt, sodass jeder Datensatz einmal zum testen verwendet wird. Zum Überprüfen der Qualität des Lerners werden die Qualitätsparameter aus Kapitel 2.5 verwendet.

4 Auswertung

Die Auswertung hat den Vergleich mehrerer Algorithmen des maschinellen Lernens zum Ziel. Dafür werden die zur Verfügung gestellten Daten erst mit Pandas [4] vorprozessiert und auf ihnen mittels mRMR [1] eine Attributselektion durchgeführt.

4.1 Vorprozessierung

Die zur Verfügung gestellten Datensätze bestehen aus insgesamt 40 000 Ereignissen. Eine Hälfte davon sind Neutrinos (Signal), die andere Myonen (Untergrund). Von diesen Ereignissen werden zuerst Attribute entfernt, die nicht in beiden Datensätzen vorhanden sind; es bleiben 237 Attribute übrig. Dann werden unphysikalische Attribute, wie die IDs der Ereignisse und Informationen, die nur in der Simulation vorhanden sind, entfernt; es bleiben 188 Attribute übrig. Dann werden Attribute mit einem Anteil an NaN-Werten von über 20% entfernt; es bleiben 169 Attribute übrig. Basis dafür ist die NaN-Verteilung in Abbildung 1. Sie zeigt, dass die meisten Attribute weniger als 20% NaNs enthalten. Die etwa 19 Attribute mit einem Anteil größer 90% enthalten nicht genügend Informationen für die Separation. Die NaNs in den restlichen Attributen werden durch die Zahl -5000 ersetzt. Eigentlich könnten alle NaNs auch zuerst mit dieser Zahl ersetzt werden und das Ausfiltern der Attribute mit zu vielen NaNs der Attributselektion überlassen werden. Sind zu viele NaNs vorhanden entspricht dies einem geringen Informationsgehalt des Attributs, was automatisch erkannt werden sollte. Konstante Attribute wurden auch nicht explizit entfernt, da diese vom mRMR automatisch herausgefiltert werden.

4.2 Attributselektion

Durchgeführt wird eine fünffache Attributselektion mittels mRMR. Dafür werden aus der Menge der Gesamtereignisse fünf gleich große Ereigismengen mit zurücklegen gezogen und auf jeder dieser Mengen eine Attributselektion durchgeführt, um so eine Abschätzung der Varianz der Selektion zu erhalten. Die dafür verwendeten Ereignismengen wurden durch Ziehen mit Zurücklegen aus der Gesamtmenge von 40 000 Ereignissen erstellt. Als Stabilität der Attributselektion wird der Jaccard-Index verwendet. Die gemittelten

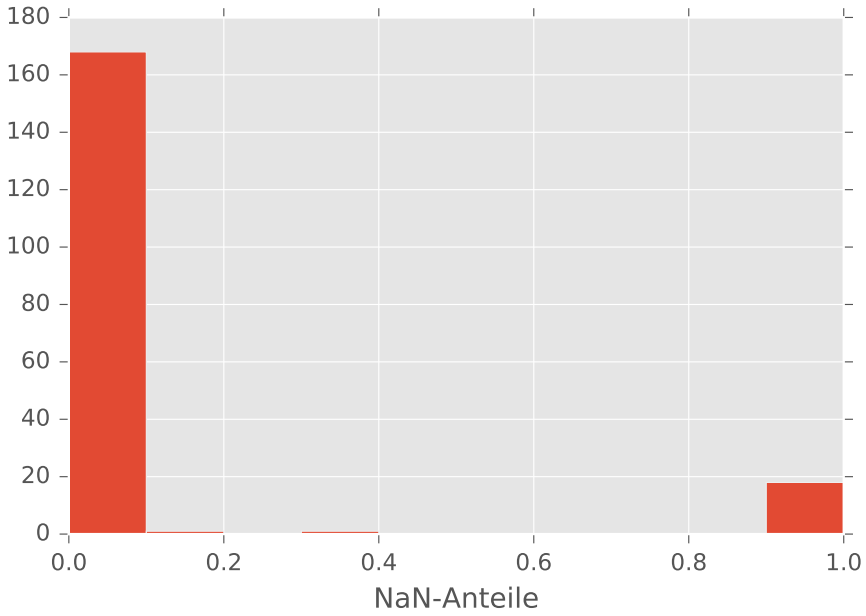

Abbildung 1: Histogramm der NaN-Anteile in den noch vorhandenen Attributen.

