Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

TITEL

Explorative Analyse von Daten aus der nanoporebasierten DNA-Sequenzierung zur Identifikation von Verdünnungseffekten

AUTOR

Christoph Stach

E-MAIL

s0555912@htw-berlin.de

PROJEKT

Masterarbeit

EINRICHTUNG

Hochschule für Technik und Wirtschaft Berlin

ABTEILUNG

Fachbereich 4 - Angewandte Informatik

BETREUER

Prof. Dr.-Ing. Piotr Wojciech Dabrowski

Alexander Hinzer

Zusammenfassung

Dieses Projekt erforscht die Möglichkeit, den Verdünnungsgrad des SARS-CoV-2-Virus in Patientenproben mithilfe von maschinellem Lernen vorherzusagen. Dabei werden Rohdaten analysiert, die durch MinION-Sequenzierung und Amplifikation mittels des ARTIC-Protokolls gewonnen wurden. Im Fokus steht eine explorative Analyse, bei der eine Vielzahl von Techniken zum Einsatz kommt, darunter Datenbereinigung, Feature Engineering, Outlier-Detection und Korrelationsanalyse. Das Ziel der Arbeit besteht darin, zu ermitteln, ob der durch CT-Werte indirekt gekennzeichnete Verdünnungsgrad aus den Sequenzierungsdaten abgeleitet werden kann, um ein tieferes Verständnis der Datenstruktur und ihrer möglichen Bedeutung zu erlangen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

NGS	Next-generation Sequencing
DNA	Deoxyribonucleic acid
PCR	Polymerase chain reaction
SNP	Single-nucleotide polymorphism
INDEL	[In]sertion or [Del]etion of a block of nucleotides in a DNA sequence
SAM	Sequence Alignment/Map format
BAM	Binary form of a SAM file

1 Einleitung

In diesem Kapitel werden die Motivation und die Zielsetzung für die Analyse von Rohdaten aus einem Datensatz im Bereich der Nanoporensequenzierung dargelegt.

1.1 Motivation

Die Untersuchung eines Rohdatensatzes erweist sich als sinnvoll, da hierbei möglicherweise Muster oder Korrelationen innerhalb der Daten aufgedeckt werden können. Selbst wenn keine konkreten Ergebnisse erzielt werden können, ermöglicht eine explorative Analyse einen tieferen Einblick und ein besseres Verständnis der Daten.

1.2 Zielsetzung

Das Ziel des Projekts besteht darin zu ermitteln, ob sich der Verdünnungsgrad des SARS-CoV-2-Virus in den Rohdaten von Patientenproben, die mit MinION sequenziert und mittels des ARTIC-Protokolls amplifiziert wurden, durch maschinelles Lernen vorhersagen lässt.

1.3 Randbedingungen

Bei der Analyse der Daten soll ausschließlich auf die Rohsignale zurückgegriffen werden. Es sollen keine Features verwendet werden, die durch nachgelagerte Algorithmen, wie beispielsweise das Basecalling, Alignment/Mapping oder Variant Calling, generiert werden.

2 Grundlagen

Die folgenden Abschnitte konzentrieren sich auf die Grundlagen der Bioinformatik und die DNA-Sequenzierung. Sie bieten Einblicke in Funktion und Struktur der DNA und beleuchten Technologien zur Analyse. Darüber hinaus erläutern sie die üblichen Schritte zur Erstellung einer explorativen Analyse.

2.1 Bioinformatik

Im Folgenden gewährt diese Arbeit einen Überblick über die Bioinformatik. Dabei werden zentrale Themen behandelt und gängige, die gängigen Schritte erläutert, die zur Analyse von DNA-Daten notwendig sind.

