

Einsatz von Methoden des Maschinellen Lernens für eine Binäre Klassifikation zur Signal-Hintergrund-Trennung

Bachelorarbeit von

Andreas Zeh-Marschke

An der KIT-Fakultät für Physik
Institut für Experimentelle Teilchenphysik

- 1. Prüfer/Prüferin: Prof. Dr. Günter Quast
- 2. Prüfer/Prüferin: PD Dr. Roger Wolf

15. August 2024 – 17. Februar 2025

Karlsruher Institut für Technologie
Fakultät für Physik
Postfach 6980
76128 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Quellen und Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Eggenstein-Leopoldshafen, 15.01.2025

.....
(Andreas Zeh-Marschke)

Zusammenfassung

Diese Arbeit richtet sich an Studierende im Bachelorstudium, die erste Erfahrungen in der Datenanalyse haben, jedoch noch keine Kenntnisse im Maschinellen Lernen.

Im ersten Teil der Arbeit (Kapitel 1 bis 3) erfolgt eine Einführung in das Maschinelle Lernen anhand einfacher physikalischer Fragestellungen.

- Zuerst wird erläutert was maschinelles Lernen ist und welche Aufgaben damit durchgeführt werden können.
- Danach wird die binäre Klassifikation genauer betrachtet.
- Dabei werden wichtige Begriffe eingeführt und an einfachen Beispielen verdeutlicht.
- Wichtig ist dabei auch der Abgleich zwischen der Sprachwelt im Maschinellen Lernen (dem ML-Jargon) und der Begriffswelt in der physikalischen Welt (Physik-Jargon), damit ein gegenseitiges Verständnis erreicht wird.

Im zweiten Teil der Arbeit (Kapitel 4) wird mit einem einfachen Beispiel aus der Physik eine Trennung von Signalen und Hintergrund durchgeführt. Daten, die mittels eines einfachen Halbleiterdetektors aufgenommen wurden, werden analysiert. Dabei wird das maschinelle Lernen auf die Kontur angewendet, also eine Bilderkennung durchgeführt. Damit wird eine Genauigkeit von etwa 99% erzielt, also ein sehr gutes Ergebnis. Auch auf Kenndaten des Verlaufs der Wellen wird maschinelles Lernen angewendet. Somit werden verschiedene Verfahren beispielhaft angewendet.

Mit einfachen Werkzeugen, einfachen Hilfsmitteln und einfachen physikalischen Fragestellungen wird in die Welt des maschinellen Lernens eingetaucht werden. Daraus können Anregungen für Experimente entstehen. Experimente die beispielsweise auch in einem physikalischen Praktikum angewendet werden können.

Es ist somit eine kleine und nicht vollständige Einführung in das Maschinelle Lernen. Eine vollständige Einführung würde den Rahmen der Bachelorarbeit sprengen. Sie kann Ausgangspunkte für weitere Schritte in der weiten Welt des maschinellen Lernens sein.

Danksagung

Ich danke ganz herzlich Prof. Dr. Günter Quast, der dieses Thema vorgeschlagen und stets konstruktiv und motivierend begleitet hat. Neben ihm gilt auch mein Dank an PD Dr. Roger Wolf, und auch weiteren Forschenden am ETP für die interessanten und fruchtbaren Diskussionen und Anregungen.

Korrigierte Fassung vom 10.02.2025

Inhaltsverzeichnis

Zusammenfassung	i
1. Einleitung	1
1.1. Ausgangslage	1
1.2. Zielsetzung der Arbeit	2
1.3. Aufbau der Arbeit	3
1.4. Verwendete Werkzeuge	4
2. Maschinelles Lernen	5
2.1. Was ist Maschinelles Lernen	5
2.2. Welche Aufgaben können mit Maschinellern Lernen durchgeführt werden?	5
2.3. Die Sprache des maschinellen Lernens	8
2.4. Erstes Verfahren: Lineare Regression	10
3. Binäre Klassifikation - Ablauf und Kennzahlen	11
3.1. Ein ML-Modell bauen	11
3.2. Kennzahlen bei einer Binären Klassifikation	13
4. Signal-Hintergrund-Trennung	17
4.1. Ausgangslage	17
4.2. Rohdaten aufbereiten und bereinigen	18
4.3. Daten anreichern	20
4.4. Bearbeitung mit der Kontur	21
4.5. Bearbeitung mit Kennwerten der Welle	26
4.6. Kontur und die vier Merkmale	31
5. Schlussbemerkungen	33
5.1. Zusammenfassung und erster Ausblick	33
5.2. Einsatz von Verfahren des Maschinellen Lernens	34
5.3. Wie geht es weiter	34
Literatur	35
A. Wörterbuch	37
A.1. Wörterbuch Deutsch - Englisch	37
A.2. Wörterbuch Englisch - Deutsch	39
B. Bilder von Wellen (Signal / Hintergrund)	41

C. Liste von Jupyter Notebooks	43
C.1. Liste von Dateien für die ersten Maschinellen Lernen-Verfahren	43
C.2. Liste von Dateien für die Signal-Hintergrund-Trennung	44

Abbildungsverzeichnis

2.1.	Beispiel für das Gruppieren von Objekten	8
4.1.	CERN-DIY-Detector	17
4.2.	Beispiel einer Welle	17
4.3.	Beispiel der Abtastpunkte eines Signals	18
4.4.	Beispiele von Signal / Hintergrund	19
4.5.	Beispiele Signal- und Hintergrunddaten - mit Kontur	22
4.6.	Beispiele Signal- und Hintergrunddaten - nur Kontur	22
4.7.	ROC-Kurve für die Logistische Regression.	23
4.8.	Lernkurve für ein Neuronales Netz (MLPClassifier) mit einem verborge- nen Layer mit 15 Knoten.	25
4.9.	Beispiele von Wellen, die bei allen drei Verfahren jeweils falsch zugeordnet wurden	27
4.10.	Histogramm PEAK1	28
4.11.	Histogramm P2P_DIFF	28
4.12.	Pairplot	29
4.13.	ROC-Kurve vier Merkmale	30
B.1.	Beispiele von Signal-Hintergrund-Wellen aus dem Datensatz, ohne Kenn- zeichnung, ob es sich um ein Signal oder ein Hintergrund handelt.	41
B.2.	Beispiele von Signal-Hintergrund-Wellen aus dem Datensatz, mit Kenn- zeichnung, ob es sich um ein Signal oder ein Hintergrund handelt.	42

Tabellenverzeichnis

4.1.	Berechnung der Genauigkeit (accuracy) für verschiedene Layerdsigns . .	25
4.2.	Kennzahlen Neuronales Netz - Kontur	26
4.3.	Kenndaten kNN	28
4.4.	Kennzahlen Neuronales Netz - Kontur plus	32
A.1.	Wörterbuch Deutsch - Englisch	37
A.1.	Wörterbuch Deutsch - Englisch, Fortsetzung	38
A.2.	Wörterbuch Englisch - Deutsch	39
A.2.	Wörterbuch Englisch - Deutsch, Fortsetzung	40

Abkürzungen

AUC	Fläche unter der Kurve (area under the curve)
CgDA	Computergestützte Datenauswertung
DPG	Deutsche Physikalische Gesellschaft
DT	Entscheidungsbaum (Decision Tree)
ETP	Institut für Experimentelle Teilchenphysik
FPR	Falsch-Positive-Rate (False-Positive-Rate)
FWHM	Halbwertsbreite, Full Width at Half Maximum
KI	Künstliche Intelligenz
KIT	Karlsruher Institut für Technologie
kNN	k-Nächster-Nachbar
LHC	Large Hadron Collider
LR	Lineare Regression
ML	Maschinelles Lernen
ML	Maschinelles Lernen
NN	Neuronales Netz
ROC	Operationscharakteristik (receiver operating characterictic)
TPR	Wahr-Positiv-Rate (True-Positive-Rate)

1. Einleitung

1.1. Ausgangslage

In der Experimentalphysik werden, grob gesprochen, aus Daten, die mittels Experimenten erfasst werden, Erkenntnisse gewonnen. Dabei entsteht bei vielen Experimenten eine große Masse von Daten. Diese Mengen können nicht mehr per Hand ausgewertet werden. Die Datenanalyse ist daher ein wichtiger Teil in der physikalischen Forschung geworden. Hierbei ist der Einsatz von Computern ein unverzichtbarer Teil. Der Grundsatz *Lernen aus Beispielen* wird angewendet. Dieses **Maschinelle Lernen** (ML), als ein Bestandteil der **Künstlichen Intelligenz** (KI), ist ein wichtiger Baustein in der physikalischen Forschung geworden. In *PHYSIK konkret* Nr. 70 [1] der Deutschen Physikalischen Gesellschaft (DPG) vom Januar 2024 steht:

Mathematische Methoden des maschinellen Lernens sind u.a. als Werkzeuge zur Analyse großer Datensätze zunehmend ein wichtiger Bestandteil der täglichen Arbeit vieler Physiker:innen.

Die besondere Bedeutung des ML in der Physik wird durch die Vergabe des Physik-Nobelpreises 2024 an John Hopfield und Geoffrey Hinton für bahnbrechende Arbeiten im Bereich des ML (siehe [2] oder *PHYSIK konkret* Nr. 73 [3]).

Maschinelles Lernen ist jedoch nicht nur innerhalb der Physik eine wichtige Aufgabe. Auch außerhalb der Physik werden Verfahren der KI und des ML immer häufiger und intensiver gebraucht - manchmal leider auch missbraucht. Daher ist es für alle Physikerinnen und Physiker wichtig, Kenntnisse in diesem Bereich zu haben.

In den verschiedenen Praktika der Physik lernen Studierende, Daten zu erheben und diese auszuwerten. In der klassischen Art und Weise wird diese Auswertung mit Hilfe von per Hand erstellten deterministischen Programmen durchgeführt. Auf Grund des meist geringen Datenvolumens ist dies auch kein größeres Problem. Aus dieser Sicht ergibt sich zuerst keine zwingende Notwendigkeit, dies zu ändern. Erdmann und andere [4] bezeichnen dies als **regelbasiertes System** (*rule based system*). Dies stößt jedoch an Grenzen, wenn beispielsweise die Likelihood-Funktion (siehe [5], Abschnitt 7.1) nicht mehr niedergeschrieben werden kann.

Solche regelbasierten Systeme werden beispielsweise in physikalischen Praktika angewendet. Das physikalische Praktikum hat jedoch nicht nur die Aufgabe, physikalische Experimente durchzuführen und damit physikalische Erkenntnisse zu sammeln. Im Modulhandbuch für das Bachelorstudium Physik an der Fakultät Physik am Karlsruher Institut

für Technologie (KIT) steht in der Beschreibung zum Praktikum klassische Physik I im Qualifikationsziel: „erlangen die Fähigkeit, experimentelle Daten zu erfassen und darzustellen, sowie die Daten zu analysieren“. Daher ergibt sich die Frage oder Herausforderung, wie Maschinelles Lernen in das Praktikum eingebunden werden kann, als ein Verfahren zur Datenauswertung.

Die Künstliche Intelligenz und das Maschinelle Lernen ist keine neue Entwicklung. Die Anfänge dazu gab es schon in den 1940er Jahren.¹ Erst in den letzten Jahren ist die Bedeutung enorm angestiegen. Dies hat mit zwei wichtigen Entwicklung zu tun:

- der enorme Anstieg von gut strukturierten und verfügbaren Daten und
- die enorm gewachsene Rechenleistung der Computer.

Das alleine ist jedoch nicht ausreichend. Klassische Verfahren des Maschinellen Lernens konnten noch nicht viele neue Errungenschaften erzielen. Diese erzielen kaum Verbesserungen gegenüber den deterministischen Verfahren. Die in den letzten Jahren immer stärker auftretenden **Neuronale Netze** (NN) führen zu „Erkenntnisgewinn durch moderne datengetriebene Methoden“ (siehe Erdmann, [6]). Hierbei ist insbesondere das **tiefe Lernen**, bekannt unter dem englischen Begriff **Deep Learning**, hervorzuheben, neuronale Netze, mit vielen Ebenen, Knoten und Verknüpfungen.

1.2. Zielsetzung der Arbeit

Diese Arbeit verfolgt mehrere Ziele:

- Die Grundbegriffe des Maschinellen Lernens werden dargestellt und erläutert. Diese Grundbegriffe werden auch beim Deep Learning verwendet. Daher ist es wichtig, diese Begriffe zu kennen und zu verstehen. Zielgruppe hierfür sind Personen, welche das Thema Maschinelles Lernen bisher noch nicht kennen.
- Es werden einige Verfahren des (klassischen) Maschinellen Lernens vorgestellt. Die Konzepte sind wichtig. Darüber hinaus werden viele Begriffe an einfachen Verfahren dargestellt. Dabei ist es wichtig, eine Verbindung der Begriffe aus der ML-Welt (das ML-Jargon) mit bereits bekannten Begriffen der physikalischen Welt (das Physik-Jargon) zusammen zu bringen. Das ist ein wichtiger Teil, damit die Physik die reichhaltigen Quellen des ML nutzen kann und zum anderen, dass das ML der Physik helfen kann.
- An Beispielen wird aufgezeigt, wie Maschinelles Lernen bei physikalischen Problemen eingesetzt wird.

¹siehe dazu *Meilensteine der Künstlichen Intelligenz*, Physik Journal 19 (2020), Nr. 4, p. 22 - 23

Damit werden in dieser Arbeit Grundlagen des Maschinellen Lernens vermittelt. Die Anwendung in physikalischen Fragestellungen wird beispielhaft demonstriert und dadurch wichtige Begriffe und Verfahren des Maschinellen Lernens vermittelt und erläutert. Mit dieser umfassenden Zielsetzung ist bereits klar, dass dies den Rahmen einer Bachelorarbeit deutlich sprengt. Daher wird die Zielsetzung etwas reduziert:

- Grundlegende Verfahren und Begriffe des ML kennenlernen und in die Sprach- und Begriffswelt der Physik einbetten.
- Anwendung von ML-Verfahren an einfachen physikalischen Beispielen kennenlernen.

Für das Maschinelle Lernen wird damit nur ein Anfang dargeboten. Die Welt des Maschinellen Lernens ist in den letzten Jahren enorm gewachsen. Immer mehr komplexere und aufwändigere Verfahren wurden entwickelt. Die Einsatzmöglichkeiten sind ebenso enorm gewachsen. Bilderkennung, Übersetzungen, autonomes Fahren, medizinische Untersuchungen sind nur einige wenige Stichworte dazu. Daher dürfte klar sein, dass diese Arbeit nur an der Oberfläche kratzt. Es soll aber ein Ausgangspunkt dafür sein, tiefer in die Materie einzudringen. Die Arbeit wendet sich damit an Bachelor-Studierende mit Kenntnissen in Statistik, wie es beispielsweise in der Vorlesung "Computergestützte Datenauswertung" vermittelt wird, jedoch ohne Kenntnisse im Maschinellen Lernen.

1.3. Aufbau der Arbeit

Der Aufbau der Arbeit orientiert sich am Anfang stark an dem Buch von Viviana Acquaviva [7]. Aus diesem Buch werden auch Beispiele aus der Physik und Astronomie entnommen.

Im Kapitel 2 (Maschinelles Lernen) wird erläutert, was unter Maschinellern Lernen zu verstehen ist. Darüber hinaus wird dargestellt, welche Aufgaben bearbeitet werden. Dann wird die Sprache, die beim Maschinellen Lernen verwendet wird, eingeführt. Das sind die grundlegenden Begriffe, die bei jedem Verfahren, egal ob einfache Verfahren oder komplexe Verfahren verwendet werden.

Im Kapitel 3 (Binäre Klassifikation - Ablauf und Kennzahlen) wird der grobe Ablauf beim Maschinellen Lernen dargestellt. Dann werden verschiedene Kennzahlen für die Binäre Klassifikation vorgestellt. Ein Punkt dabei ist die Fragestellung, wie das Ergebnis eines Modells bewertet wird. Die Bewertung ist ein Maß für die Güte, die Qualität des Verfahrens. Die Güte soll möglichst hoch sein.

Das Kapitel 4 (Signal-Hintergrund-Trennung) befasst sich mit einer einfachen Aufgabe. Impulse, die mit einem Detektor empfangen werden, werden analysiert. Es werden Werkzeuge des ML angewendet, um ein Signal vom Hintergrund zu unterscheiden. Dies ist eine zentrale Aufgabe in vielen Experimenten, insbesondere in der Teilchenphysik.

Im abschließenden Kapitel 5 (Schlussbemerkungen) wird ein Ausblick auf den weiteren Weg im Maschinellen Lernen dargestellt. Denn diese Arbeit kann nur ein kleiner Einstieg sein. Es gibt noch viele Methoden und mächtige Werkzeuge, die im ML verwendet werden.

1.4. Verwendete Werkzeuge

Das Lernen von Maschinellern Lernen kann nicht als Trockenkurs durchgeführt werden. Wichtig ist es, die Verfahren zu implementieren, auszuprobieren und damit zu „spielen“, also bewusst Veränderungen vorzunehmen und zu beobachten, was sich daraus ergibt. Dazu gibt es einige hilfreiche Werkzeuge. Es werden nun einige Werkzeuge kurz aufgeführt, die verwendet werden.