Stabilitäten mit ihren Standardabweichungen gegen die Anzahl an selektierten Attributen k sind in Abbildung 2 zu sehen. Die Mittelwerte steigen erst und sättigen sich bei $k = 28$ zu etwa 0.9. Hier ist auch die Standardabweichung der Mittelwerte am kleinsten. Dieser k-Wert wird als Zahl der Attribute für die Separation gewählt. Welche 28 Attribute genau ausgewählt werden, wird über eine eine Mehrheitsentscheidung der fünf Attributselektionen bestimmt. Die 28 Attribute, die im Durchschnitt der Selektionen die höchsten Bewertungen erhalten, werden ausgewählt.

4.3 Separation

Für die Separation werden innerhalb der Optimierung durch auto-sklearn [2] Algorithmen aus scikit-learn [6] verwendet. Außerdem wird noch der Naive-Bayes und Gradient Boosted Trees Operator aus RapidMiner [5], sowie der WEKA Random Forest [3] verwendet.

4.3.1 Naive-Bayes

Als Referenz für die Separationsqualitäten wurde der Naive-Bayes (NB) Algorithmus betrachtet. Die gemittelten Reinheiten und Effizienzen aus der Kreuzvalidierung gegen die Konfidenz sind in Abbildung 3 zu finden. Die Standardabweichungen des Mittelwerts sind als transparentes Band um die Mittelwerte eingezeichnet. Der NB zeigt bereits eine sehr starke Trennung der beiden Klassen. Die Reinheit steigt für größer werdende Konfidenzen nur in einem einstelligen Prozentbereich und erreicht einen maximalen Wert nahe 1. Die Standardabweichungen sind im Verhältnis zu den Mittelwerten vernachlässigbar

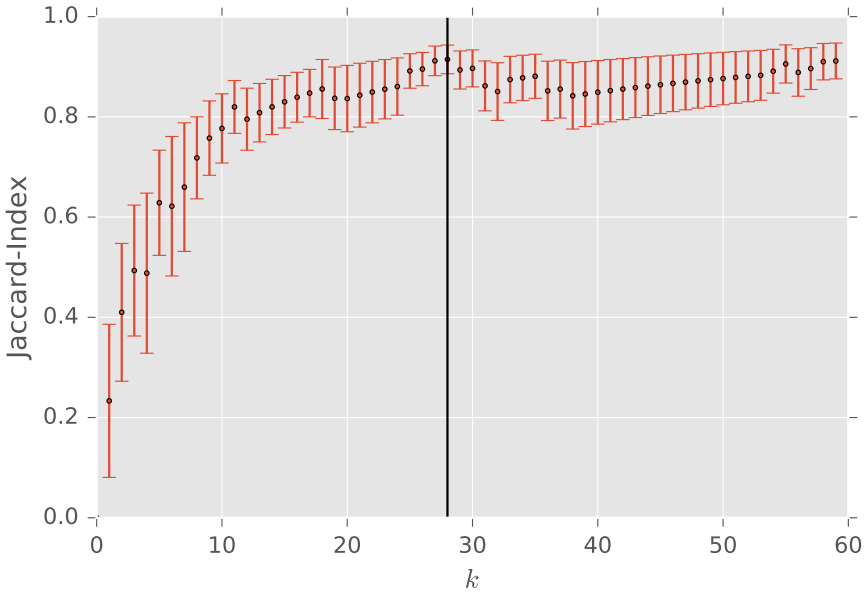

Abbildung 2: Die Mittelwerte der Jaccard-Indices mit ihren Standardabweichungen, gegen die Anzahl an selektierten Attributen.

klein.

4.3.2 Random Forest

Der Random Forest wurde mit RapidMiner mit 200 Bäumen und dem Entropie Optimierungskriterium auf 40 000 Ereignissen trainiert. Die Reinheit und Effizienz gegen die Konfidenz des Random Forests ist in Abbildung 4 dargestellt. Der Random Forest zeigt ab Konfidenzen größer 0.5 eine höhere Reinheit als auch Effizienz als der NB. Bei einer Konfidenz von 1.0 holt der NB in der Reinheit auf, während die Effizienz um etwa 25% schlechter wird. Die Standardabweichungen sind im Verhältnis zu den Mittelwerten vernachlässigbar klein.