2.1.1 DNS / DNA

Die Desoxyribonukleinsäure (DNS), bekannt als DNS im Deutschen und DNA (Deoxyribonucleic Acid) im Englischen, ist ein essenzieller Bestandteil aller Lebewesen und enthält die Anweisungen, welche für die Entwicklung, Funktion und Fortpflanzung von Zellen notwendig sind. Sie setzt sich aus langen Ketten kleiner Einheiten zusammen, die als Basen bekannt sind und sich in einer Doppelhelixstruktur anordnen. Diese Basen – Adenin, Thymin, Cytosin und Guanin – sind in spezifischen Mustern angeordnet, die als Gene bezeichnet werden. [1]

Die Gene, lokalisiert auf der DNA, enthalten die Anweisungen für die Synthese von Proteinen, welche für zelluläre Funktionen und die Vererbung von Merkmalen von Eltern an Nachkommen verantwortlich sind. Die in den Genen enthaltenen Anweisungen werden von Zellen interpretiert und in Proteine umgewandelt. Diese Proteine sind für diverse Funktionen in unserem Körper zuständig, beispielsweise für die Kontrolle der Zellteilung und die Regulation von Stoffwechselvorgängen. [2]

Ein bedeutsamer Aspekt der DNA ist ihre Fähigkeit zur Replikation und Übertragung

von einer Zelle in die nächste. Dies ermöglicht Lebewesen, sich fortzupflanzen und ihre Gene an die folgende Generation weiterzugeben. [3]

Gelegentlich können während des Replikationsprozesses Fehler auftreten, die als Mutationen bezeichnet werden. Diese Fehler entstehen, wenn Basen während der DNA-Synthese falsch eingefügt, gelöscht oder ausgetauscht werden. Mutationen können spontan oder durch Umweltfaktoren, wie beispielsweise Strahlung oder chemische Substanzen, verursacht werden. Während einige Mutationen neutral oder sogar vorteilhaft für das betroffene Individuum sein können, können andere negative Auswirkungen haben, die zu Krankheiten oder genetischen Störungen führen. [4]

2.1.2 NGS: Next Generation Sequencing

Next-Generation Sequencing (NGS) bezieht sich auf eine Gruppe von Technologien, die das schnelle und effiziente Sequenzieren großer Mengen von DNA oder RNA ermöglichen. Im Gegensatz zu älteren Sequenzierungsmethoden, wie der Sanger-Sequenzierung, welche nur in der Lage ist, DNS in kleinen Fragmenten zu sequenzieren, kann NGS Millionen von DNS-Fragmenten gleichzeitig sequenzieren. Dadurch können beispielsweise Genome von Organismen deutlich schneller und kosteneffektiver sequenziert werden. [5]

In den letzten Jahren haben NGS-Technologien die Genomforschung revolutioniert und ermöglichten die Beantwortung komplexer biologischer Fragestellungen, die zuvor undenkbar waren. Sie trugen auch zur Entdeckung neuer Gene und Genvarianten bei und ermöglichten die Identifizierung von Krankheitsursachen für Diagnose und Therapie. NGS findet Anwendung in der medizinischen Forschung und Diagnostik, der Agrar- und Lebensmittelindustrie sowie in der Umweltüberwachung und Forensik. [6]

Auf dem Markt existiert eine Vielzahl von NGS-Technologien, darunter Illumina, Pacific Biosciences, Oxford Nanopore Technologies und Ion Torrent. Jede Technologie weist eigene Vor- und Nachteile hinsichtlich Durchsatz, Länge der Sequenzierungsreads, Fehlerhäufigkeit und Kosten auf. [7]

2.1.3 Nanopore-Sequenzierung

Nanopore-Sequenzierung ist ein Verfahren zur Bestimmung der Abfolge von Basen in DNS-Molekülen. Das Verfahren basiert auf der Verwendung von engen Poren, durch die ein DNS-Strang gezogen wird. Während der DNS-Strang durch die Pore gezogen wird, wird ein elektrisches Signal erzeugt, das von der chemischen Struktur der gerade durch

die Pore gezogenen Basen abhängt. Durch die Messung dieser Signale kann die Abfolge der Basen in der DNS bestimmt werden.

Ein großer Vorteil von Nanopore-Sequenzierung gegenüber anderen Methoden wie Illumina-Sequenzierung ist, dass keine Voraufbereitung der DNS-Probe erforderlich ist. Mit der Illumina-Sequenzierung wird die DNA in kleinere Fragmente geschnitten und dann vervielfältigt, bevor sie sequenziert wird. Dieser Schritt ist bei der Nanopore-Sequenzierung nicht erforderlich, sodass lange DNS-Fragmente direkt sequenziert werden können.