- **Python** ([8], [9]): Dazu schreibt VanderPlas ([10], Einleitung): „Python hat sich in den vergangenen Jahrzehnten zu einem erstklassigen Tool für wissenschaftliche Berechnungen entwickelt, insbesondere auch für Analyse und Visualisierung großer Datensätze“. Mit Python muss auch **Jupyter Notebook** ([11], [12]) erwähnt werden, denn die Implementierung werden zum Großteil mit Hilfe von Jupyter Notebooks realisiert.
- **NumPy** ([13], [14]): Diese Programmbibliothek bietet vieles für die Benutzung von (auch sehr großen) Vektoren, Matrizen und mehr-dimensionalen Datenstrukturen von numerischen Daten. Das sind die mathematischen Gebilde, die wichtig in der Bearbeitung im Maschinellen Lernen sind.
- **Pandas** ([15]): Diese Programmbibliothek kann verschiedenartige Daten bearbeiten. Es ist daher universeller als Numpy, da es auch nicht numerische Daten bearbeitet. In dieser Arbeit wird davon ausgegangen, dass die Daten sauber zur Verarbeitung zur Verfügung stehen, was in der Realität selten der Fall ist. An wenigen Stellen werden mit Hilfe von Pandas Daten aufbereitet und dann als NumPy-Array zur Verfügung gestellt.
- **Matplotlib** ([16], [17]): Mit der Programmbibliothek ergeben sich viele Möglichkeiten zur Visualisierung von Datenmengen.
- **Seaborn** ([18], [19]): Diese Programmbibliothek baut auf Matplotlib auf, ist jedoch an vielen Stellen moderner und einfacher anzuwenden als Matplotlib.
- **Scikit-Learn** (kurz **sklearn**) ([20], [21]): ist eine Programmbibliothek in Python für das Lernen von Maschinellern Lernen. Die Algorithmen, die in dieser Arbeit vorgestellt werden, sind in Scikit-Learn implementiert. Teilweise wird eventuell zuerst ein Algorithmus per Hand implementiert, um das Verständnis für den Algorithmus zu fördern. Später wird dann Scikit-Learn als Black-Box-Bibliothek verwendet.

Nicht verwendete Werkzeuge, die jedoch später, für die weitere Bearbeitung, insbesondere im Deep Learning, wichtig und hilfreich sind: **PyTorch** [22] und **TensorFlow** [23]. Große Deep Learning Netze können damit gestaltet und bearbeitet werden. Für die Anfänge reicht **scikit-learn**. Auch damit kann bereits viel durchgeführt werden.

2. Maschinelles Lernen

2.1. Was ist Maschinelles Lernen

Es gibt keine klare eindeutige Definition für den Begriff *Maschinelles Lernen*. Er gibt nur einige Versuche zu erklären, was unter Maschinellern Lernen verstanden wird. **Maschinelles Lernen** ist nach Acquiaviva [7] „der Prozess einer Maschine beizubringen, fundierte, datengestützte Entscheidungen zu treffen“ („the process of teaching a machine to make informed, data-driven decisions“).

Der enorme Anstieg der Bedeutung von ML ist in den letzten Jahren durch zwei wichtige Entwicklungen vorangetrieben worden:

- die Masse von strukturiert vorliegenden Daten ist enorm angewachsen und
- die Rechenleistung der Computer ist angestiegen.

Gemäß VanderPlas [10] basiert ML auf *mathematisch-statistischen Kenntnissen*, um riesige Datenmengen zu modellieren und *Fähigkeiten als Programmierer*, um mit riesigen Datenmengen in der Speicherung, der Verarbeitung und der Visualisierung umzugehen. Dies alleine reicht jedoch nicht. Er führt weiter aus, dass auch *fundierte Fachwissen* benötigt wird, damit die richtigen Fragen gestellt werden können und damit die Antworten klar interpretiert werden können.

Maschinelles Lernen ist dann geeignet, wenn die klassischen Verfahren nicht ausreichen, wenn beispielsweise die Likelihood-Funktion (siehe [5]) nicht bekannt ist oder nicht aufgeschrieben werden kann.

2.2. Welche Aufgaben können mit Maschinellern Lernen durchgeführt werden?

Acquiaviva [7] beschreibt die Aufgaben des ML folgendermaßen:

1. **erkennen** (*recognize*): korrektes Erkennen von Mustern oder Bildern,
2. **vorhersagen** (*predict*): zukünftiges Verhalten vorhersagen, vervollständigen fehlender Daten,
3. **gruppieren** (*group together*): Objekte vergleichen und zu Gruppen anordnen und

4. **vereinfachen** (*simplify*): Daten zusammenfassen und kondensieren, so dass sie besser verständlich und einfacher verarbeitbar sind.

Die Verfahren des ML werden eine dieser Aufgaben erfüllen. Die Verfahren des ML lassen sich gemäß Frochte [24] oder Nguyen und Zeigermann [25] in drei Kategorien einteilen:

- **überwachtes Lernen** (*supervised learning*),
- **bestärkendes Lernen** (*reinforcement learning*) und
- **unüberwachtes Lernen** (*unsupervised learning*).

In allen Verfahren geht es darum, eine Funktion $f : X \rightarrow Y$ zu finden, welche Elemente aus einer Eingabemenge X auf Elemente einer Ergebnis- oder Zielmenge abbilden. Auf Grund von Messfehlern, statistischen Fehlern und ähnlichen Sachverhalten ist es möglich, dass für gleiche Werte in der Eingabemenge unterschiedliche Ergebnisse zugeordnet werden. Die Funktion muss dann so konstruiert sein, dass möglichst gute Ergebnisse bestimmt werden, so dass die Daten gut wiedergegeben werden. Was dies bedeutet, wie dies definiert wird, das folgt später.

2.2.1. Überwachtes Lernen

Beim überwachten Lernen gibt es (ausreichend viele) Datensätze von Eingabewerten, bei denen die dazu gehörigen Ergebniswerte bekannt sind. Aufbauend auf diesen Daten wird ein Algorithmus trainiert. Das bedeutet, dass Parameter für den Algorithmus bestimmt werden. Der trainierte Algorithmus kann dann auf noch unbestimmte Eingabewerte angewendet werden, also ein Ergebnis bestimmt, vorhergesagt werden.

Bei der **Klassifikation** (*classification*) ist die Ergebnismenge eine diskrete Menge. Die Elemente werden **Klasse** genannt. Der Algorithmus soll auf Basis der Eingabedaten eines Datensatzes **erkennen**, zu welcher Klasse die Daten gehören. Beispiele für Klassifikationen sind:

- Das Erkennen von Iris-Blüten in dem berühmten Irisdatensatz des Botanikers *Edgar Anderson*. Hier soll anhand von vier Merkmalen entschieden werden, zu welchen von drei Arten eine Blüte gehört.
- Das Erkennen von Ziffern aus handschriftlichen Notizen. Hier gibt es zehn verschiedene Zeichen, die erkannt werden sollen.
- Das Erkennen von möglicherweise bewohnbaren Planeten. Hier gibt es zwei verschiedene Möglichkeiten: bewohnbar oder nicht bewohnbar.
- Das Erkennen von bestimmten Zerfallsprodukten oder -prozessen im LHC (Large Hadron Collider) bei einer Proton-Proton-Kollision bei hoher Energie.

Gibt es nur zwei Klassen, dann ist dies eine **binäre Klassifizierung** (*binary classification*). Gibt es mehr als zwei Klassen, dann ist das eine **Multi-Klassen Klassifizierung** (*multi-class classification*). Jedes Klassifizierungsproblem mit n Klassen ($\{c_1, \dots, c_n\}$) kann in n binäre Klassifizierungsprobleme ($\{c_i, \text{nicht-}c_i\}$) umgewandelt werden.

Bei der **Regression** (*regression*) ist die Zielmenge eine kontinuierliche Menge. Hier wird eine Funktion gesucht, welche von neuen Daten in der Eingabemenge ein Ergebnis **vorhersagt**. Dazu zählen beispielsweise:

- Die Bestimmung von Steigung und Achsenabschnitt für eine lineare Regression.
- Die Rekonstruktion fehlender Teile in einem Bild,
- Auf Basis von einigen Daten Werte für andere Daten vorhersagen.

2.2.2. Einschub: eifriges und faules Lernen

Beim ML ist der Lernprozess wichtig, um das Erlernte auf neue Situationen anzuwenden. Dabei gibt es zwei verschiedene Arten.

- Beim **eifrigen Lernen** (*eager learning*) ist der Trainingsprozess aufwändig. Die Trainingsphase kann Stunden und Tage dauern, um eine optimale Funktion, ein globales Modell, zu bestimmen. Das Ziel ist, dass die Anwendung des Gelernten schnell geht. Beim Protonen-Protonen-Stoß im LHC entstehen viele Zerfallsprodukte. In den Detektoren muss sehr schnell entschieden werden, ob der Zerfall ein interessantes Ereignis (*signal*) oder ein nicht interessantes Ereignis (*background, noise*) ist.
- Beim **faulen Lernen** (*lazy learning*) ist die Trainingsphase sehr kurz oder nicht vorhanden. Die Bearbeitung erfolgt erst, wenn die neuen Eingabedaten kommen. Es wird ein lokales Modell erstellt. Der Trainingsaufwand ist gering, der Aufwand für die Anwendung dagegen groß.

Faules Lernen hat den Vorteil, dass lokale Gegebenheiten der Daten passgenauer bearbeitet werden. Beim eifrigen Lernen werden globale Modelle erstellt, die lokal zu Qualitätsverlusten führen können.

2.2.3. Bestärkendes Lernen

Beim bestärkenden Lernen wird beim Lernprozess durch positive oder negative Rückmeldungen nach einer Aktion die Optimierung eines Verhaltens gefördert. Beispiele sind hierfür: Lernen von Spielen, Robotersteuerung oder autonomes Fahren

2.2.4. Unüberwachtes Lernen

Beim unüberwachten Lernen sind keine Ergebnisse zu den Eingabedaten bekannt.

Beim **Clustering** (*clustering*) werden Daten verglichen und versucht, Daten zu **gruppieren** (*group together*). Es ergibt sich also die Frage, welche Eigenschaften dazu führen, dass verschiedene Objekte als Objekte einer Gruppe angesehen werden. Es gilt also Gemeinsamkeiten und Trennendes von Daten zu erkennen. Dazu gehören beispielsweise verschiedene Typen von Galaxien an Hand von Bildern zu erkennen.

Im Abbildung 2.1 sind vier Objekte dargestellt, die gruppiert werden sollen. Es können ohne Probleme zwei verschiedene Gruppierungen erstellt werden: (1) Die Gruppe der Quadrate und die Gruppe der Kreise oder (2) Die Gruppe der gefüllten Objekte und die Gruppe der nicht gefüllten Objekte.

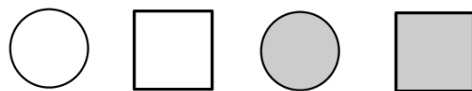


Abbildung 2.1.: Beispiel für das Gruppieren von Objekten

Beide Gruppierungen sind korrekt. Welche Gruppierung ist jedoch die „gewünschte“? Dies zeigt, dass das Clustering nicht so einfach ist. Was passiert mit dem Clustering bei diesem kleinen Beispiel, wenn eines der vier Objekte aus dem Bild entfernt wird?

Bei der **Reduktion von Dimensionen** (*dimensionality reduction*) geht es darum, Daten zu **vereinfachen** (*simplify*). Dazu kann die Reduktion der Dimensionen der Daten gehören, um die Daten handhabbarer, anschaulicher oder einfacher verarbeitbar zu machen. Beispielsweise können Datenpunkte eines Kreises im 2-dimensionalen Raum mittels zweier Parameter (x - und y -Koordinaten) oder mittels eines Parameters (Winkel - bei gegebenen Radius) dargestellt werden.

2.3. Die Sprache des maschinellen Lernens

2.3.1. Instanzen, Eigenschaften, Ergebnis und Modell

Für das ML werden Daten herangezogen und analysiert. Die Daten bestehen aus N **Instanzen** (*instances*). Die Instanzen werden manchmal auch kurz Beispiele (*samples*, *examples*) oder in der Physik Beobachtungen (*observations*) genannt. Diese Instanzen bestehen aus n bekannten **Eigenschaften** (*features*) auch die *Input*-Daten genannt. Diese Eigenschaften können numerische Werte sein, müssen es jedoch nicht. Für die weitere Verarbeitung ist es jedoch notwendig, wenn alle Daten numerisch sind. Hierzu kann eine einfache Abbildung von nicht-numerischen Werten auf numerische Werte durchgeführt werden. Darüber hinaus gibt es eine Eigenschaft, die aus den *Input*-Daten abgeleitet werden soll. Dieses Datum heißt **Ergebnis** (*target*), das auch *Output*-Datum genannt wird.

Damit sind die Input-Daten für alle Instanzen X eine reelle $N \times n$ -Matrix, $X = (x_{i,j}) \in \mathbb{R}^{N \times n}$. In jeder der N Zeilen stehen die Eigenschaften einer Instanz. In jeder einzelnen der n Spalten stehen die Daten für eine Eigenschaft. Die Output-Daten für alle Instanzen bilden einen reellen Vektor der Länge N : $\mathbf{y} = (y_i)^T \in \mathbb{R}^N$. In ihm stehen die Ergebnisse der einzelnen Instanzen.

Bei klassischen Programmen wird eine Funktion geschrieben, welche die Eingabedaten in die Ausgabedaten transformiert. Hierzu muss die Funktion jedoch bekannt sein. Beim ML

hingegen ist die Funktion oder sind die Parameter der Funktion nicht bekannt. Es soll ein möglichst gutes **Maschinelles Lernen Modell** (*machine learning model*) erstellt - gelernt - werden, welches die Output-Daten aus den Input-Daten ermittelt. Dann wird auch für unbekannte Datensätze aus gegebenen Input-Daten die Output-Daten vorhergesagt.

2.3.2. Lerndaten, Trainingsdaten, Testdaten

Beim überwachten Lernen wird die Sammlung von Daten, die für das Lernen zur Verfügung stehen als **Lerndaten** (*learning set*) bezeichnet. Für diese Lerndaten sind die Input-Daten und die dazugehörigen Output-Daten bekannt. Diese Lerndaten werden in zwei Teile zerlegt:

- Mit den **Trainingsdaten** (*training set*) wird ein Modell und die dazugehörigen Parameter berechnet, welche die Beziehung zwischen den Input-Daten und den Output-Daten beschreibt.
- Mit den **Testdaten** (*test set*) wird das Modell und die bestimmten Parameter überprüft. Es wird somit geprüft, ob das Modell, das aus den Trainingsdaten bestimmt wird, die Testdaten gut beschreibt. Was dabei „gut beschreibt“ bedeutet, das muss definiert werden.

Die N Instanzen werden aufgeteilt in N_{train} Trainingsdaten ($X_{train}, \mathbf{y}_{train}$) und N_{test} Testdaten ($X_{test}, \mathbf{y}_{test}$). Dies ist ein zufälliger Prozess. Eine gebräuchliche Aufteilung ist 70% Trainingsdaten und 30% Testdaten. Diese Werte können jedoch variieren.

Wichtig dabei ist die zufällige Auswahl der Trainingsdaten, damit die Trainingsdaten ein repräsentatives Bild der gesamten Daten abgeben! Nach Acquaviva [7] ist das überwachte Lernen nur so gut, wie die Daten, die in den Lerndaten enthalten sind (*a supervised learning method is only as good as its learning set*):

- Gibt es zu wenig Daten, dann kann das Modell den Zusammenhang zwischen Input und Output nicht richtig erlernen.
- Ist die Auswahl der Daten in den Lerndaten nicht repräsentativ für alle Daten, dann lernt das Modell einen falschen Zusammenhang.

Daher ist es wichtig, die Daten zu verstehen. Dazu wird das Fachwissen benötigt, um die Daten richtig zu interpretieren. Acquaviva [7] schreibt dazu: „Because [machine learning techniques] are driven by the data as opposed to relying on physical intuition, we are bound to make a fool of ourselves if we don't understand the data well.“

2.3.3. Fehlerrate, Erfolgsrate

Für die Überprüfung, ob das gewählte Modell eine gute Beschreibung der Beziehung zwischen Input-Daten und Output-Daten ist, wird die **Fehlerrate** (*rate of failure (error)*)

beziehungsweise die **Erfolgsrate** (*rate of success (score)*) bestimmt. Dies wird sowohl bei den Trainingsdaten als auch bei den Testdaten durchgeführt. Damit werden

- **Trainingsfehlerrate** (*training error*), **Trainingserfolgsrate** (*training score*) und
- **Testfehlerrate** (*test error*), **Testerfolgsrate** (*test score*)

ermittelt. Wenn das Modell auf Daten angewendet wird, die nicht in den Lerndaten enthalten sind, dann werden ebenso Fehler und Erfolg gemessen. Dies wird dann **Generalisierungsfehler** (*generalization error*) beziehungsweise **Generalisierungserfolg** (*generalization score*) genannt.

Durch den Vergleich von diesen Kennzahlen (später werden noch weitere Kennzahlen eingeführt) werden die Ergebnisse verschiedener Modelle und auch unterschiedlicher Parametrisierungen von Modellen verglichen. Damit wird abgewogen, welche Modelle und welche Parameter für die Modelle geeignet sind.