4.3.3 Gradient Boosting

Die Algorithmenauswahl und Hyperparameterbestimmung erfolgt durch die Python Bibliothek auto-sklearn [2], die ein Ensemble aus mehreren Lernern erstellt und ihre Hyperparameter anhand von selbst auswählbaren Qualitätskriterien optimiert. Als zu maximierendes Kriterium wurde die der Anteil richtig Klassifizierter Ereignisse an der Gesamtmenge der Ereignisse gewählt. Die zu verwendenden Algorithmen wurden über 27 Stunden optimiert. Dabei hat auto-sklearn zuerst ein Gemisch aus Random Forests, Gradient Boosting und anderen Methoden verwendet. Nach mehreren Stunden Rechenzeit wurden alle Algorithmen durch eine Kombination von 20 Gradient Boosting Lernern ersetzt.

Reinheit und Effizienz mit ihren Standardabweichungen gegen die Konfidenz sind in

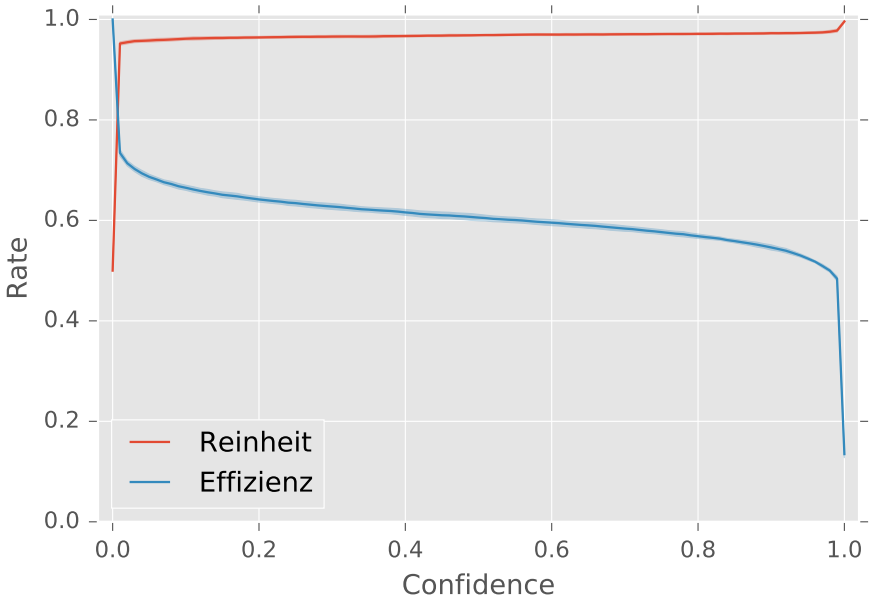

Abbildung 3: Aufgetragen sind Mittelwerte der Reinheiten und Effizienzen mit ihren Standardabweichungen (als farbige Bänder) gegen die Konfidenz für den Naive-Bayes aus einer fünffachen Kreuzvalidierung.

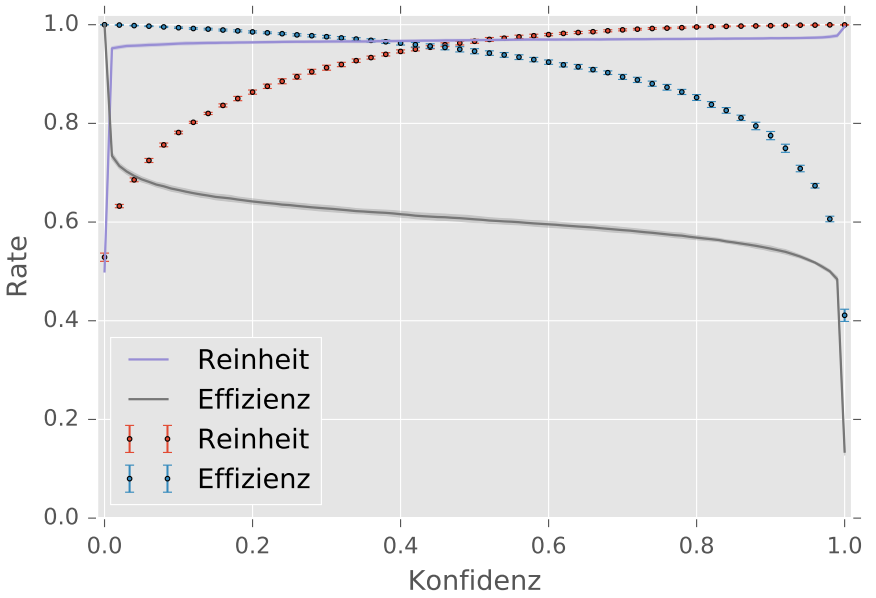

Abbildung 4: Aufgetragen sind Mittelwerte der Reinheit und Effizienz mit ihren Standardabweichungen gegen die Konfidenz für den Random Forest (Punkte) und den Naive-Bayes (durchzogene Linien) aus einer fünffachen Kreuzvalidierung. Die Standardabweichungen sind durch die farbigen Bänder und Fehlerbalken gekennzeichnet.