Ein weiterer Vorteil der Nanopore-Sequenzierung ist, dass sie in Echtzeit durchgeführt werden kann. Es gibt keine Notwendigkeit, die Signale zu speichern und später zu analysieren, wie es bei anderen Verfahren der Fall ist. Dies ermöglicht es, schnell auf Veränderungen in der DNA-Sequenz zu reagieren, was für Anwendungen wie die Diagnose von Krankheiten oder die Überwachung von Infektionskrankheiten von großer Bedeutung sein kann.

2.1.4 CT-Wert

Der CT-Wert, auch bekannt als „Cycle Threshold“, ist eine wichtige Metrik im Zusammenhang mit der Polymerase Chain Reaction (PCR). Die PCR ist eine Methode, die dazu dient, spezifische DNA-Sequenzen in einer Probe zu vervielfältigen. Bei diesem Prozess verdoppelt die PCR die Menge an DNA in der Probe mit jedem Zyklus. Der CT-Wert gibt die Anzahl der Zyklen an, die benötigt werden, um genug DNA zu erzeugen, sodass ein vorgegebener Schwellenwert erreicht wird.

Während der COVID-19-Pandemie wurde der CT-Wert verwendet, um die Viruslast in den Testproben abzuschätzen. Ein niedriger CT-Wert wies auf eine hohe Viruslast hin, während ein hoher Wert auf eine geringere Viruslast hindeutete.

2.1.5 Analyseschritte in der Bioinformatik

Die Rohdaten, die von einem Sequenzierer erzeugt werden, werden in der Regel nicht direkt analysiert. Stattdessen werden zusätzliche Schritte eingeleitet, um die Sequenzen besser analysieren zu können.

2.1.6 ARTIC Network

Das ARTIC Network ist ein internationales Konsortium, das sich der Verbesserung und Standardisierung der genetischen Sequenzierung von RNA-Viren widmet. Es hat besondere

Bekanntheit durch die Entwicklung eines umfassenden Amplifikations- und Sequenzierungsprotokolls für das SARS-CoV-2-Virus erlangt. Dieses Protokoll wird in Forschungslabors eingesetzt, um das Genom von SARS-CoV-2 zu sequenzieren und zu untersuchen.

Basecalling

Sobald eine Probe durch einen Sequenzierer verarbeitet wurde, resultieren Rohdaten, die in der Regel schwer zu interpretieren sind. Um diese Daten nutzbar zu machen, wird ein Basecalling-Algorithmus angewendet. Der grundlegende Prozess besteht darin, die während der Sequenzierung generierten Rohdaten in eine lesbare Zeichenkette von DNA-Basen zu transformieren.

Darüber hinaus nehmen gängige Basecalling-Algorithmen nicht nur die Identifikation der Basen vor, sondern dokumentieren auch die Zuverlässigkeit ihrer Bestimmung. Das bedeutet, sie geben eine Art Qualitätsmaß an, das aufzeigt, mit welcher Sicherheit sie eine bestimmte Base identifiziert haben.

Sequenzalignment

Das Basecalling wird üblicherweise vom Sequenzalignment gefolgt. Das Sequenzalignment dient dazu, Gemeinsamkeiten zwischen zwei oder mehr Sequenzen festzustellen. Im Rahmen dieses Verfahrens werden die durch den Sequenzierer erzeugten Reads mit einer Referenzsequenz abgeglichen und entsprechend den Positionen in dieser Referenzsequenz ausgerichtet.

Variantcalling

Nachdem die Sequenzdaten ausgerichtet wurden, kann der nächste Schritt in der bioinformatischen Analyse das Variant Calling sein. Dieser Prozess beinhaltet die Identifizierung von Stellen in den Sequenzen, an denen Variationen im Vergleich zu einer Referenzsequenz auftreten. Diese Variationen können einzelne Nukleotidpolymorphismen (SNPs), Insertionen, Deletionen oder andere Arten von strukturellen Varianten umfassen.