2.4. Erstes Verfahren: Lineare Regression

Im Jupyter Notebook `cha02-MaschinellesLernen.ipynb` ist ein erstes Verfahren dargestellt, eine **Lineare Regression** (**LR**, *Linear Regression*). Die eingeführten Begriffe des Maschinellen Lernens werden angewendet. Dabei ergibt sich die Frage, ob die Lineare Regression überhaupt ein ML-Algorithmus ist oder zum Bereich statistisches Lernen gehört. Wenn n Datenpunkte gegeben sind, dann kann dafür leicht eine Ausgleichsgerade nach der Methoden der kleinsten Quadrate (siehe [5]) bestimmt werden. Es stellt sich die Frage, wann Verfahren des ML geeignet eingesetzt werden können, wann aber das direkte Berechnen des Modells geeigneter ist. Dazu schreibt Acquaviva [7] über Vor- und Nachteile der beiden Verfahren:

Maschinelles Lernen

- ML ist datengetrieben und daher nur so gut wie die Daten.
- Eine Verallgemeinerung ist nur schwer möglich, Modell nicht blindlings auf andere Daten anwendbar.
- Eine Interpretation der Ergebnisse ist möglich, aber nicht einfach.
- ML ist schnell.
- Robust gegenüber fehlenden Daten.
- Es ermöglicht glückliche Entdeckungen.

Modell direkt fitten

- Dies ist getrieben durch Intuition oder Modell und daher nur so gut wie das vorhandene Wissen.
- Verallgemeinerung gut möglich, wenn die Physik gut verstanden ist.
- Einfacher zu interpretieren.
- Es ist teilweise sehr rechenintensiv.
- Der Umgang mit heterogenen Daten oftmals schwierig.
- Führt zu Informationsverlust, wenn das Modell zu einfach ist.

3. Binäre Klassifikation - Ablauf und Kennzahlen

In diesem Abschnitt wird zuerst (Abschnitt 3.1) der Ablauf für die Durchführung eines ML-Algorithmus vorgestellt. Der allgemeine Ablauf gilt grundlegend für alle ML-Algorithmen. Anschließend (Abschnitt 3.2) werden wichtige Kennzahlen für eine Bewertung bei einer binären Klassifikation aufgeführt. Für eine Multi-Klassifikation können diese ausgebaut werden. Für eine Regression werden andere Bewertungsmaße ermittelt, welche beispielsweise schon bei einer Linearen Regression (siehe Jupyter Notebook cha02-MaschinellesLernen.ipynb angewendet wurden.

3.1. Ein ML-Modell bauen

Wie sieht der Ablauf beim Maschinellen Lernen aus, um ein ML-Modell aufzubauen? Der erste grobe Ablauf orientiert sich an dem von Acquviva [7] beschriebenen Ablauf. Er besteht aus verschiedenen Schritten, die sukzessive durchgeführt werden.

Schritt 1: vorbereiten / prepare

Daten aufbereiten: Die vorhandenen Daten werden zuerst angesehen und aufbereitet. Dazu gehört auch der Umgang mit fehlenden Daten und die Kombination der Daten, um neue Kennzahlen zu erzeugen. Hierbei ist es wichtig, die Daten fachlich zu verstehen, damit sinnvolle Anpassungen durchgeführt werden. Ziel ist es, dass die Daten als numerisch verarbeitbare Daten zur Verfügung stehen. Für die Darstellung und Aufbereitung wird oftmals pandas verwendet. Hier wird pandas nur sehr spärlich verwendet. Ziel ist, dass die Daten als numpy-Array für die weitere Verarbeitung zur Verfügung stehen.

splitten: Die Daten werden aufgeteilt in Trainingsdaten und Testdaten. Damit stehen am Ende dieses Schrittes die Trainingsdaten X_{train} , y_{train} und die Testdaten X_{test} , y_{test} zur Verfügung.

Schritt 2: auswählen / select

In diesem Schritt wird der ML-Algorithmus / das ML-Modell gewählt, der für das Training angewendet wird. Dabei werden auch die Parameter für den Algorithmus festgelegt.

Beispiel: Soll ein Polynom bestimmt werden, dann ist festzulegen, welchen Grad das Polynom haben soll. Der Grad des Polynoms ist dann ein **Hyperparameter**, der bestimmt werden sollen.

Es gibt sehr viele ML-Algorithmen, die ausgewählt werden können. Es können hier nur einige wenige vorgestellt werden. Grundlegende Begriffe, wie Trainingsdaten, Testdaten, Fehlerrate, Erfolgsrate und weitere Begriffe, die im folgenden noch besprochen werden sind jedoch für alle gleich.

Hinweis: Auch die Bestimmung der geeigneten Hyperparameter ist ein aufwändiger Lernprozess. Es gibt teilweise sehr viele verschiedene Einstellungen für die einzelnen Algorithmen. Variiert man die Hyperparameter, dann ergeben sich andere Funktionen, mit anderen Bewertungen. Auch die Hyperparameter können mit Hilfe von ML bestimmt werden. Dann werden weitere Daten verwendet, um die Leistungen verschiedener ML-Modelle zu vergleichen.

Schritt 3: trainieren / train Der ausgewählte Algorithmus wird auf die Trainingsdaten angewendet. Das Training des Modells wird durchgeführt (*fit*). Dies generiert dann eine vorläufige Beziehung zwischen den Input- und Outputdaten. Damit wird also der Zusammenhang, gemäß dem gewählten Algorithmus beschrieben. Dies ist in der Regel ein sehr rechenintensiver Vorgang.

Schritt 4: anwenden / apply Das trainierte Modell, wird auf die Testdaten angewendet (*predict*). Aus den Inputdaten der Testdaten werden Vorhersagen für die Output-Daten \hat{y}_{test} erstellt. Das sind die Vorhersagewerte auf Basis des trainierten Modells.

Schritt 5: bewerten / evaluate Auf Basis eines vorab bestimmten **Bewertungsmaßstabes** wird die Performance des Modells auf die Daten ermittelt. Hierzu findet ein Vergleich der Vorhersagewerte \hat{y}_{test} mit den echten Werten y_{test} statt. Aus den Differenzen wird eine Bewertung erstellt.

Schritt 6: analysieren / analyse Es kann sein, dass das erzielte Ergebnis gut ist - das ist jedoch in der Regel unwahrscheinlich. Oftmals muss man überlegen, was verbessert werden kann oder muss. Soll ein anderer Algorithmus verwendet werden, sollen die Hyperparameter für den Algorithmus anders gewählt werden, werden mehr Daten benötigt, was funktioniert nicht so gut, was muss besser gestaltet werden. Daraufhin kann der ML Algorithmus geändert oder verändert werden und der Ablauf wiederholt sich.

Eine wichtige und oftmals wiederkehrende Aufgabe, nicht nur in der Physik, ist eine **Binäre Klassifikation** (engl. *Binary Classification*). Beispiele hierfür sind:

- Ist in einem Bild ein Hund zu sehen oder nicht.
- Ist eine Person kreditwürdig oder nicht.
- Trennung bei physikalischen Experimenten zwischen Signal (*signal*) und Untergrund (*background*)
- Ist ein Planet potenziell bewohnbar (*habitable*) oder nicht.

Im Jupyter Notebook cha03-BinClass-ErsteModelle werden zwei ML-Algorithmen vorgestellt am Beispiel über Exoplaneten angewendet, wie es auch bei Acquaviva [7] durchgeführt wird. Die beiden ML-Algorithmen sind **Entscheidungsbaum** (*Decision Tree, DT*) und **k-Nächster Nachbar** (*k-Nearest Neighbour, kNN*).

3.2. Kennzahlen bei einer Binären Klassifikation

Im Jupyter Notebook `cha04-BinClass-Bewertung.ipynb` wird am Beispiel von (potenziell) bewohnbaren Exoplaneten die Ermittlung der Kennzahlen demonstriert.

3.2.1. Genauigkeit, Präzision und Trefferquote

Die **Genauigkeit** (*accuracy*) wurde bereits in vorherigen Untersuchungen berechnet. Es ist der Quotient zwischen korrekten Vorhersagen und allen Elementen. Beim Beispiel der bewohnbaren Planeten gibt es 70 (potenziell) bewohnbare Planeten von insgesamt 5350 untersuchten Planeten. Wenn automatisch jeder Planet als nicht bewohnbar vorhergesagt wird (*faule Zuordnung*), dann ist die Genauigkeit bei $(5350 - 70)/5350 = 0.987$, also bei 98.7%. Ein scheinbar hervorragender Wert. Dieser Wert ergibt sich jedoch, da es fast keine bewohnbare Planeten gibt. Daher ist bei dieser extrem unausgeglichene Menge die Aussagekraft der Kennzahl Genauigkeit sehr gering. Es bedarf weiterer Kennzahlen.

Bei einer binären Klassifikation gibt es für jedes Element zwei mögliche Klassen und zwei mögliche Vorhersagen. Die beiden Klassen werden mit positiv (*P*) und negativ (*N*) bezeichnet. Die positive Klasse ist die Basis- oder **Grundklasse** (*ground truth*). Am Beispiel der bewohnbaren Planeten stehe *P* für bewohnbar und *N* für nicht bewohnbar. Die Vorhersage für die Elemente können wahr (*True, T*) sein, die Klasse wurde richtig vorhergesagt, oder falsch (*False, F*) sein, die Klasse wurde nicht richtig vorhergesagt. Daher gibt es vier Kennzahlen:

- *TP* (true positive): die Anzahl der Elemente, bei denen das positive Merkmal korrekt vorhergesagt wurde (korrekt klassifiziert als positiv),
- *TN* (true negative): die Anzahl der Elemente, bei denen das negative Merkmal korrekt vorhergesagt wurde (korrekt klassifiziert als negativ),
- *FP* (false positive): die Anzahl der Elemente, die falsch als Positiv vorhergesagt wurden (nicht korrekt klassifiziert als positiv) und
- *FN* (false negative): die Anzahl der Elemente, die falsch als Negativ vorhergesagt wurden (nicht korrekt klassifiziert als negativ).

Somit betrachtet der zweite Buchstabe (*P / N*) die Klassifikation durch das ML-Modell. Der erste Buchstabe (*T / F*) besagt, ob die getroffene Klassifizierung korrekt (*T*) oder nicht korrekt (*F*) ist.

Die **Genauigkeit** (*accuracy*) ist damit definiert durch

$$accuracy = \frac{TN + TP}{TP + TN + FP + FN}.$$

Sie wird manchmal mit **acc** abgekürzt.

Es werden zwei weitere wichtige Kennzahlen definiert.

Die **Präzision** (*precision*)

$$precision = \frac{TP}{TP + FP}$$

ist der Anteil der Elemente in der positiven Klasse, die korrekt vorhergesagt wurden. Sie wird manchmal mit **pre** abgekürzt. Diese Kennzahl ist auch als **Reinheit** (*purity*) bekannt.

Die **Trefferquote** (*recall*)

$$recall = \frac{TP}{TP + FN}$$

ist der Anteil der korrekt vorhergesagten Elemente in der Menge, die der positiven Klasse angehören. Sie wird manchmal mit **rec** abgekürzt. Diese Kennzahl ist auch als **Sensitivität** (*sensitivity*) oder **Effizienz** (*efficiency*) bekannt.

Bei dem obigen Beispiel der bewohnbaren Planeten mit der *faulen Zuordnung* ist die Präzision undefiniert, da $TP + FP = 0$ ist. Die Trefferquote ist $0 = 0 / (0 + 70)$. Das zeigt, dass die *faule Zuordnung* zwar eine hohe Genauigkeit hat, aber nicht brauchbar ist.

Wünschenswert ist ein ML-Modell, das eine sehr hohe Präzision und eine sehr hohe Trefferquote hat. Am Besten jeweils 1. Dann sind FP und FN jeweils Null, da alle Elemente korrekt vorhergesagt werden. Damit ergibt sich für die Genauigkeit der Wert 1.

Hinweis: Präzision und Trefferquote sind nicht symmetrisch. Werden die positive und negative Klasse getauscht, dann ergeben sich (mit den bisherigen Bezeichnungen) $precision = TN / (TN + FN) = 1$ und $recall = TN / (TN + FP) = 0.987$. Dies sind komplett andere Werte. Daher ist es wichtig darauf zu achten, welches die positive Klasse ist!

3.2.2. ROC und AUC

Bei der binären Klassifikation ist das Ergebnis der Klassifikation ein Wert, der ohne Beschränkung der Allgemeinheit als 1 für *Positiv* und 0 für *Negativ* bezeichnet wird. Manchmal erfolgt auch die Zuordnung 1 und -1. Oftmals ist die Entscheidung, welche Klasse gewählt wird nur eine Aussage mit einer bestimmten Wahrscheinlichkeit. Es sei $P(Class = Positiv) = p$ die Wahrscheinlichkeit, dass die Klasse *Positiv* erkannt wird. Für die Entscheidung, dass die Klasse *Positiv* gewählt wird, wird ein **Schwellwert** (*threshold*) t ($0.0 \leq t \leq 1.0$) bestimmt. Ist $p \geq t$, dann wird die Klasse *Positiv* festgelegt, ansonsten die Klasse *Negativ*. In der Regel ist der Schwellwert $t = 0.5$.

Wie verändern sich die Kennwerte, insbesondere TP , TN , FP und FN , wenn der Schwellwert verändert wird?

Wenn der Schwellwert verringert wird, dann wird die Klasse P öfters festgelegt. Auch einige P -Objekte, die bisher nicht als korrekt erkannt wurden, werden jetzt korrekt zugeordnet. Das bedeutet, dass TP ansteigt und FN sinkt. Aber auch einige N -Objekte, die bisher

richtig klassifiziert wurden, werden als P -Objekte erkannt. Damit steigt FP . Im Extremfall, wenn der Schwellwert bei 0.0 liegt, wird alles als positiv klassifiziert.

Wenn der Schwellwert vergrößert wird, dann wird die Klasse P seltener festgelegt. Auch einige P -Objekte, die bisher als korrekt erkannt wurden, werden jetzt nicht mehr korrekt zugeordnet. Das bedeutet, dass TP sinkt und FN steigt. Aber auch einige N -Objekte, die bisher falsch als P -Objekte klassifiziert wurden, werden jetzt als N -Objekte richtig klassifiziert. Damit sinkt FP . Im Extremfall, wenn der Schwellwert bei 1.0 liegt, wird alles als negativ klassifiziert.

Veränderungen am Schwellwert verändern somit die Kennwerte TP , TN , FP und FN . Dies hat Auswirkungen auf die Präzision und die Trefferquote. Wird der Schwellwert kleiner, dann geht die Präzision gegen 0.0, die Trefferquote gegen 1.0. Wird der Schwellwert größer, dann geht die Präzision gegen 1.0, die Trefferquote gegen 0.0.

Nur eine Metrik (Genauigkeit, Präzision oder Trefferquote) zu verwenden ist nicht ausreichend. Daher werden neue Kennzahlen eingeführt, wie dies beispielsweise bei Tom Fawcett [26] dargestellt ist.

Die **Wahr-Positiv-Rate** (*true positive rate*) oder kurz TP -Rate oder **TPR**,

$$TPR = \frac{TP}{TP + FN}$$

ist der Anteil der korrekt als P -Objekt klassifizierten Objekte unter allen P -Objekten, also die Trefferquote. Diese Kennzahl ist auch als α -**Fehler** bekannt.

Die **Falsch-Positiv-Rate** (*false positive rate*) oder kurz FP -Rate oder **FPR**

$$FPR = \frac{FP}{FP + TN}$$

ist der Anteil der falsch als P -Objekt klassifizierten N -Objekte unter allen N -Objekten. Diese Kennzahl ist auch als $1 - \beta$ -Fehler bekannt.

In der Statistik werden TPR , also die Trefferquote, als **Sensitivität** (*sensitivity*) und $1 - FPR$ als **Spezifität** (*specificity*) bezeichnet.

Es wird nun die Funktion $TPR(FPR)$ gebildet, also die Wahr-Positiv-Rate in Abhängigkeit von der Falsch-Positive-Rate gebildet. Der Definitionsbereich ist $[0.0, 1.0]$, ebenso der Wertebereich. Die Funktion verläuft vom Punkt $(0, 0)$ zum Punkt $(1, 1)$. Die Funktion ist monoton wachsend. Sie heißt **Operationscharakteristik** (*receiver operating characteristic curve*), kurz **ROC-Kurve**. Ein Beispiel einer ROC-Kurve ist in Abbildung 4.7 zu sehen.

Ideal ist, wenn die FP -Rate = 0 und gleichzeitig die TP -Rate = 1 ist. Das entspricht dem Punkt links oben im Quadrat. Die Kurve hat dann den Verlauf von $(0, 0)$ über $(1, 0)$ nach $(1, 1)$. Je näher die tatsächliche Kurve an dieses Ideal herankommt, desto besser ist das Modell. Bei einem schlechten Klassifizierer ist die Kurve mehr an der Diagonalen (siehe [7], Abschnitt 3.2). Daher wird ein weiteres Maß eingeführt, Die **Fläche unter der Kurve** (*area under the curve*) AUC ist die Fläche unter der ROC-Kurve. Dieser Wert summiert die Leistung eines Modells. Der Wert ist zwischen 0.0 und 1.0. Der Wert 1.0 entspricht einem idealen Modell.