Abbildung 5 zu sehen. Der Durchschnitt der Konfidenzen der einzelnen Lerner ist die Konfidenz des Ensembles. Die Effizienz des auto-sklearn Ensembles ist für alle Konfidenzen außer 1.0 höher als die des Naive-Bayes. Für Konfidenzen größer 0.5 zeigen sich, wie beim Random Forest, höhere Reinheiten als beim NB. Die Standardabweichungen sind im Verhältnis zu den Mittelwerten vernachlässigbar klein.

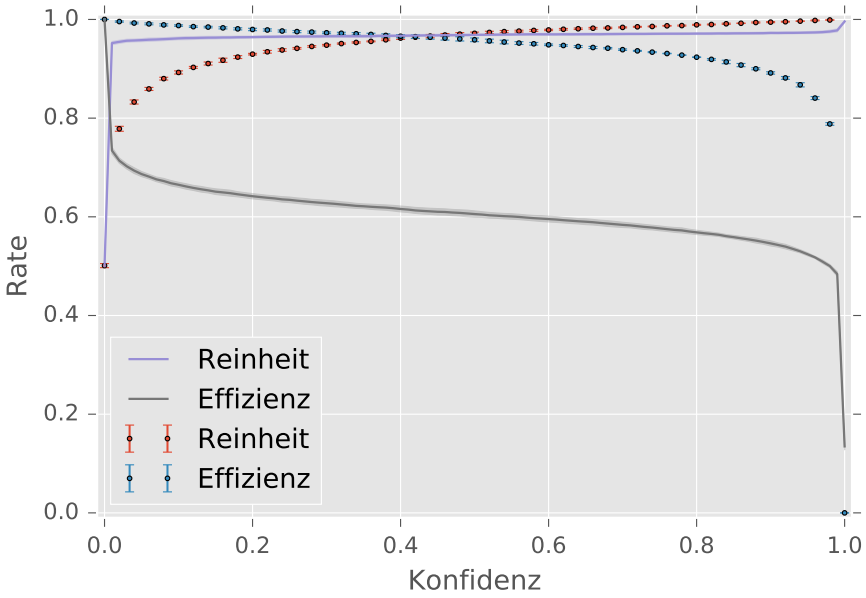

Abbildung 5: Aufgetragen sind Mittelwerte der Reinheit und Effizienz mit ihren Standardabweichungen gegen die Konfidenz für das auto-sklearn ensemble (Punkte) und den Naive-Bayes (durchzogene Linien) aus einer fünffachen Kreuzvalidierung. Die Standardabweichungen sind durch die farbigen Bänder und Fehlerbalken gekennzeichnet.

Abbildung 6 ist analog zu Abbildung 5 erstellt worden. Es wurde hier das Gradient Boosting Ensemble von auto-sklearn durch den RapidMiner Gradient Boosted Trees Operator ersetzt. Es zeigt sich ein sehr ähnliches Bild wie in Abbildung 5.

4.4 Vergleich der Gradient Boosted Trees

Die Qualitätsunterschiede zwischen der RapidMiner Gradient Boosted Trees Implementierung und dem optimierten Ensemble von Gradient Boosted Trees aus auto-sklearn werden in Abbildung 7 in einem kleineren Ausschnitt verglichen. Die durchzogenen Linien sind die Mittelwerte des auto-sklearn Ensembles. Es zeigen sich nur geringe Abweichungen zwischen den beiden Lernern, die nur bei Konfidenzen um 0.5 größer als eine Standardabweichungen der beiden Lerner werden. Die Lerner sind somit weitgehend äquivalent.