2.1.7 Übliche Datenformate

Im Bereich der Bioinformatik und DNA-Sequenzierung kommen häufig die folgenden Dateiformate zum Einsatz:

- FASTQ: Dieses Format beinhaltet die sequenzierten Basen sowie die Qualitätswerte jeder einzelnen Base.
- FASTA: Dieses Format enthält die Basen einer bereits bekannten Zielsequenz.
- SAM (Sequence Alignment/Map): Ein nicht komprimiertes Format, das die Sequenzierungs-Reads in Form von Alignments darstellt.
- BAM (Binary Alignment/Map): Ein komprimiertes Format, das ebenfalls die Sequenzierungs-Reads in Form von Alignments beinhaltet.
- VCF (Variant Call Format): Enthält Informationen über genetische Variationen innerhalb einer DNA-Sequenz.

2.2 Explorative Analyse

Die explorative Analyse ist ein entscheidender Bereich in diesem Projekt. Sie umfasst die Vorbereitung der Rohdaten, die Erzeugung und Untersuchung neuer Features, die Erkennung und Bereinigung von Ausreißern und die Bewertung von Korrelationen zwischen verschiedenen Merkmalen. Aufbauend darauf können Klassifizierungsverfahren zum Einsatz kommen. Deep-Learning-Techniken können als alternativer Lösungsansatz bei der Analyse eine Rolle spielen.

2.2.1 Datenvorbereitung

Bevor die explorative Analyse beginnen kann, erfolgt eine sorgfältige Aufbereitung der Rohdaten. Oft verteilen sich diese Daten über verschiedene Dateien, daher sind Indizes nützlich, die den Speicherort der Daten angeben. Es ist ratsam, die Rohdaten in ein leichter lesbares Datenformat umzuwandeln. In vielen Fällen enthalten Rohdatendateien Metadaten, deren Relevanz variiert. Entsprechend können diese Metadaten bei der Übertragung in das gewählte Format berücksichtigt oder ignoriert werden.

2.2.2 Erstellung eines Unterdatensatzes / Binning

Ein wesentlicher Schritt in der explorativen Analyse ist das Erstellen von Unterdatensätzen oder Binning. Diese Methode wird genutzt, um die Komplexität der Daten zu reduzieren und sie so besser handhabbar zu machen. Bei der Erstellung von Unterdatensätzen wird eine große Datenmenge in kleinere, leichter zu analysierende Segmente unterteilt.

Das Binning ist ein ähnlicher Prozess, bei dem kontinuierliche Daten in diskrete „Bins“ oder Gruppen eingeteilt werden. Dies kann besonders hilfreich sein, um Muster und Trends in den Daten zu erkennen, die sonst durch den Rauschanteil verdeckt sein könnten. Beide Verfahren, das Erstellen von Unterdatensätzen und das Binning, tragen dazu bei, die Daten auf ein handhabbares Maß zu reduzieren und sie für weitere Analyseprozesse vorzubereiten.

2.2.3 Feature Engineering

Das Feature Engineering spielt eine bedeutende Rolle in der Datenanalyse und kann entscheidend zur Leistungsfähigkeit von Modellen beitragen. In diesem Schritt werden bestehende Merkmale (Features) der Daten transformiert oder neue Merkmale aus den vorhandenen Daten erzeugt, um die Informationen für die nachfolgenden Schritte effektiver nutzbar zu machen.

Die Erzeugung neuer Features kann auf vielfältige Weisen erfolgen, beispielsweise durch Kombinationen existierender Merkmale, durch Zerlegung von Merkmalen in ihre Bestandteile oder durch Anwendung mathematischer Funktionen. Ziel ist es stets, neue Perspektiven auf die Daten zu eröffnen und verborgene Zusammenhänge zu entdecken.

2.2.4 Outlier-Detection

Die Erkennung von Ausreißern (Outlier Detection) ist ein wichtiger Schritt in der explorativen Datenanalyse. Ausreißer sind Beobachtungen, die sich signifikant von den übrigen Datenpunkten unterscheiden. Sie können auf Messfehler, Dateneingabefehler oder auf tatsächliche, aber ungewöhnliche Ereignisse zurückzuführen sein.