3.2.3. F-Metrik

Trefferquote und Präzision kann in eine gemeinsame Metrik zusammengefasst werden. Das ungewichtete harmonische Mittel

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2TP}{2TP + FN + FP}$$

heißt F_1 -**Metrik** (F_1 -score). Dabei werden beide ursprünglichen Metriken gleich gewichtet. Ist die Präzision oder die Trefferquote gering, dann ist die F_1 -Metrik klein. Ist **FN** oder **FP** hoch, dann ist die F_1 -Metrik klein. Die F_1 -Metrik ist hoch, wenn Präzision **und** Trefferquote gut sind, wenn also **FP** und **FN** klein sind. Werden die beiden ursprünglichen Metriken unterschiedlich gewichtet, dann entsteht die F_β -**Metrik** (F_β -score)

$$F_\beta = \frac{(1 + \beta^2) \cdot precision \cdot recall}{\beta^2 precision + recall} = \frac{1 + \beta^2}{(1 + \beta^2)TP + FP + \beta^2 FN} \cdot$$

Für $\beta = 1$ ergibt sich die F_1 -Metrik. Für $\beta > 1$ wird die Präzision höher gewichtet (um das β -fache). Für $\beta < 1$ wird die Trefferquote höher gewichtet.

3.2.4. Auswahl einer Metrik

Es wurden nun verschiedene Metriken vorgestellt: Genauigkeit, Präzision, Trefferquote, Wahr-Positiv-Rate, Falsch-Positiv-Rate, ROC und AUC. Welche davon ist die richtige Metrik?

Für das Beispiel der bewohnbaren Planeten ist die Genauigkeit eine ungeeignete Metrik, da die Menge sehr unausgewogen ist. Die beiden Metriken Präzision und Trefferquote reagieren unterschiedlich auf Fehlklassifizierungen. Zehn falsch klassifizierte nicht-bewohnbare Planeten sind nur etwa 0.2 % der gesamten Menge der nicht-bewohnbaren Planeten. Bei den bewohnbaren Planeten entsprechen zehn falsch klassifizierte Planeten etwa 14 %.

Die Maximierung der Präzision ist eine geeignete Metrik, wenn eine möglichst reine Positiv-Klasse gewünscht ist. Dabei werden jedoch eventuell einige positive Objekte falsch klassifiziert, die Trefferquote sinkt daher.

Sind die positiven Objekte jedoch rar in der Menge, dann ist eine hohe Trefferquote geeignet, um möglichst viele davon einzufangen. Dadurch sinkt jedoch die Präzision, da auch manche negative Objekte positiv klassifiziert werden.

Daher ist die geeignete Wahl der Metrik vom Problem abhängig.

4. Signal-Hintergrund-Trennung

4.1. Ausgangslage

Eine wichtige Aufgabe bei Experimenten in der Physik ist die Trennung zwischen Signalen für welche ein Interesse für die weitere Untersuchung besteht von anderen Signalen und insbesondere dem Hintergrundrauschen. Diese Entscheidung soll darüber hinaus auch schnell erfolgen, denn oftmals gibt es in kurzer Zeit eine große Menge von zu untersuchenden Objekten.

In Abbildung 4.1 ist ein einfacher Teilchendetektor vom CERN für Ausbildungszwecke dargestellt (siehe auch <https://scoollab.web.cern.ch/diy-particle-detector>). Dieser Detektor wird verwendet um Signale aufzuspüren. Ein Beispiel einer aufgezeichneten Welle ist in Abbildung 4.2 dargestellt.

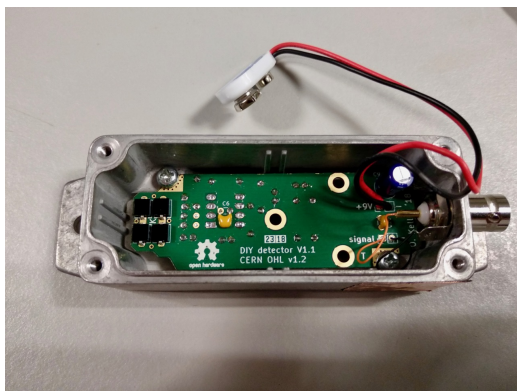


Abbildung 4.1.: CERN-DIY-Detektor
credit:https://github.com/ozel/DIY_particle_detector/

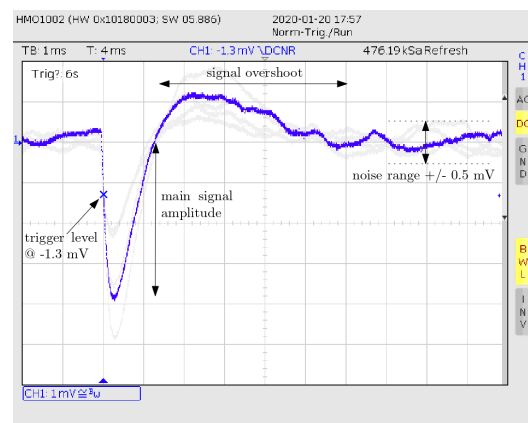


Abbildung 4.2.: Beispiel einer Welle
credit:https://github.com/ozel/DIY_particle_detector/

Die Ausgabe des Detektors wird an eine Soundkarte übertragen. Die Soundkarte erfasst die Welle mit einer Abtastrate von 96 kHz . Es werden dabei jeweils 100 Messpunkte erfasst, welche das Signal repräsentieren. Dies sind diskrete Messwerte (siehe Abbildung 4.3). Das auslösende Ereignis soll dabei in der Mitte des Abtastfensters sein. Ein Signal hat dabei eine hohen negativen Ausschlag, gefolgt von einem positiven Ausschlag, etwa halb so ausgeprägt, aber doppelt so lang, der sogenannte *overshoot*. Das ist der grobe Verlauf, wenn dieses Signal nicht durch Rauschen gestört wird, welches die Form verändert.

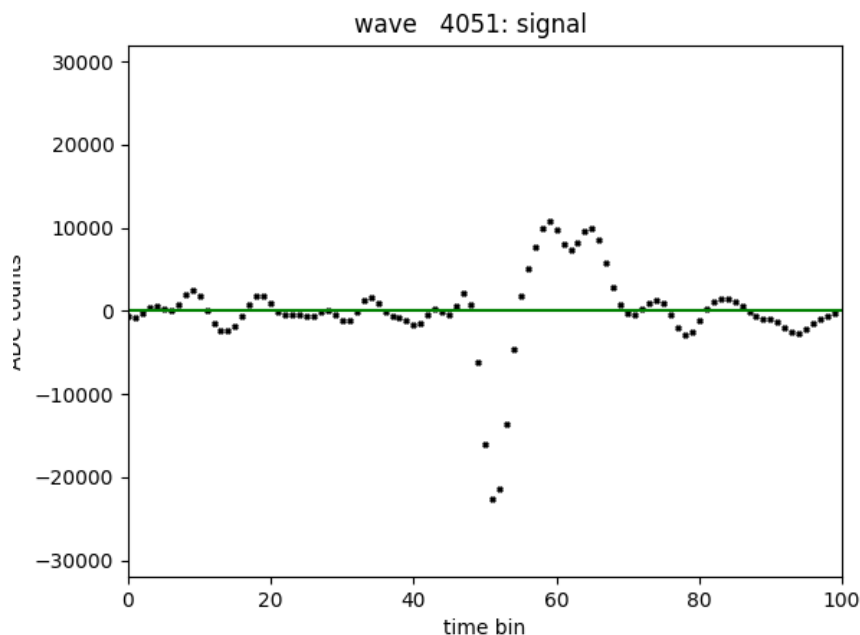


Abbildung 4.3.: Beispiel der Abtastpunkte eines Signals

Mit dem Detektor wurden zwei Messreihen durchgeführt, um ausreichend viele Messdaten zu erhalten.

- Bei der ersten Messreihe (Signale) wird eine aktive Probe neben den Detektor gestellt, so dass der Detektor permanent Signale aufnimmt. Damit sind die damit erfassten Messdaten Beispiele für Signale.
- Bei der zweiten Messreihe (Hintergrund) wird ohne ein aktives Präparat in der Nähe die Messungen durchgeführt. Die Messung wird ohne besondere Abschirmung durchgeführt. Daher können auch echte, natürliche Signale erfasst werden. Das bedeutet, dass in der Messreihe auch Signale gemessen werden.

Die Messwerte sind ohne größere Abschirmung aufgenommen worden. Daher sind bei den Hintergrundwellen auch mögliche natürliche Signale aufgenommen worden. Diese beiden Messreihen werden dazu verwendet, mit Hilfe von Methoden des Maschinellen Lernens, die Trennung von Signal und Hintergrund zu ermöglichen. In der Abbildung 4.4 sind einige wenige Aufnahmen von aufgenommenen Wellen gezeigt. Bei einigen mag es einfach sein zu entscheiden, ob es ein Signal oder ein Hintergrund ist. Dies kann jedoch nicht immer sicher erfolgen.

4.2. Rohdaten aufbereiten und bereinigen

In einem ersten Schritt (siehe Jupyter-Notebook `sb01-ym12csv.ipynb`) werden die von der Soundkarte aufgezeichneten Daten, die in einer yaml-Datei ([27]) vorliegen aufbereitet

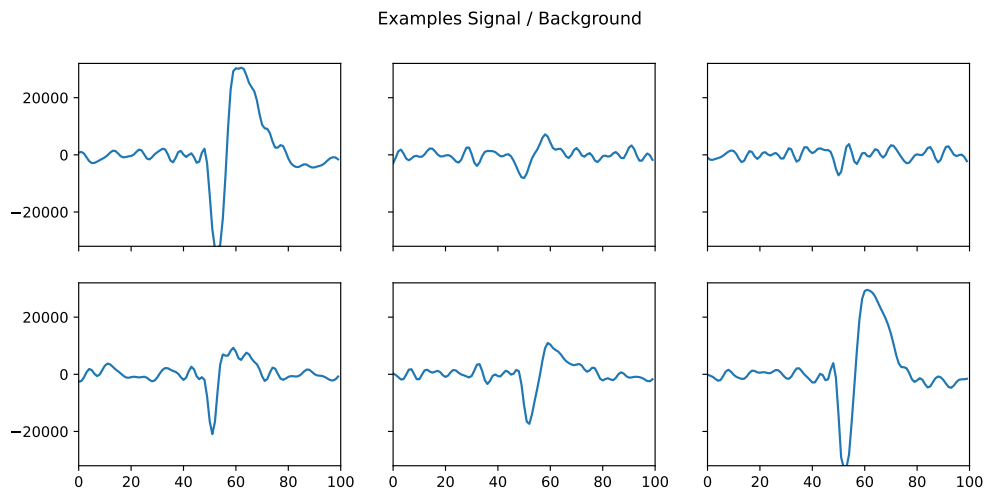


Abbildung 4.4.: Beispiele von Signal / Hintergrund-Daten, die mit dem CERN DIY Particle Detector erfasst und mit einer Soundkarte aufgezeichnet wurden. Es ist keine Kennzeichnung enthalten, ob es Signal oder Hintergrund ist.

und in eine CSV-Datei geschrieben. Eine Welle besteht aus 100 Messwerten der Pulshöhen. Es sind zwei Dateien, welche die Daten als yaml-Datei beinhalten. Eine Datei für die Messreihe mit den Signalen, eine Datei für die Messreihe vom Hintergrund. Ausgegeben werden die Daten in einer CSV-Datei, wobei jede Instanz neben den 100 Messwerten auch das Kennzeichen beinhaltet, ob die Welle eine Signal-Welle oder eine Hintergrund-Welle ist. Damit kann dann ein überwachtes Maschinelles Lernen durchgeführt werden.

Die Daten werden bei dieser Transformation bereinigt:

- Wellen, deren maximaler (negativer) Impuls eher am Rande als in der Mitte des betrachteten Zeitfensters ist, werden nicht weiter behandelt und somit aussortiert. Sie stehen somit nicht für die Trainingsmenge zur Verfügung.
- Bei den Wellen, welche in der Datei mit dem Hintergrund enthalten sind, deren maximaler (negativer) Impuls den Wert von 8000 überschreiten werden aussortiert. Diese Wellen sind potentielle Signale, also nicht sicher Hintergrund. Dies kann erfolgen, da der Detektor nicht abgeschirmt ist.

Bei diesem Experiment sind folgende Anzahlen von Signal- und Hintergrundwellen eingelesen und ausgegeben worden. Dabei ist bei $s+b$ *gesamt* die Summe von Signal- und Hintergrundwellen angegeben.

	eingelesen	entfernt	ausgegeben
signal	15250	584	14666
background	10817	1729	9088
s + b gesamt	26067	2313	23754

In der Ausgabedatei stehen die Daten von 23754 Instanzen, davon 14666 Signale und 9088 Hintergrund. Jede Instanz besteht aus 100 Messwerten und dem Kennzeichen, ob diese Instanz ein Signal oder Hintergrund ist. Das ist der Ausgangspunkt für eine Bilderkennung mit einem überwachten Lernen.

Im Anhang (siehe Kapitel B) sind Bilder von Wellen mit Signalen und Hintergrund dargestellt. In der Abbildung B.1 sind Beispiele von Wellen dargestellt, wobei das Kennzeichen Signal oder Hintergrund nicht angegeben ist. Manchmal ist es einfach festzustellen was ein Signal ist, was nur Hintergrundrauschen ist. Bei einigen ist dies jedoch schwer festzustellen. In der Abbildung B.2 ist jeweils mit angegeben, ob es ein Signal oder ein Hintergrundrauschen ist.

4.3. Daten anreichern

Im vorherigen Abschnitt wurde beschrieben, wie die Rohdaten aufbereitet wurden. Diese Rohdaten (die 100 Pixel für den Verlauf der Welle plus das Kennzeichen, ob es Signal oder Hintergrundrauschen ist) werden nun aufbereitet. Ziel ist es Kenndaten und Merkmale für die Anwendung von Verfahren des Maschinellen Lernens hinzuzufügen.

Es werden zwei verschiedene grundlegende Daten ergänzt:

1. Aus dem Verlauf der Welle wird ein Teil herausgeschnitten. Der Ausschnitt beginnt vier Pixel vor der Position des maximalen (negativen) Peaks der Welle. Es werden ab dieser Position 25 Pixel herangezogen. Dieser Ausschnitt entspricht der Kontur als dem Verlauf des Signals, wenn es ein Signal ist. Diese Daten werden für die grafische Ausgabe normiert, so dass der maximale (negative) Peak bei 32000 liegt, Für die spätere Bearbeitung im ML wird der maximale Peak auf 1 normiert.

Die Normierung erfolgt, damit bei diesem Auszug nur der Verlauf der Welle eine Rolle spielt. Die Höhe des Peaks hat dabei keine Bedeutung. Diese Höhe wurde schon bei der Filterung der Daten eingesetzt. Somit hat dieses Merkmal Vorwissen bezüglich der Trennung von Signal und Hintergrund. Damit ist es als Trainingsdatum nicht geeignet.

2. Es werden einige Kennzahlen, wie Peakhöhe oder Breite, eines Ausschlags ermittelt und als neue Merkmale in die Daten eingefügt.

Die gesamten Daten werden dann in einer CSV-Datei gespeichert, welche für die weitere Verarbeitung und damit die Anwendung verschiedener Verfahren des ML verwendet werden. Dabei werden nicht unbedingt alle Daten verwendet.

Die Implementierung ist im Jupyter-Notebook `sb02-feateng.ipynb` zu sehen. Im nachfolgenden sind die Merkmale erläutert, die in der Ausgabe vorhanden sind:

- `P0 ... P99`: Die 100 Pixel des originalen Verlauf der Welle. Diese Daten sind in der Ausgabe vorhanden, da sie für die grafische Darstellung verwendet werden. Sie werden nicht für die Anwendung in einem ML-Verfahren verwendet.

- INDEX Laufende Nummer der Welle im Datensatz, nur ein informelles Merkmal.
- POS_FROM, POS_TILL: Erste Position und letzte Position (plus 1) des 25-Pixel-Auszugs der Welle. Diese Daten werden für die grafische Ausgabe der Welle benötigt.
- POS_PEAK: Position des maximalen (negativen) Peaks.
- C0 ... C24: Die 25 Pixel für den Verlauf / die Kontur der Welle. Die Daten sind für die grafische Ausgabe normiert, so dass der höchste Wert bei 32000 ist. (Damit geht der normierte Auszug der Welle bis zum unteren Rand des Zeichenbereiches.)
- PEAK1 Höhe des maximalen (negativen) Peaks der Welle. Es wird ein positiver Wert gespeichert.
- PEAK2 Höhe des maximalen (positiven) Peaks nach dem maximalen (negativen) Peak.
- P2P_DIFF Vertikaler Abstand zwischen den beiden Peaks: $P2P_DIFF = PEAK1 + PEAK2$. Da für die Aufbereitung der Wellen diejenigen Wellen im Hintergrund aussortiert wurden, deren maximaler (negativer) Peak größer als 8000 ist, ist auch dieser Wert für das Lernen nicht geeignet. Eine zu hohe Differenz bedeutet, dass es sich um ein Signal handelt.
- P2P_RATIO Verhältnis der Peak-Höhen: $P2P_RATIO = PEAK1/PEAK2$.
- P2P_DIST Horizontaler Abstand (Distanz) zwischen den beiden Peaks.
- WIDTH1 Halbwertsbreite (FWHM, Full Width at Half Maximum) von PEAK1.
- WIDTH2 Halbwertsbreite von PEAK2.
- LABEL Kennzeichen, ob es sich um ein Signal (1) oder Hintergrund (-1) handelt.

4.4. Bearbeitung mit der Kontur

Kann die Trennung zwischen Signal und Hintergrund anhand dem Verlauf der Welle um den maximalen (negativen) Peak herum erfolgen? In Abbildung 4.5 sind einige Beispiele von Wellen aufgezeigt. Aus dem gesamten Verlauf mit 100 Pixeln werden 25 Pixel herangezogen. Der Verlauf beginnt vier Pixel vor dem maximalen (negativen) Peak.