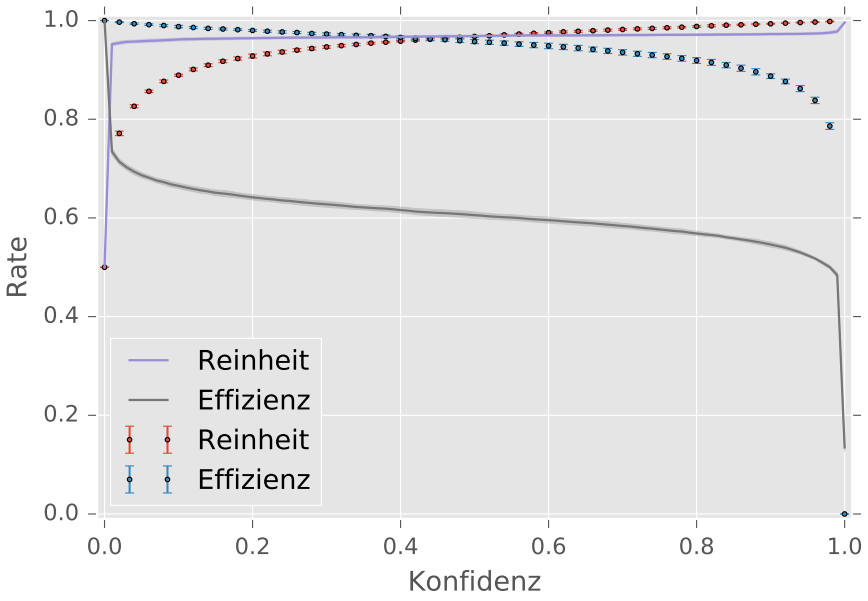

Abbildung 6: Aufgetragen sind Mittelwerte der Reinheit und Effizienz mit ihren Standardabweichungen gegen die Konfidenz für das den RapidMiner Gradient Boosted Trees Operator (Punkte) und den Naive-Bayes (durchzogene Linien) aus einer fünffachen Kreuzvalidierung. Die Standardabweichungen sind durch die farbigen Bänder und Fehlerbalken gekennzeichnet.

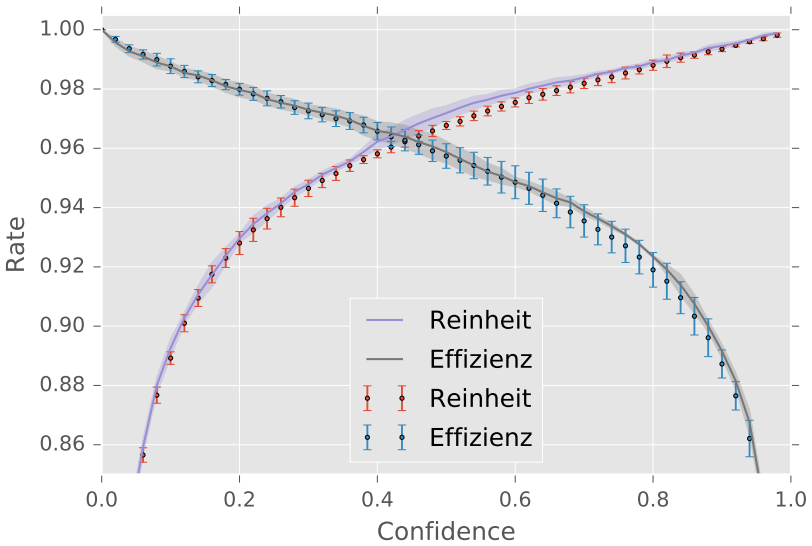

Abbildung 7: Aufgetragen sind Mittelwerte der Reinheit und Effizienz mit ihren Standardabweichungen gegen die Konfidenz für das auto-sklearn Ensemble (durchzogene Linien) und den RapidMiner Gradient Boosted Trees Operator (Punkte) aus einer fünffachen Kreuzvalidierung. Die Standardabweichungen sind durch die farbigen Bänder und Fehlerbalken gekennzeichnet.

4.5 Vergleich von Random Forest und Gradient Boosted Trees

In Abbildung 8 wird der RapidMiner WEKA Random Forest Operator mit dem Gradient Boosted Trees (GBT) Operator verglichen. Der GBT zeigt für alle Konfidenzen bis auf 1.0 eine höhere Effizienz, während für Konfidenzen größer 0.5 seine Reinheit leicht hinter der des Random Forests liegt. Es ist zu beachten, dass für den GBT die Reinheiten für die selben Konfidenzen in diesem Bereich zwar niedriger als die des Random Forests sind, die Reinheiten des GBT für vorgegebene Werte der Effizienz aber höher als die des Random Forest sind. Diese Überlegenheit des GBT gilt bis zu Reinheiten von 99.9%, erst ab hier hat der GBT eine niedrigere Effizienz als der RF. Die Effizienz des GBT wird dort gleich Null.