Ausreißer können die Ergebnisse statistischer Analysen verzerren und das Training von maschinellen Lernmodellen beeinträchtigen. Deshalb ist es wichtig, sie frühzeitig zu erkennen und angemessen zu behandeln. Dabei können sie entweder korrigiert, ignoriert oder gesondert untersucht werden, je nachdem, was in dem jeweiligen Kontext sinnvoll ist.

Es gibt verschiedene Techniken zur Ausreißererkennung, von einfachen statistischen Methoden, die auf der Verteilung der Daten basieren, bis hin zu komplexen maschinellen Lernverfahren.

2.2.5 Korrelationsmatrix

Die Korrelationsmatrix ist ein effektives Werkzeug in der explorativen Datenanalyse. Sie zeigt die Korrelationen oder Zusammenhänge zwischen verschiedenen Merkmalen (Features) in einem Datensatz auf. Jeder Eintrag in der Matrix gibt den Korrelationskoeffizienten zwischen zwei Merkmalen an. Dieser Wert kann zwischen -1 und +1 liegen, wobei -1 eine perfekte negative Korrelation (wenn ein Merkmal steigt, sinkt das andere), +1 eine perfekte positive Korrelation (wenn ein Merkmal steigt, steigt auch das andere) und 0 keine Korrelation darstellt.

Die Erstellung einer Korrelationsmatrix kann dabei helfen, versteckte Beziehungen zwischen Merkmalen zu erkennen, die für die nachfolgende Analyse oder Modellierung von Bedeutung sein könnten.

Eine Korrelationsanalyse kann sowohl auf univariater als auch auf multivariater Ebene durchgeführt werden. Bei einer univariaten Analyse betrachten wir jeweils nur zwei Merkmale und deren Korrelation miteinander. Eine multivariate Korrelationsanalyse hingegen bezieht sich auf die gleichzeitige Betrachtung von mehr als zwei Merkmalen.

2.2.6 Klassifizierungsverfahren

Klassifizierungsverfahren sind ein zentraler Bestandteil des maschinellen Lernens und der Datenanalyse. Sie dienen dazu, Muster und Strukturen in den Daten zu erkennen, die zur Klassifikation von neuen, bisher unbekannten Datenpunkten genutzt werden können.

Zu den gängigsten Klassifizierungsverfahren gehören Entscheidungsbäume, Random Forests, Support Vector Machines, Naive Bayes-Klassifikatoren, k-Nearest Neighbor (kNN) und neuronale Netze. Diese Verfahren basieren auf unterschiedlichen Grundprinzipien und Annahmen und bieten verschiedene Vorteile in Bezug auf Genauigkeit, Robustheit, Interpretierbarkeit und Rechenleistung.

2.2.7 Deep Learning

Deep Learning, ein Unterbereich des maschinellen Lernens, konzentriert sich auf die Konstruktion und Anwendung von künstlichen neuronalen Netzwerken. Diese Netzwerke, inspiriert von der Arbeitsweise des menschlichen Gehirns, bestehen aus miteinander verbundenen Einheiten, den sogenannten Neuronen. Diese Neuronen repräsentieren die kleinsten Bausteine in einem neuronalen Netz. Sie nehmen Eingabewerte auf, multiplizieren diese

mit zugeordneten Gewichten und summieren die Ergebnisse. Zusätzlich wird ein sogenannter Bias hinzugefügt. Das resultierende Signal wird durch eine Aktivierungsfunktion geführt.