Bei den Beispielen ist angegeben, ob es sich um ein Signal oder Hintergrund handelt. Bei einigen Wellen scheint es klar zu sein, ob es Signal oder Hintergrund ist. Bei anderen kann es jedoch schwierig sein. Wenn die Normierung auf die maximalen Impulshöhe von 32000 nicht durchgeführt wird, dann ist es einfach zu entscheiden. Alles mit einer Impulshöhe über 8000 ist ein Signal. Durch die Normierung entfällt diese Möglichkeit. In der Abbildung 4.6 sind auch wieder Beispiel von Konturen aufgezeigt. Dieses Mal fehlt jedoch die Originalwelle und es wird auch nicht angegeben, ob es ein Signal oder Hintergrund ist.

4. Signal-Hintergrund-Trennung

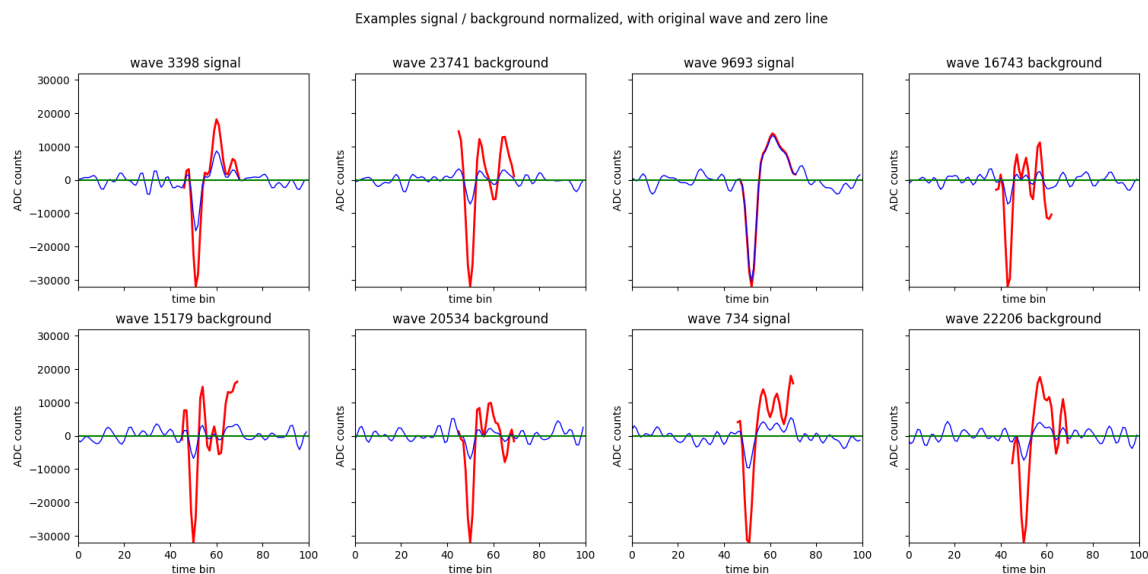


Abbildung 4.5.: Einige Beispiele von Signal- und Hintergrunddaten. Es werden die Originalwelle (schwarz), die Nulllinie (grün) und der Verlauf / die Kontur um den maximalen (negativen) Peak (rot) geplottet. Die Kontur ist dabei (für die grafische Darstellung) so normiert, dass der maximale (negative) Wert bei 32000 liegt. Bei den einzelnen Beispielen wird auch angegeben, ob es ein Signal oder Hintergrund ist.

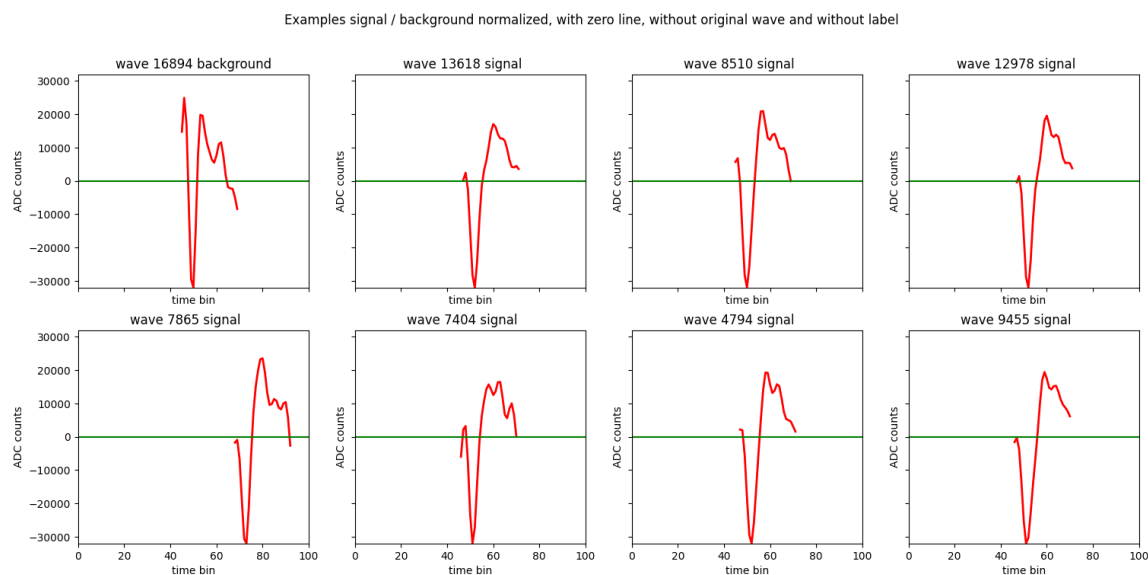


Abbildung 4.6.: Einige Beispiele von Signal- und Hintergrunddaten. Es werden nur die Nulllinie (grün) und der normierte Verlauf / die Kontur um den maximalen (negativen) Peak (rot) geplottet. Hier ist die Angabe, ob es ein Signal oder Hintergrund ist nicht gegeben.

Kann mit Hilfe einfacher ML-Algorithmen die Entscheidung, ob ein Signal oder Hintergrund vorliegt, getroffen werden, also allein auf Basis des Verlaufs, der Kontur? Die entsprechenden Implementierungen sind im Jupyter Notebook `sb03-kontur.ipynb` zu finden.

Für die Bearbeitung werden die Konturdaten so normiert, dass der Wert für den maximalen (negativen) Impuls bei -1 ist. Dies ist für die Verarbeitung in den Algorithmen vorteilhafter. Grafisch liegt der Wert weiterhin bei -32000.

4.4.1. ML-Algorithmus: Logistische Regression

Zuerst wird eine **Logistische Regression** (*Logistic Regression*) durchgeführt (siehe [28], [29]). Dabei wird zuerst die ROC-Kurve erstellt. Mit Hilfe von `sklearn` sehen die wesentlichen Befehle für die Logistische Regression folgendermaßen aus:

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression (max_iter=1000, solver='lbfgs')
log_reg.fit (X_train_scaled, y_train)
y_pred_prob = log_reg.predict_proba (X_test_scaled)[: , 1]
```

Die Kurve (siehe Abbildung 4.7) ist fast eine ideale ROC-Kurve und deutet an, dass die Trennung zwischen Signal und Hintergrund mittels einer Logistischen Regression gut erfolgen kann. Der AUC-Wert (Fläche unter der Kurve) beträgt dabei 0.9923.

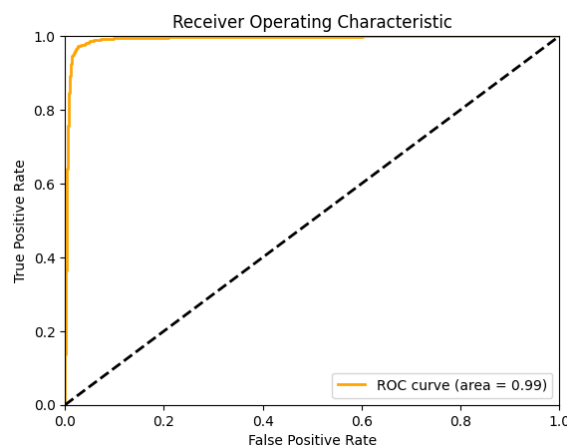


Abbildung 4.7.: ROC-Kurve für die Logistische Regression.

Anschließend wird die Logistische Regression durchgeführt. Von den 5939 Testdaten werden 174 falsch zugeordnet ($FP = 90$, $FN = 84$). Das entspricht einer Genauigkeit von 97.1 %. Die Reinheit beträgt 97.6 %, die Trefferquote 97.7 %.

4.4.2. ML-Algorithmus: Perzeptron

Dann wird ein erstes einfaches Neuronales Netz angewendet, ein **Perzeptron** (*Perceptron*) (siehe [28], [29]). Hier wird ein Eingangslayer direkt mit den Ausgangsknoten verknüpft. Hier hat der Eingangslayer 25 Einträge, für jeden Pixelwert der Kontur eine Eingabe.

Der wesentliche Aufruf mit Hilfe von sklearn ist

```
from sklearn.linear_model import Perceptron

perceptron = Perceptron ()
perceptron.fit (X_train_scaled, y_train)
y_pred_perc = perceptron.predict (X_test_scaled)
```

Es werden die Standardeinstellung von sklearn genommen.

Von den 5939 Testdaten werden 184 falsch zugeordnet ($FP = 110$, $FN = 74$). Das entspricht einer Genauigkeit von 96.9 %. Die Reinheit beträgt 97.0 %, die Trefferquote 98.0 %.

4.4.3. ML-Algorithmus: Neuronales Netz

Als nächstes wird ein (einfaches) **Neuronales Netz** (*Neuronal Network*) verwendet (siehe [28], [29]). Dabei werden für die verborgenen Layer unterschiedliche Designs getestet - sowohl mit einem verborgenen Layer, als auch mit mehreren verborgenen Layern. Es wird nicht nach dem (einem) optimalen Design der Layer gesucht. Für das Neuronale Netz wird als Eingangslayer 25 Knoten gewählt. Jeder Wert der 25 Pixel ist eine Eingabe. Beim Ausgangslayer werden zwei Knoten gewählt: Signal oder Hintergrund. Zuerst wird mit einem verborgenen Layer mit 15 Knoten getestet. Die Skalierung der Daten ist wiederum so gewählt, dass der maximale (negative) Impuls den Wert -1 hat. Der wesentliche Aufruf mit Hilfe von sklearn ist