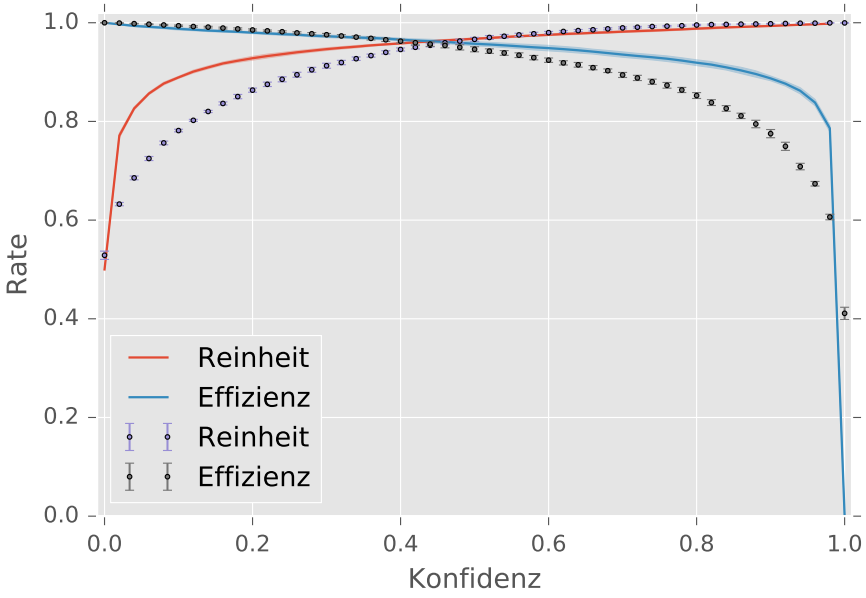

Abbildung 8: Aufgetragen sind Reinheit und Effizienz mit ihren Standardabweichungen gegen die Konfidenz für die RapidMiner Operatoren WEKA Random Forest (Punkte) und Gradient Boosted Trees (durchzogene Linien) aus einer fünffachen Kreuzvalidierung. Die Standardabweichungen sind durch die farbigen Bänder und Fehlerbalken gekennzeichnet.

5 Diskussion

In dieser Separation zeigen alle Lerner, selbst der Naive-Bayes, eine hohe Trennkraft, was für eine sehr gute Konstruktion der Attribute spricht. Weiter zeigt sich, dass Gradient Boosted Trees für die vorliegenden Daten einem Random Forest überlegen sind, so lange nicht die bei einer Konfidenz von 1.0 auftretende höchstmögliche Reinheit gefordert wird.

Das auto-sklearn Ensemble aus 20 Gradient Boosted Tree Lernern zeigt trotz seines höheren Rechenaufwands nur geringe Abweichungen von der einfachen GBT Implementierung in Rapidminer. Kann RapidMiner ohne weiteres Verwendet werden ist auto-sklearn den damit verbundenen Mehraufwand nicht wert. Es bleibt jedoch offen, ob die Separation von auto-sklearn durch mehr Optimierungszeit verbesserbar ist, da hier etwa die hälfte der 27 Stunden gebraucht wurde um von verschiedenen Lernern auf ein Ensemble von Gradient Boosted Trees zu kommen, deren Einstellungen noch weiter angepasst werden können.

Literatur

[1] Nicolas De Jay u. a. „MRMRe: An R package for parallelized mRMR ensemble feature selection“. In: *Bioinformatics* 29.18 (2013), S. 2365–2368. ISSN: 13674803. DOI: 10.1093/bioinformatics/btt383.

[2] Matthias Feurer u. a. „Efficient and Robust Automated Machine Learning“. In: *Advances in Neural Information Processing Systems 28*. Hrsg. von C. Cortes u. a. Curran Associates, Inc., 2015, S. 2962–2970. URL: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf.

[3] Mark Hall u. a. „The WEKA data mining software: an update“. In: *ACM SIGKDD explorations newsletter* 11.1 (2009), S. 10–18.

[4] Wes McKinney u. a. *pandas: Python Data Analysis Library*. URL: http://pandas.pydata.org/. Version 0.17.1.

[5] I. Mierswa u. a. „YALE: Rapid Prototyping for Complex Data Mining Tasks“. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)* (Aug. 2006), S. 935–940. URL: http://rapid-i.com/component/option,com_docman/task,doc_download/gid,25/Itemid,62/.

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