```
from sklearn.neural_network import MLPClassifier

layer_size = (15,)
mlp = MLPClassifier (hidden_layer_sizes=layer_size)
mlp.fit (X_train, y_train)
y_pred = mlp.predict (X_test)
```

Es werden die Standardeinstellung von sklearn genommen.

Von den 5939 Testdaten werden 130 falsch zugeordnet ($FP = 58$, $FN = 72$). Das entspricht einer Genauigkeit von 97.8 %. Die Reinheit beträgt 98.4 %, die Trefferquote 98.0 %.

Für dieses Layerdesign wird das trainierte Modell auch auf die Trainingsdaten angewendet. Die Ergebnisse in kompakter Darstellung sind:

	TP	FN	FP	TN	acc	pre	rec	FPR	TPR	F1
test	3607	72	58	2202	0.9781	0.9842	0.9804	0.0257	0.9804	0.9823
train	10810	177	158	6670	0.9812	0.9856	0.9839	0.0231	0.9839	0.9847

Für dieses Design wird auch die Lernkurve ermittelt (siehe Abbildung 4.8).

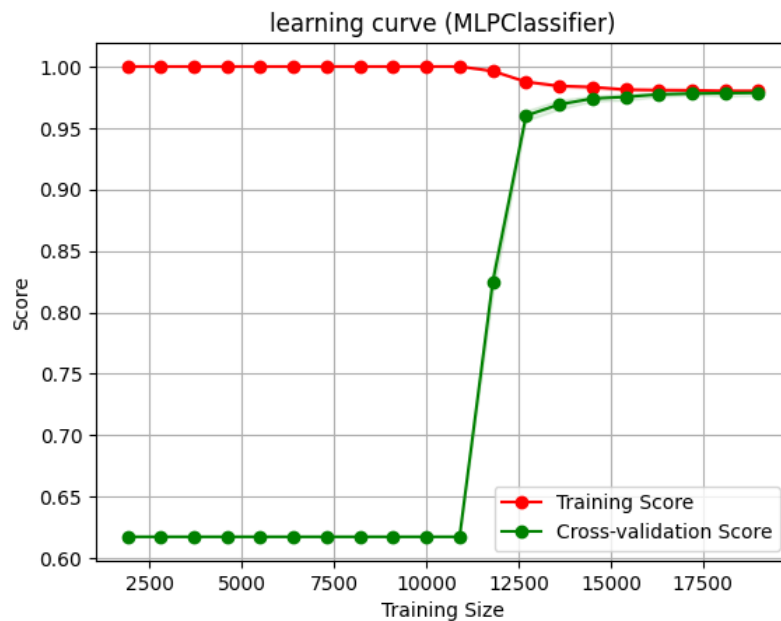


Abbildung 4.8.: Lernkurve für ein Neuronales Netz (MLPClassifier) mit einem verborgenen Layer mit 15 Knoten.

Bei den obigen Untersuchungen ist die Aufteilung zwischen Trainingsdaten und Testdaten fixiert. Mit Hilfe der Cross-Validation (CV) wird nun für drei verschiedene Layerdesigns die Genauigkeit (accuracy) bestimmt (siehe Tabelle 4.1).

Layerdesign	Genauigkeit
(15,)	0.9791 ± 0.0019
(5,)	0.9770 ± 0.0026
(15,10,)	0.9791 ± 0.0027

Tabelle 4.1.: Berechnung der Genauigkeit (accuracy) für verschiedene Layerdesigns. Für das CV wird die Trainingsmenge in zehn Teile aufgeteilt.

Die Genauigkeit unterscheidet sich nur wenig, trotz unterschiedlicher Designs der Layer.

Nun werden verschiedene Designs der Layer durchgearbeitet und die wichtigsten Kennzahlen ermittelt. Die Ergebnisse sind in der Tabelle 4.2 zu sehen.

Die Werte für die Genauigkeit liegen im Bereich $0.9715 \dots 0.9808$, für die Präzision im Bereich $0.9764 \dots 0.9858$ und für die Trefferquote im Bereich $0.9758 \dots 0.9848$.

Layerdesign	TP	FN	FP	TN	acc	pre	rec	FPR	TPR	F1
(5,)	3607	72	67	2193	0.9766	0.9818	0.9804	0.0296	0.9804	0.9811
(10,)	3607	72	82	2178	0.9741	0.9778	0.9804	0.0363	0.9804	0.9791
(15,)	3607	72	58	2202	0.9781	0.9842	0.9804	0.0257	0.9804	0.9823
(20,)	3613	66	57	2203	0.9793	0.9845	0.9821	0.0252	0.9821	0.9833
(25,)	3622	57	57	2203	0.9808	0.9845	0.9845	0.0252	0.9845	0.9845
(30,)	3618	61	55	2205	0.9805	0.9850	0.9834	0.0243	0.9834	0.9842
(40,)	3622	57	57	2203	0.9808	0.9845	0.9845	0.0252	0.9845	0.9845
(50,)	3620	59	58	2202	0.9803	0.9842	0.9840	0.0257	0.9840	0.9841
(4,)	3597	82	87	2173	0.9715	0.9764	0.9777	0.0385	0.9777	0.9770
(3,)	3594	85	82	2178	0.9719	0.9777	0.9769	0.0363	0.9769	0.9773
(2,)	3590	89	79	2181	0.9717	0.9785	0.9758	0.0350	0.9758	0.9771
(5, 5)	3596	83	67	2193	0.9747	0.9817	0.9774	0.0296	0.9774	0.9796
(10, 5)	3623	56	68	2192	0.9791	0.9816	0.9848	0.0301	0.9848	0.9832
(15, 5)	3614	65	61	2199	0.9788	0.9834	0.9823	0.0270	0.9823	0.9829
(15, 10)	3604	75	52	2208	0.9786	0.9858	0.9796	0.0230	0.9796	0.9827
(15, 15)	3622	57	58	2202	0.9806	0.9842	0.9845	0.0257	0.9845	0.9844
(15, 10, 5)	3613	66	62	2198	0.9784	0.9831	0.9821	0.0274	0.9821	0.9826

Tabelle 4.2.: Wichtige Kennzahlen bei der Bearbeitung mit unterschiedlichen Designs der Layer bei einem einfachen Neuronalen Netz (MLPClassifier).

4.4.4. Kurzes Resümee bezüglich der Verwendung der Kontur

Wird die Kontur der Welle um den maximalen (negativen) Peak zur Trennung von Signal und Hintergrund verwendet, dann sind die Ergebnisse sehr gut. Die Genauigkeit, die Präzision und die Trefferquote sind im Bereich über 97 %. Bei den verschiedenen Layerdesigns für die Neuronalen Netze ist der α -Fehler (FPR) bei 2 % bis 4 %, der β -Fehler (1 - TPR) bei etwa 2 %. (siehe Tabelle 4.2)

Einige Wellen sind bei allen drei Methoden (Logistische Regression, Perzeptoren, Neuronales Netz (MLPClassifier) mit dem Layerdesign (15,)) falsch zugeordnet worden. Es sind insgesamt 91 Wellen. Einige davon sind in Abbildung 4.9 zu sehen. Das angegebenen Label ist die korrekte Klasse für die Welle.

4.5. Bearbeitung mit Kennwerten der Welle

Neben der Kontur werden auch einigen Kenndaten der Welle erfasst, wie dies in Abschnitt 4.3 dargelegt ist. Dies sind die Kenndaten PEAK1, PEAK2, P2P_DIFF, P2P_RATIO, P2P_DIST, WIDTH1 und WIDTH2. Kann mit Hilfe dieser Merkmale auch eine Separation der Daten erfolgen? Die entsprechenden Implementierungen sind im Jupyter Notebook `sb04-features.ipynb` zu finden.

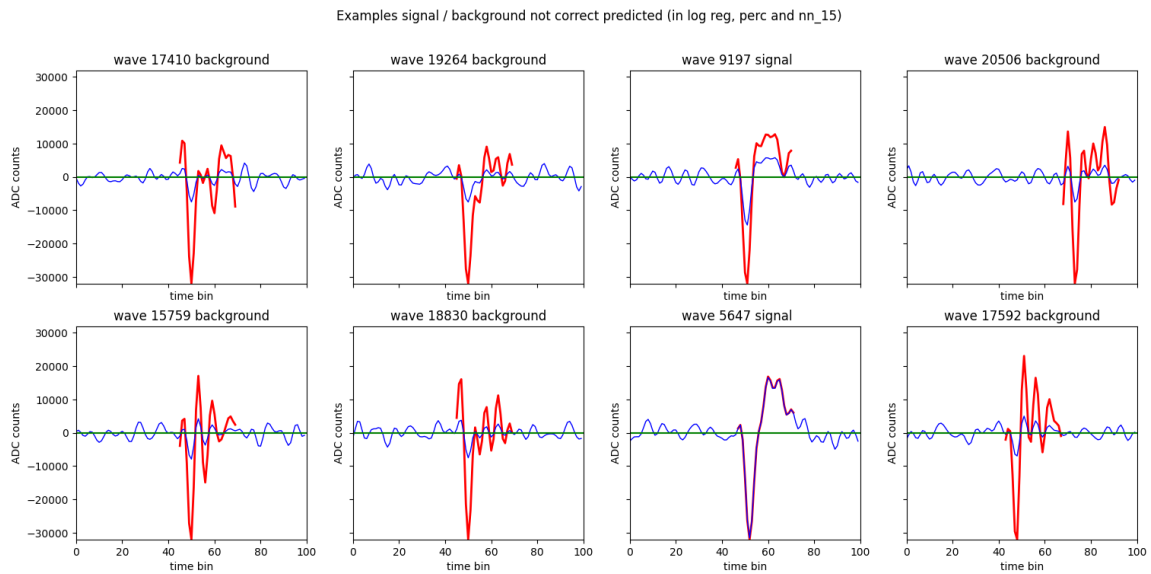


Abbildung 4.9.: Beispiele von Wellen, die bei allen drei Verfahren (Logistische Regression, Perzeptoren, Neuronales Netz (MLPClassifier) mit dem Layerdesign (15,)) jeweils falsch zugeordnet wurden. Das Label ist jeweils die korrekte Zuordnung.

4.5.1. Betrachtung der Kennwerte

In den beiden Merkmalen PEAK1 und P2P_DIFF ist die maximale (negative) Pulshöhe verarbeitet. Dieses Merkmal wird bei den Daten des Hintergrunds genutzt, um potenzielle Signale herauszufiltern. Daher ist bei den Hintergrunddaten diese Impulshöhe maximal 8000.

In der Abbildung 4.10 ist das Histogramm der Impulshöhe PEAK1, getrennt nach Signal (1) und Hintergrund (-1) zu sehen. Es gibt eine ziemlich klare Trennung. Ist die Impulshöhe größer 8000, dann ist es ein Signal, aber dementsprechende Hintergrundwellen wurden ausgefiltert.

In der Abbildung 4.11 ist das Histogramm der Differenz zwischen maximalen (negativen) und maximalen positivem Impuls, also P2P_DIFF, dargestellt. Hier gibt es einen überschneidenden Bereich. Aber auch hier hat das Kriterium der maximalen Pulshöhe (PEAK1) zum Herausfiltern von Hintergrundwellen wieder einen starken Einfluss.

Daher werden diese beiden Merkmale für die Trennung von Signal und Hintergrund nicht herangezogen

Für die Merkmale P2P_RATIO, P2P_DIST, WIDTH1 und WIDTH2 wird ein Pairplot, also ein paarweiser Plot erstellt, siehe Abbildung 4.12.

4. Signal-Hintergrund-Trennung

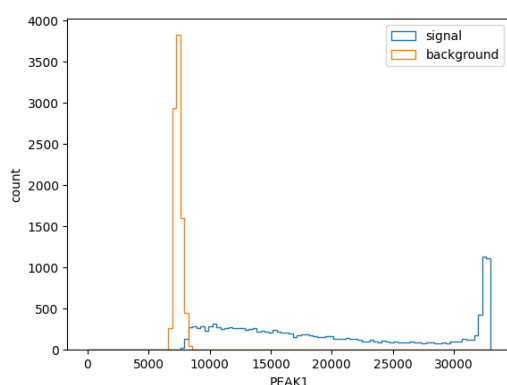


Abbildung 4.10.: Histogramm der maximalen (negativen) Impulshöhe PEAK1, getrennt nach Signal und Hintergrund

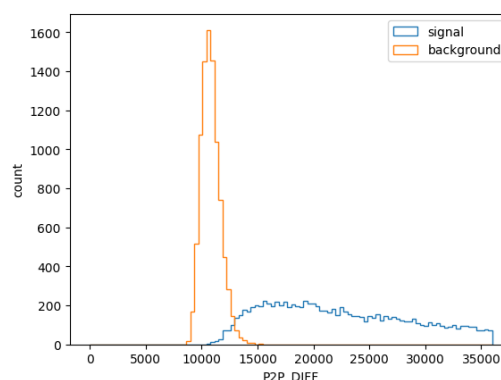


Abbildung 4.11.: Histogramm der maximalen (negativen) Impulshöhe P2P_DIFF, getrennt nach Signal und Hintergrund

4.5.2. Klassische ML-Algorithmen

Keines dieser Merkmale und auch keine Kombination davon ermöglicht eine klare saubere Trennung von Signal und Hintergrund. Trotzdem wird mit zwei einfachen klassischen ML-Algorithmen (Entscheidungsbaum (DT) und k-Nächste Nachbarn (kNN)) untersucht, welche Möglichkeiten anhand dieser vier Merkmale es gibt.

Beim **Entscheidungsbaum** gibt es bei den Testdaten eine Genauigkeit von 89.8 %, eine Präzision von 91.7 % und eine Trefferquote von 91.8 %. Es werden 605 Wellen falsch klassifiziert (FP = 304, FN = 301). Damit ist das Ergebnis gut, aber bei weitem nicht so gut wie das Ergebnis bei der Bearbeitung der Kontur.

Beim **k-Nächster Nachbar** werden mehrere Nachbarschaftsgrößen getestet. Die Ergebnisse sind in Tabelle 4.3 zu sehen.

Anzahl Nachbarn	TP	FN	FP	TN	acc	pre	rec	FPR	TPR	F1
3	3487	192	269	1991	0.9224	0.9284	0.9478	0.1190	0.9478	0.9380
5	3510	169	262	1998	0.9274	0.9305	0.9541	0.1159	0.9541	0.9422
7	3508	171	268	1992	0.9261	0.9290	0.9535	0.1186	0.9535	0.9411
9	3515	164	265	1995	0.9278	0.9299	0.9554	0.1173	0.9554	0.9425
11	3515	164	259	2001	0.9288	0.9314	0.9554	0.1146	0.9554	0.9432
13	3517	162	265	1995	0.9281	0.9299	0.9560	0.1173	0.9560	0.9428
15	3515	164	261	1999	0.9284	0.9309	0.9554	0.1155	0.9554	0.9430
17	3519	160	260	2000	0.9293	0.9312	0.9565	0.1150	0.9565	0.9437

Tabelle 4.3.: Wichtige Kennzahlen bei der Bearbeitung mit einer unterschiedlichen Anzahl von Nachbarn bei einem klassischen ML-Algorithmus k-Nächster Nachbar.

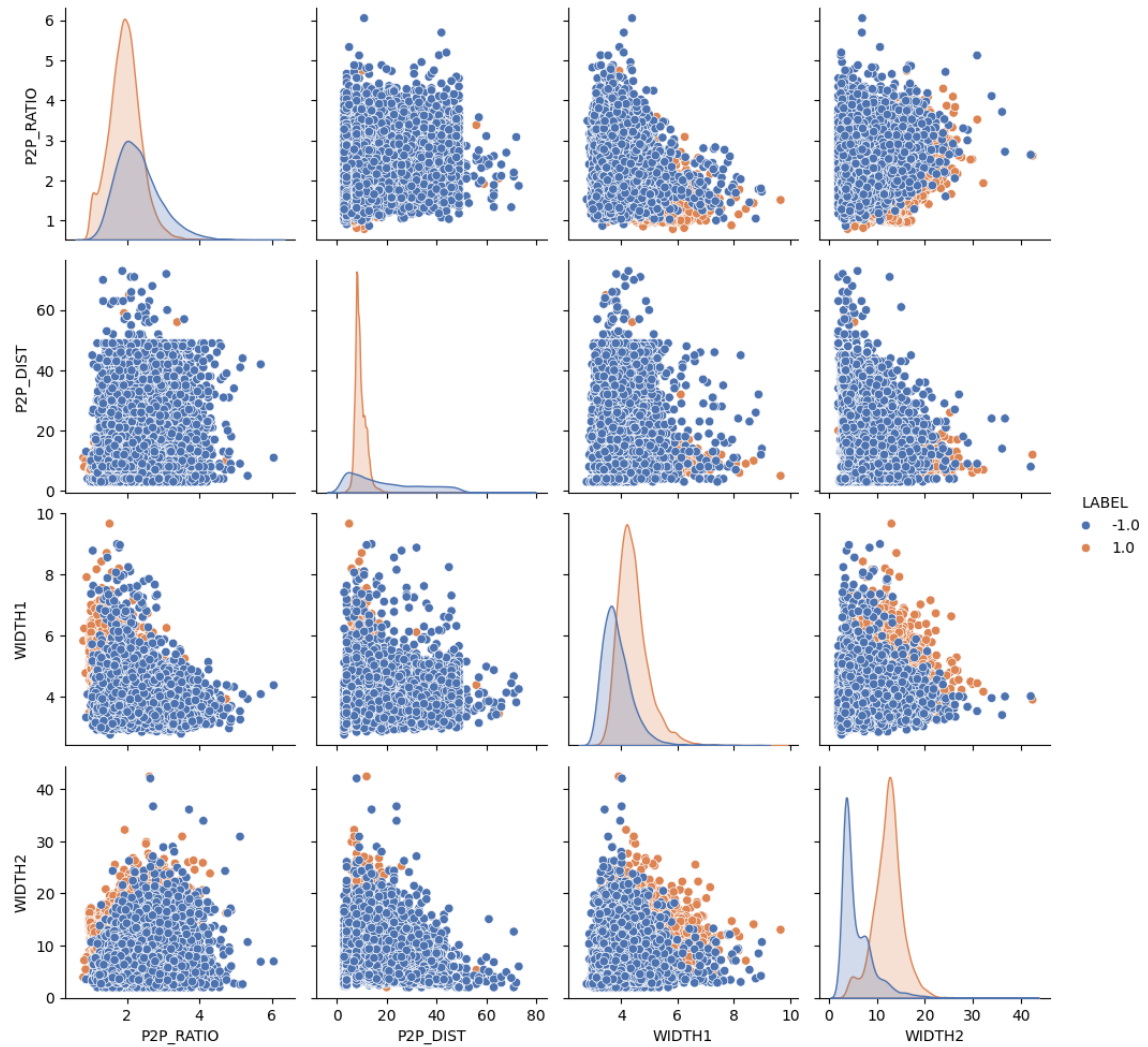


Abbildung 4.12.: Pairplot für die Merkmale P2P_RATIO, P2P_DIST, WIDTH1 und WIDTH2

Beim kNN ist das Ergebnis nicht viel besser als beim Entscheidungsbaum.

4.5.3. Logistische Regression, Perceptron und Neuronales Netz

Wie schon bei der Bearbeitung der Kontur werden die ML-Algorithmen **Logistische Regression**, **Perzeptron** und **Neuronales Netz** angewendet.

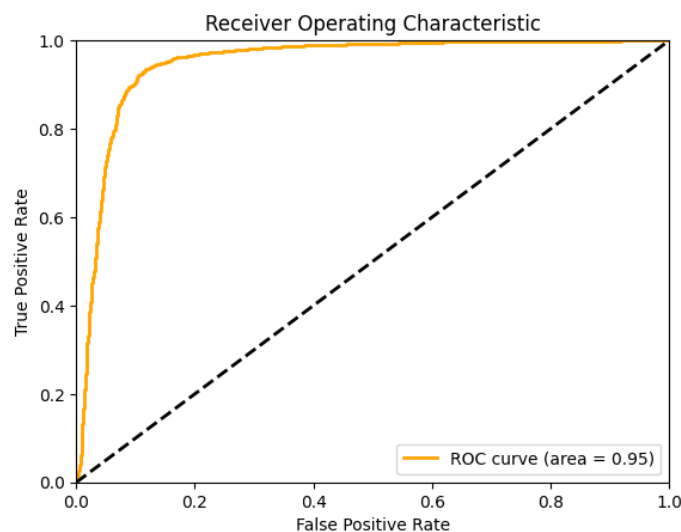


Abbildung 4.13.: ROC-Kurve bei der Anwendung einer Logistischen Regression auf die vier Merkmale P2P_RATIO, P2P_DIST, WIDTH1 und WIDTH2

In der Abbildung 4.13 ist die ROC-Kurve dargestellt, die mit Hilfe der **Logistischen Regression** bei der Anwendung auf die vier Merkmale P2P_RATIO, P2P_DIST, WIDTH1 und WIDTH2 erzeugt wurde. Das Ergebnis ist nicht so gut wie bei der Anwendung auf die Kontur. Bei den Testdaten ergab die Logistische Regression eine Genauigkeit von 91.3 %, eine Präzision von 91.5 % und eine Trefferquote von 94.8 %. Es werden 516 Wellen falsch klassifiziert (FP = 325, FN = 191).

Bei den Testdaten ergab der ML-Algorithmus **Perzeptron** eine Genauigkeit von 84.2 %, eine Präzision von 83.1 % und eine Trefferquote von 93.4 %. Es werden 940 Wellen falsch klassifiziert (FP = 697, FN = 243).

Bei den Testdaten ergab der ML-Algorithmus **Neuronales Netz** mit einem versteckten Layer mit 4 Knoten eine Genauigkeit von 92.9 %, eine Präzision von 93.1 % und eine Trefferquote von 95.7 %. Es werden 419 Wellen falsch klassifiziert (FP = 262, FN = 157).

Die vier Merkmale bringen keine so guten Ergebnisse wie die Verwendung der Kontur.

4.6. Kontur und die vier Merkmale

Das Ergebnis bei der Verwendung der vier Kenndaten im letzten Abschnitt war schlechter als die Ergebnisse bei der Bearbeitung mit der Kontur. Kann das Ergebnis bei der Verwendung der Kontur verbessert werden, wenn noch die vier Merkmale mit hinzugefügt werden? Dies ist im Jupyter Notebook `sb05-konturplus.ipynb` untersucht.

Für die Anwendung der ML Algorithmen werden die 25 Pixel der Kontur wieder dahingehend normiert, dass der maximale (negative) Impuls den Wert -1 hat. Es werden die Logistische Regression, das Perzeptron und das Neuronale Netz angewendet.

4.6.1. ML-Algorithmus: Logistische Regression

Zuerst wird wieder die ROC-Kurve erstellt. Der Verlauf ähnelt sehr der Kurve bei der Bearbeitung Kontur ohne die zusätzlichen vier Merkmale (siehe Abbildung 4.7). Es ist also ebenso fast eine ideale ROC-Kurve und deutet an, dass die Trennung zwischen Signal und Hintergrund mittels einer Logistischen Regression gut erfolgen kann. Der AUC-Wert (Fläche unter der Kurve) beträgt dabei sogar 0.9948. Er ist somit sogar noch etwas besser.

Anschließend wird die Logistische Regression durchgeführt. Von den 5939 Testdaten werden 109 falsch zugeordnet ($FP = 56$, $FN = 53$). Das entspricht einer Genauigkeit von 98.2 %. Die Reinheit beträgt 98.5 %, die Trefferquote 98.6 %.

4.6.2. ML-Algorithmus: Perzeptron

Dann wird ein erstes einfaches Neuronales Netz angewendet, ein **Perzeptron**, bei dem ein Eingangslayer direkt mit den Ausgangsknoten verknüpft ist. Hier hat der Eingangslayer 29 Einträge, für jeden Pixelwert der Kontur eine Eingabe und jeweils einen für die vier zusätzlichen Merkmale (P2P_RATIO, P2P_DIST, WIDTH1 und WIDTH2).

Von den 5939 Testdaten werden 146 falsch zugeordnet ($FP = 70$, $FN = 76$). Das entspricht einer Genauigkeit von 97.5 %. Die Reinheit beträgt 98.1 %, die Trefferquote 97.9 %.

4.6.3. ML-Algorithmus: Neuronales Netz

Als nächstes wird ein (einfaches) **Neuronales Netz** verwendet. Dabei werden für die verborgenen Layer unterschiedliche Designs getestet - sowohl mit einem verborgenen Layer, als auch mit mehreren verborgenen Layern. Es wird nicht nach dem (einem) optimalen Design der Layer gesucht. Es werden die selben Layerdesign wie in Abschnitt 4.4.3 verwendet.

Für das Neuronale Netz hat das Eingangslayer 29 Knoten. Jeder Wert der 25 Pixel ist eine Eingabe, sowie die vier Merkmale. Beim Ausgangslayer werden zwei Knoten gewählt: Signal oder Hintergrund.

Als ersten wird mit einem verborgenen Layer mit 15 Knoten getestet. Die Skalierung der Daten ist wiederum so gewählt, dass der maximale (negative) Impuls den Wert -1 hat.

Von den 5939 Testdaten werden 113 falsch zugeordnet ($FP = 50$, $FN = 63$). Das entspricht einer Genauigkeit von 98.1 %. Die Reinheit beträgt 98.6 %, die Trefferquote 98.3 %.

Nun werden verschiedene Designs der Layer durchgearbeitet und die wichtigsten Kennzahlen ermittelt. Die Ergebnisse sind in der Tabelle 4.4 zu sehen.

Layerdesign	TP	FN	FP	TN	acc	pre	rec	FPR	TPR	F1
(5,)	3626	53	53	2207	0.9822	0.9856	0.9856	0.0235	0.9856	0.9856
(10,)	3621	58	55	2205	0.9810	0.9850	0.9842	0.0243	0.9842	0.9846
(15,)	3616	63	50	2210	0.9810	0.9864	0.9829	0.0221	0.9829	0.9846
(20,)	3622	57	53	2207	0.9815	0.9856	0.9845	0.0235	0.9845	0.9850
(25,)	3630	49	53	2207	0.9828	0.9856	0.9867	0.0235	0.9867	0.9861
(30,)	3624	55	46	2214	0.9830	0.9875	0.9851	0.0204	0.9851	0.9863
(40,)	3621	58	50	2210	0.9818	0.9864	0.9842	0.0221	0.9842	0.9853
(50,)	3622	57	44	2216	0.9830	0.9880	0.9845	0.0195	0.9845	0.9862
(4,)	3615	64	45	2215	0.9816	0.9877	0.9826	0.0199	0.9826	0.9851
(3,)	3623	56	49	2211	0.9823	0.9867	0.9848	0.0217	0.9848	0.9857
(2,)	3621	58	58	2202	0.9805	0.9842	0.9842	0.0257	0.9842	0.9842
(5, 5)	3622	57	53	2207	0.9815	0.9856	0.9845	0.0235	0.9845	0.9850
(10, 5)	3622	57	52	2208	0.9816	0.9858	0.9845	0.0230	0.9845	0.9852
(15, 5)	3621	58	49	2211	0.9820	0.9866	0.9842	0.0217	0.9842	0.9854
(15, 10)	3624	55	51	2209	0.9822	0.9861	0.9851	0.0226	0.9851	0.9856
(15, 15)	3632	47	50	2210	0.9837	0.9864	0.9872	0.0221	0.9872	0.9868
(15, 10, 5)	3625	54	52	2208	0.9822	0.9859	0.9853	0.0230	0.9853	0.9856

Tabelle 4.4.: Wichtige Kennzahlen bei der Bearbeitung mit unterschiedlichen Designs der Layer bei einem einfachen Neuronalen Netz (MLPClassifier). Bearbeitet werden die Kontur (25 Pixel) und vier weitere Merkmale.

4.6.4. Kurzes Resümee bezüglich der Verwendung der Kontur mit den vier zusätzlichen Merkmalen

Wird die Kontur der Welle um den maximalen (negativen) Peak und zusätzlich vier Kennzahlen der Welle zur Trennung von Signal und Hintergrund verwendet, dann sind die Ergebnisse sogar noch etwas besser als wenn die vier zusätzlichen Merkmale nicht mit verwendet werden. Die Genauigkeit, die Präzision und die Trefferquote sind im Bereich über 97 %, teilweise sogar über 98 %. Bei den verschiedenen Layerdesigns für die Neuronalen Netze ist der α -Fehler (FPR) bei etwa 2 %, der β -Fehler ($1 - \text{TPR}$) knapp unter 2 %. (siehe Tabelle 4.4)

Einige Wellen sind bei allen drei Methoden (Logistische Regression, Perzeptoren, Neuronales Netz (MLPClassifier) mit dem Layerdesign (15,)) falsch zugeordnet worden. Es sind insgesamt 71 Wellen.

5. Schlussbemerkungen

5.1. Zusammenfassung und erster Ausblick

Diese Arbeit ist nur ein kleiner und (leider) unvollständiger Einstieg in das Themenfeld des Maschinellen Lernens. Dazu ist dieses Gebiet viel zu umfangreich.

Eine der Hauptaufgaben in den ersten Kapiteln ist es, Studierenden der Physik, die im Bachelorstudium Kenntnisse in Mathematik und in Statistik (beispielsweise in der Vorlesung „Computergestützte Datenauswertung“ (CgDA)) erworben haben, in das ML einzuführen. Ziel ist es dabei, die Verfahren an physikalischen Problemen anschaulich einzuführen.

Dazu benötigt es jedoch noch mehr Inhalt, um weitere klassische Verfahren einzuführen und insbesondere den Themenkomplex der Regression zu adressieren. Dies sprengt den Rahmen einer Bachelorarbeit, ist jedoch eine sinnvolle Fortsetzung, um Studierende auf die Verwendung in Bachelorarbeiten oder auf weitere Kurse (beispielsweise die Vorlesung „Moderne Datenanalyse“) vorzubereiten.

Eine weitere Hauptaufgabe in den ersten Kapiteln ist es, die Begriffswelt des Maschinellen Lernens einzuführen und die Begriffe mit der physikalischen Begriffswelt abzugleichen. Einige Begriffe, die im ML verwendet werden, haben in der Physik eine andere Bedeutung (beispielsweise der Begriff *Modell*). Andere Begriffe die im ML verwendet werden, sind in der physikalischen Begriffswelt schon länger bekannt, jedoch unter einem anderen Namen (beispielsweise Präzision *precision* gegenüber Reinheit (*purity*)). Es ist daher wichtig, eine Übersetzung der Begriffe zu ermöglichen. Damit können dann Physikerinnen und Physiker die Werkzeuge des ML einschätzen und anwenden. Auf der anderen Seite können dann Expertinnen und Experten des ML die Physik unterstützen. Dabei ist die Anwendung von Werkzeugen des ML nicht einfach nur die Anwendung der Werkzeuge, sondern auch das Verstehen der Fragestellungen. Damit können dann die Werkzeuge zielgerichtet angewendet werden.

Die Hauptaufgabe im Kapitel 4 der Ausarbeitung ist es aufzuzeigen, wie mit Hilfe des ML eine wichtige Aufgabe in der Physik, die Trennung von Signal und Hintergrund durchgeführt werden kann. Hierbei wird im wesentlichen, die Kontur einer Welle analysiert und zur Trennung herangezogen. Damit entspricht diese Aufgabe einer Bilderkennung. Dies liefert gute Ergebnisse, mit Genauigkeit, Präzision (Reinheit) und Trefferquote (Sensitivität, Effizienz) von etwa 99 %. Durch die Verwendung weiterer Kennzahlen der Welle konnte dies sogar noch leicht verbessert werden. Diese Aufgabe kann zu einem Experiment für das Physikalische Praktikum ausgebaut werden.

5.2. Einsatz von Verfahren des Maschinellen Lernens

Verfahren des ML können in verschiedenen Bereichen in der Physik eingesetzt werden, wie dies in Physik konkret Nr. 73 ([3]) detaillierter aufgeführt ist.

- Moderne physikalische Experimente liefern oftmals eine riesige Menge von Daten. Beispiel hierfür ist das LHC in der Teilchenphysik. Hierbei gibt es zwei wichtige Aufgaben: Zum einen sehr schnell erkennen, ob interessante Daten vorliegen, die sofort weiter bearbeitet werden sollten. Zum anderen die immens große Datenmenge durchsehen, ob interessante Sachverhalte erkannt werden. Dabei geht es darum, die Daten effizient zu analysieren beziehungsweise Muster zu erkennen. Damit werden diese Daten dann für die weitere Forschung aufbereitet.
- Auch im Bereich der Simulation können Methoden der KI eingesetzt werden.
- Ein weiteres Einsatzfeld ist die Optimierung von Experimenten, damit diese präziser gesteuert und überwacht werden können. Damit können die vorhandenen Ressourcen effizienter genutzt werden.

Dies sind einige wichtige Aufgabenfelder, die zeigen, dass der Einsatz von ML-Verfahren in den letzten Jahren an Bedeutung gewonnen hat und auch weiterhin an Bedeutung gewinnt. Daher ist es sinnvoll, bereits im Bachelorstudium dies in den Lehrplan zu integrieren.

5.3. Wie geht es weiter

Wie oben schon ausgeführt, ist diese Arbeit nur ein kleiner Einstieg in das ML. Es fehlen viele grundlegende Verfahren. Zudem ist nur die Binäre Klassifikation enthalten. Die Multi-Klassen Klassifikation und die Regression sind nur kurz erwähnt. Dies hätte jedoch den Umfang dieser Arbeit gesprengt. Für eine grundlegende Ausbildung im Themenfeld ML ist es jedoch notwendig, auch diese Verfahren kennenzulernen und mit Beispielen aus der Physik zu verknüpfen. Beispiele, die auch für Studierende im Bachelorstudium verständlich sind. Auch der Einsatz von ML-Verfahren im Physikalischen Praktikum ist zu überlegen, denn Methoden des ML sind aus der physikalischen Arbeit nicht mehr wegzudenken. Wobei es nicht nur in der Physik, sondern auch in vielen anderen Bereichen in Forschung und Wirtschaft wichtig ist. Daher sollte es wichtig sein, dass jede angehende Physikerin und Physiker ausreichend Kenntnisse darüber hat.

Daher ergeben sich folgende beispielsweise Aufgabenfelder:

- Aufbau eines Kurses oder Tutorials für die „Einführung in das Maschinelle Lernen“ für Bachelorstudierende (im 5. Semester), das sowohl in Deutsch, als auch in Englisch verfügbar ist. Damit kann es als Einführung auch für Studierende angewendet werden, die erst zum Masterstudium einsteigen.
- Aufbau weiterer Beispiele von einfachen physikalischen Aufgaben, die dann in Praktikumsversuchen für das Physikalische Praktikum umgesetzt werden können.

Literatur

- [1] Tim Ruhe. *Künstliche Intelligenz - Die Bedeutung von Daten*. https://www.dpg-physik.de/veroeffentlichungen/publikationen/physikkonkret/pk70_kuenstliche-intelligenz. Zugriffsdatum: 16-Dez-2024.
- [2] Matthias Bethge und Jakob Macke. „An der Schwelle zur Physik des neuronalen Lernens“. In: *Physik Journal* 23.12 (Dez. 2024), S. 24–28. ISSN: 16179439. (Besucht am 03. 10. 2024).
- [3] Gregor Kasieczka. *Physiknobelpreis 2024: Physik in der Künstlichen Intelligenz*. https://www.dpg-physik.de/veroeffentlichungen/publikationen/physikkonkret/pk73_physiknobelpreis_ki. Zugriffsdatum: 16-Dez-2024.
- [4] Martin Erdmann u. a. *Deep Learning for Physics Research*. Hallbergmoos: World Scientific, 2021. 327 S. ISBN: 978-981-12-8532-5.
- [5] Martin Erdmann, Thomas Hebbeker und Alexander Schmidt. *Statistische Methoden in der Experimentalphysik*. Pearson ph - Physik. Hallbergmoos: Pearson, 2020. 218 S. ISBN: 978-3-86894-391-7.
- [6] Martin Erdmann. „Erkenntnisgewinn durch moderne datengetriebene Methoden: Deep Learning“. In: *Physik Journal* 19.4 (2020), S. 31–36. ISSN: 16179439. URL: <https://pro-physik.de/zeitschriften/download/19777> (besucht am 04. 10. 2024).
- [7] Viviana Acquaviva. *Machine Learning for Physics and Astronomy*. Princeton Oxford: Princeton University Press, 2023. 259 S. ISBN: 978-0-691-20641-7.
- [8] Guido Van Rossum und Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.
- [9] Python Software Foundation. *Python*. <http://www.python.org>. [Zugegriffen: 19-Dez-2024]. 2024.
- [10] Jake VanderPlas. *Handbuch Data Science mit Python: grundlegende Tools für die Arbeit mit Daten*. Übers. von Knut Lorenzen und Jørgen W. Lang. 1. Auflage. Heidelberg: O'Reilly, 2024. 574 S. ISBN: 978-3-96009-225-4.
- [11] Thomas Kluyver u. a. „Jupyter Notebooks – a publishing format for reproducible computational workflows“. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Hrsg. von F. Loizides und B. Schmidt. IOS Press. 2016, S. 87–90.
- [12] Project Jupyter. *Jupyter Notebook*. <https://jupyter.org>. [Zugegriffen: 19-Dez-2024]. 2024.
- [13] Charles R. Harris u. a. „Array programming with NumPy“. In: *Nature* 585.7825 (Sep. 2020), S. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: <https://doi.org/10.1038/s41586-020-2649-2>.

- [14] Project NumPy. *NumPy*. <https://numpy.org>. [Zugegriffen: 19-Dez-2024]. 2024.
- [15] The pandas development team. *Pandas*. <https://pandas.pydata.org>. [Zugegriffen: 19-Dez-2024]. 2024.
- [16] J. D. Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in Science & Engineering* 9.3 (2007), S. 90–95. DOI: 10.1109/MCSE.2007.55.
- [17] Matplotlib. *Matplotlib*. <https://matplotlib.org/>. [Zugegriffen: 12-Jan-2025]. 2025.
- [18] Michael L. Waskom. „seaborn: statistical data visualization“. In: *Journal of Open Source Software* 6.60 (2021), S. 3021. DOI: 10.21105/joss.03021. URL: <https://doi.org/10.21105/joss.03021>.
- [19] Seaborn. *Seaborn*. <https://seaborn.pydata.org>. [Zugegriffen: 19-Dez-2024]. 2024.
- [20] F. Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011). [Zugegriffen: 19-Dez-2024], S. 2825–2830.
- [21] Scikit-learn. *Scikit-learn*. <https://scikit-learn.org>. [Zugegriffen: 19-Dez-2024]. 2024.
- [22] Adam Paszke u. a. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. <https://pytorch.org>. [Zugegriffen: 19-Dez-2024]. 2019.
- [23] Martín Abadi u. a. *TensorFlow: A System for Large-Scale Machine Learning*. <https://www.tensorflow.org>. [Zugegriffen: 19-Dez-2024]. 2016.
- [24] Jörg Frochte. *Maschinelles Lernen: Grundlagen und Algorithmen in Python*. 3., überarbeitete und erweiterte Auflage. Plus.Hanser-Fachbuch. München: Hanser, 2021. 608 S. ISBN: 978-3-446-46144-4.
- [25] Chi Nhan Nguyen und Oliver Zeigermann. *Machine Learning - kurz & gut*. 2. Auflage, 1., korrigierter Nachdruck. kurz & gut. Heidelberg: O'Reilly, 2022. 208 S. ISBN: 978-3-96009-161-5.
- [26] Tom Fawcett. „An Introduction to ROC Analysis“. In: *Pattern Recognition Letters* 27.8 (Juni 2006), S. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010. URL: <https://linkinghub.elsevier.com/retrieve/pii/S016786550500303X> (besucht am 03. 10. 2024).
- [27] Yaml. *Yaml*. <https://yaml.org/>. [Zugegriffen: 12-Jan-2025]. 2025.
- [28] Sebastian Raschka und Vahid Mirjalili. *Python Machine Learning - Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow2*. 3. Birmingham - Mumbai: Packt Publishing, 2019. 742 S. ISBN: 978-1-78995-575-0.
- [29] Aurélien Géron. *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. Übers. von Kristian Rother und Thomas Demmig. 3., aktualisierte und erweiterte Auflage. Heidelberg: O'Reilly, 2023. 876 S. ISBN: 978-3-96009-212-4.

A. Wörterbuch

A.1. Wörterbuch Deutsch - Englisch

Tabelle A.1.: Wörterbuch Deutsch - Englisch

α -Fehler	α -error
β -Fehler	β -error
bestärktes Lernen	reinforcement learning
Bewertungsmaßstab	evaluation metric
binäre Klassifizierung	binary classification
Clustering	clustering
Effizienz	efficiency
eifriges Lernen	eager learning
Eigenschaft	feature
Entscheidungsbaum	Decision Tree
Erfolgsrate	rate of success (score)
Ergebnis	target
erkennen	recognize
F_1 -Metrik	F_1 -score
F_β -Metrik	F_β -score
Falsch-Positiv-Rate	false positive rate
faules Lernen	lazy learning
Fehlerrate	rate of failure (error)
Fläche unter der Kurve	area under the curve
Genauigkeit	accuracy
Generalisierungserfolg	generalization score
Generalisierungsfehler	generalization error
Grundklasse	ground truth
gruppieren	group together
Halbwertsbreite	full width at half maximum
Hintergrund	background
Hyperparameter	hyper parameter
Instanz	instance
k-Nächster Nachbar	k-Nearest Neighbour
Klassifikation	classification
Künstliche Intelligenz	artificial intelligence (AI)

...

Tabelle A.1.: Wörterbuch Deutsch - Englisch, Fortsetzung

Lerndaten	learning set
Lineare Regression	linear regression
Logistische Regression	logistics regression
Maschinelles Lernen	machine learning
Maschinelles Lernen Modell	machine learning model
Multi-Klassen Klassifizierung	multi-class classification
Neuronales Netz	neuronal network
Operationscharakteristik	receiver operating characterisic curve
Perzeptor	perceptor
Präzision	precision
Reduktion von Dimensionen	dimensionality reduction
regelbasiertes System	rule based system
Regression	regression
Reinheit	purity
Schwellwert	threshold
Sensitivität	sensitivity
Signal	signal
Spezifität	specificity
Testdaten	test set
Testerfolgsrate	test score
Testsfehlerrate	test error
tiefes Lernen	deep learning
Trainingsdaten	training set
Trainingserfolgsrate	training score
Trainingsfehlerrate	training error
Trefferquote	recall
überwachtes Lernen	supervised learning
unüberwachtes Lernen	unsupervised learning
vereinfachen	simplify
vorbereiten	prepare
vorhersagen	predict
Wahr-Positiv-Rate	true positive rate

A.2. Wörterbuch Englisch - Deutsch

Tabelle A.2.: Wörterbuch Englisch - Deutsch

α -error	α -Fehler
accuracy	Genauigkeit
area under the curve	Fläche unter der Kurve
artificial intelligence (AI)	Künstliche Intelligenz
β -error	β -Fehler
background	Hintergrund
binary classification	binäre Klassifizierung
classification	Klassifikation
clustering	Clustering
Decision Tree	Entscheidungsbaum
deep learning	tiefes Lernen
dimensionality reduction	Reduktion von Dimensionen
eager learning	eifriges Lernen
efficiency	Effizienz
evaluation metric	Bewertungsmaßstab
F_1 -score	F_1 -Metrik
F_β -score	F_β -Metrik
false positive rate	Falsch-Positiv-Rate
feature	Eigenschaft
full width at half maximum	Halbwertsbreite
generalization score	Generalisierungserfolg
generalization error	Generalisierungsfehler
ground truth	Grundklasse
group together	gruppieren
hyper parameter	Hyperparameter
instance	Instanz
k-Nearest Neighbour	k-Nächster Nachbar
lazy learning	faules Lernen
learning set	Lerndaten
linear regression	Lineare Regression
logistics regression	Logistische Regression
machine learning	Maschinelles Lernen
machine learning model	Maschinelles Lernen Modell
multi-class classification	Multi-Klassen Klassifizierung
neuronal network	Neuronales Netz
perceptor	Perzeptor
precision	Präzision
predict	vorhersagen
prepare	vorbereiten
purity	Reinheit

...

Tabelle A.2.: Wörterbuch Englisch - Deutsch, Fortsetzung

rate of failure (error)	Fehlerrate
rate of success (score)	Erfolgsrate
recall	Trefferquote
receiver operating characterisic curve	Operationscharakteristik
recognize	erkennen
regression	Regression
reinforcement learning	bestärktes Lernen
rule based system	regelbasiertes System
sensitivity	Sensitivität
signal	Signal
simplify	vereinfachen
specificity	Spezifität
supervised learning	überwachtes Lernen
target	Ergebnis
test error	Testsfehlerrate
test score	Testerfolgsrate
test set	Testdaten
threshold	Schwellwert
training error	Trainingsfehlerrate
training score	Trainingserfolgsrate
training set	Trainingsdaten
true positive rate	Wahr-Positiv-Rate
unsupervised learning	unüberwachtes Lernen

B. Bilder von Wellen (Signal / Hintergrund)

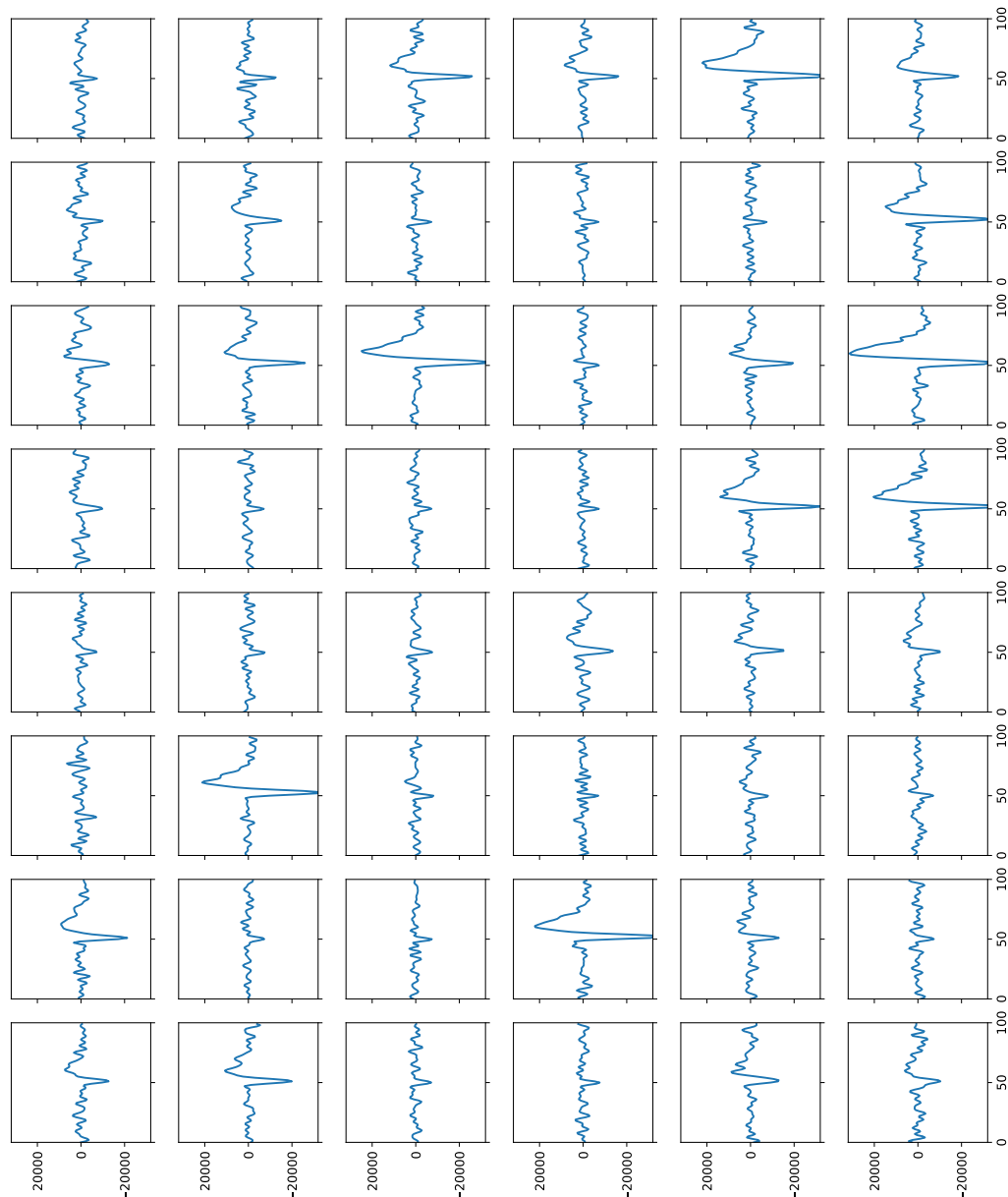


Abbildung B.1.: Beispiele von Signal-Hintergrund-Wellen aus dem Datensatz, ohne Kennzeichnung, ob es sich um ein Signal oder ein Hintergrund handelt.

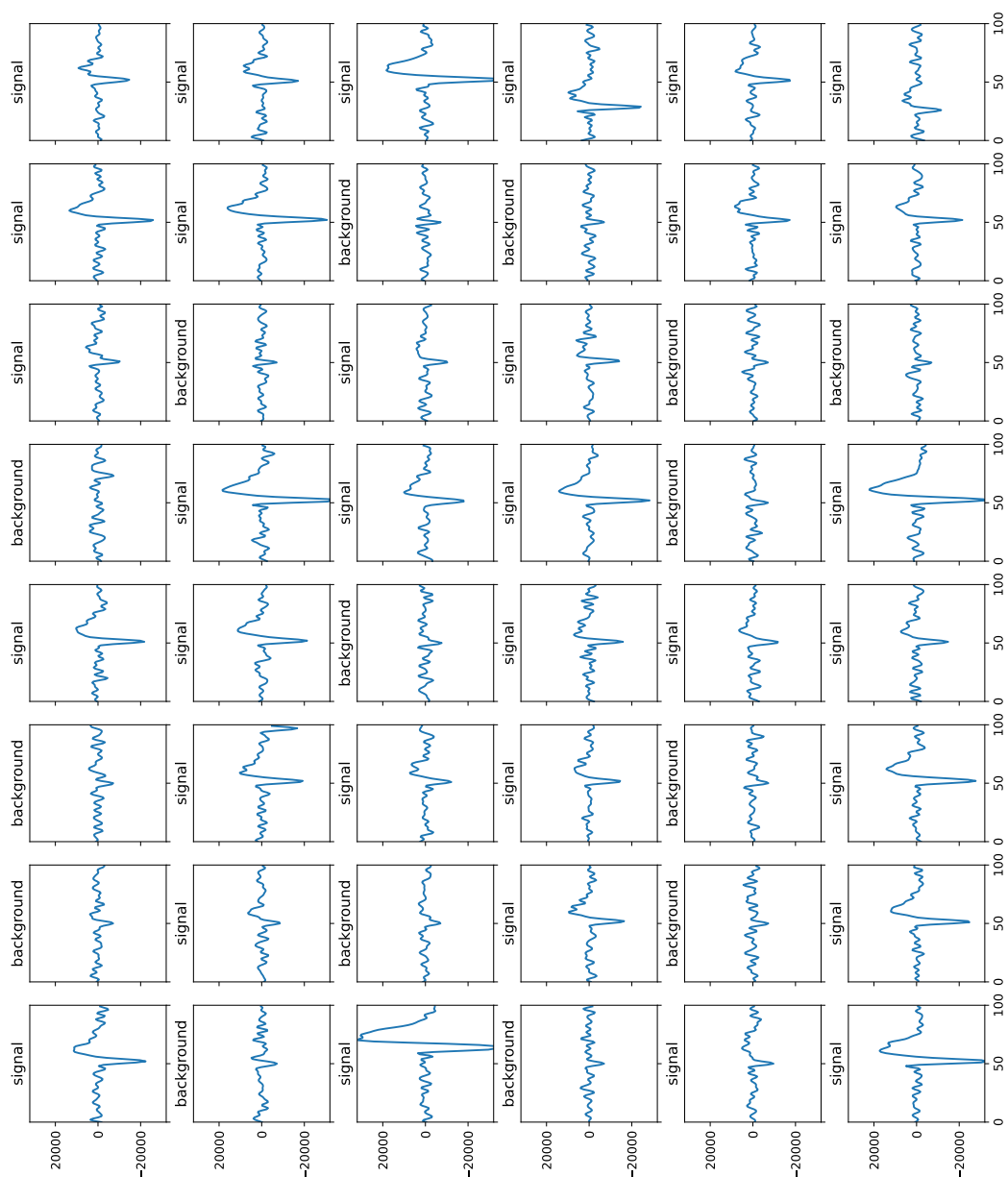


Abbildung B.2.: Beispiele von Signal-Hintergrund-Wellen aus dem Datensatz, mit Kennzeichnung, ob es sich um ein Signal oder ein Hintergrund handelt.

C. Liste von Jupyter Notebooks

Die ganzen Dateien sind im github-Verzeichnis <https://github.com/Zeh-Marschke/MLe> hinterlegt. Es sind hauptsächlich Jupyter Notebooks und einige Python-Dateien. Darüber hinaus sind Datendateien, die verwendet werden gespeichert.

C.1. Liste von Dateien für die ersten Maschinellen Lernen-Verfahren

- **cha02-MaschinellesLernen.ipynb** JN für die Definition grundlegender Begriffe des ML. Darüber hinaus werden einige Begriffe anhand einer Linearen Regression erläutert.
- **tools.py** Hilfsroutinen für die Lineare Regression.
- **cha03-BinClass-ErsteModelle.ipynb** JN in dem Begriffe für eine Binäre Klassifikation erläutert werden. Es werden zwei einfache binäre Klassifizierungsverfahren (Entscheidungsbaum und n-Nächster-Nachbar) eingeführt. Die Methoden werden an einem kleinen Beispiel von wenigen potenziell bewohnbaren Exoplaneten erläutert.
- **cha04-BinClass-Bewertung.ipynb** JN in dem Begriffe für die Bewertung von Methoden der Binären Klassifikation definiert und erläutert. Als Beispieldaten werden eine Liste von Exoplaneten, die potenziell bewohnbar sind, verwendet. Es werden die ML-Verfahren Entscheidungsbaum und k-Nächster Nachbar angewendet.
- **data/HabitablePlanets_LearningSet.csv** Liste von 18 Exoplaneten, die im Jupyter-Notebook **cha03-BinClass-ErsteModelle.ipynb** verwendet werden.
- **data/HabitablePlanets_complete.csv** Liste von 5350 Exoplaneten (davon 70 möglicherweise bewohnbare Exoplaneten). Die Daten wurden von der Webseite <https://phl.upr.edu/hwc> des **Planet Habitability Laboratory**, an der University of Puerto Rico in Arecibo am 22.08.2024 exportiert. Die Daten werden im Jupyter-Notebook **cha04-BinClass-Bewertung.ipynb** verwendet.

C.2. Liste von Dateien für die Signal-Hintergrund-Trennung

- **sb01-ym12csv.ipynb** Wandelt Rohdaten von Wellen (Signal / Hintergrund) in eine CSV-Datei um. Jede Zeile besteht aus 100 Pixel der Zählrate und das Label (Signal oder Hintergrund)
- **sb02-festeng.ipynb** Aus den 100-Pixel-Daten werden die Kontur (25-Pixel-Auszug) und weitere Merkmale ermittelt und in eine CSV-Datei geschrieben.
- **sb03-kontur.ipynb** Es werden ML-Verfahren auf die 25-Pixel-Kontur angewendet.
- **sb04-features.ipynb** Es werden ML-Verfahren für Merkmalen der Welle angewendet.
- **sb05-konturplus.ipynb** Es werden ML-Verfahren auf die 25-Pixel-Kontur, ergänzt um Merkmalen angewendet.
- **sb09-plotexamples.ipynb** Es werden Wellen geplottet, wobei verschiedenen Einstellungen gesetzt werden können.
- **general_parameters.py** Allgemeine Parameter, die in allen JN benötigt werden.
- **tools_configmatrix.py** Ausgabe von Wertungsdaten einer Binäre Klassifikation in verschiedenen Formaten (ausführlich Wahrheitsmatrix mit Kennzahlen, Kennzahlen in einer Zeile).
- **tools_showwaves.py** Ausgaberroutine für das Plotten von Wellen.
- **data/rawSignal.zip** Zip-Datei mit den yaml-Daten von aufgezeichneten Signalen. Die Datei wird im JN **sb01-ym12csv** verwendet.
- **data/rawBackground.zip** Zip-Datei mit den yaml-Daten von aufgezeichneten Hintergrundwellen. Die Datei wird im JN **sb01-ym12csv** verwendet.

Index

- F_1 -Metrik, 16
- F_β -Metrik, 16
- α -Fehler, 15
- bestärkendes Lernen, 6
- Bewertungsmaßstab, 12
- Binäre Klassifikation, 12
- binäre Klassifizierung, 6
- Clustering, 7
- Deep Learning, 2
- Effizienz, 14
- eifrigen Lernen, 7
- Eigenschaften, 8
- Entscheidungsbaum, 12
- Erfolgsrate, 10
- Ergebnis, 8
- erkennen, 5
- Falsch-Positiv-Rate, 15
- faulen Lernen, 7
- Fehlerrate, 9
- Fläche unter der Kurve, 15
- Genauigkeit, 13
- Generalisierungserfolg, 10
- Generalisierungsfehler, 10
- Grundklasse, 13
- gruppieren, 5
- Hyperparameter, 11
- Instanzen, 8
- k-Nächster Nachbar, 12
- Klasse, 6
- Klassifikation, 6
- Künstlichen Intelligenz, 1
- Lerndaten, 9
- Logistische Regression, 23
- Maschinelles Lernen, 1, 5
- Maschinelles Lernen Modell, 9
- Multi-Klassen Klassifizierung, 6
- Neuronales Netz, 2, 24
- Operationscharakteristik, 15
- Perzeptron, 24
- Präzision, 14
- Reduktion von Dimensionen, 8
- Regression, 7
- Reinheit, 14
- Schwellwert, 14
- Sensitivität, 14, 15
- Spezifität, 15
- Testdaten, 9
- Testerfolgsrate, 10
- Testsfehlerrate, 10
- tiefe Lernen, 2
- Trainingsdaten, 9
- Trainingserfolgsrate, 10
- Trainingsfehlerrate, 10
- Trefferquote, 14
- unüberwachtes Lernen, 6
- vereinfachen, 6
- vorhersagen, 5
- Wahr-Positiv-Rate, 15
- überwachtes Lernen, 6