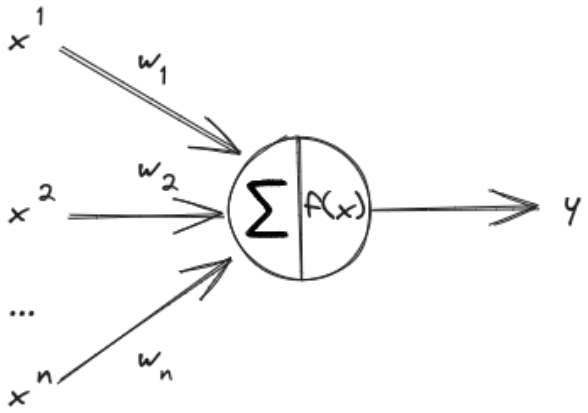


Abbildung 2.1: Visualisierung eines Neurons (auch Perzeptron genannt)

Ein neuronales Netzwerk ist eine Ansammlung dieser grundlegenden Einheiten (Neuronen). Sie können auf vielfältige Weise miteinander vernetzt sein, was zu einer breiten Palette von Netzwerkstrukturen führt. Das vollständige neuronale Netzwerk wird dann typischerweise mit umfangreichen Datensätzen trainiert. Der mathematische Algorithmus der Backpropagation passt dabei die Gewichte des Netzwerks iterativ an, um den Fehler (auch als „Loss“ bezeichnet) zwischen den von dem Netzwerk berechneten Ausgaben und den tatsächlichen Zielwerten zu minimieren.

2.2.8 Neuronale Netzwerkarchitekturen

In künstlichen neuronalen Netzwerken sind Neuronen häufig in Schichten, auch als „Layer“ bezeichnet, organisiert. Zwei gängige Arten von Schichten sind die Fully-Connected Layer und die Convolutional Layer.

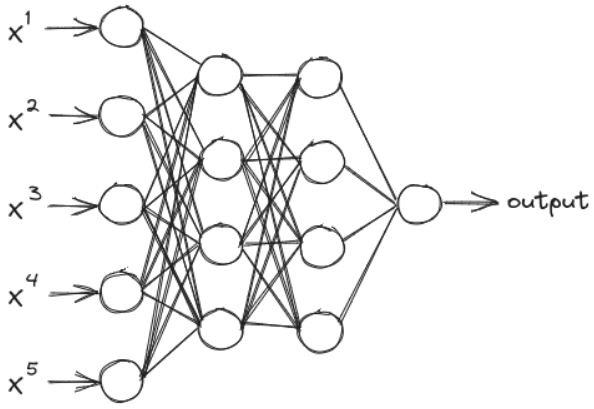


Abbildung 2.2: Beispiel eines Fully-Connected Netzwerks mit 5 Eingabeneuronen, 2 Hidden Layern mit jeweils 4 Neuronen und einem Ausgabeneuron

Fully-Connected Layer sind Schichten, in denen jedes Neuron mit jedem Neuron in der vorhergehenden Schicht verbunden ist. Diese Art von Layer wird häufig in traditionellen neuronalen Netzwerken eingesetzt.

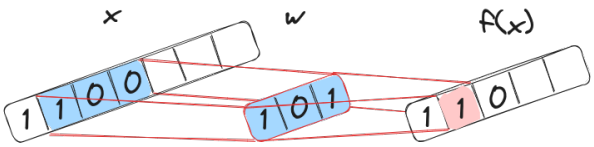


Abbildung 2.3: Visualisierung einer 1D Convolution mit einer Kernelgröße von 3

Convolutional Layer, andererseits, sind von der menschlichen Wahrnehmung inspiriert. Sie „blicken“ nur auf kleine, lokale Bereiche der Eingabedaten. Diese Methode ist besonders nützlich für die Verarbeitung von zweidimensionalen Daten wie Bildern, da sie Strukturen innerhalb des Bildes erkennen können, unabhängig von ihrer Position im Bild. Convolutional Layer sind jedoch nicht ausschließlich auf die Verarbeitung von Bildern begrenzt. Es gibt auch Implementierungen für eindimensionale Daten, wie beispielsweise Zeitreihendaten oder elektrische Signale.

3 Methodik

Dieses Kapitel stellt die theoretische Grundlage für die in dieser Arbeit angewendeten Technologien und Methoden bereit. Ein besonderes Augenmerk liegt dabei auf der Generierung und Auswahl der Features, die für die weitere Analyse und Modellbildung unerlässlich sind. Es werden sowohl die Feature-Generierung, unter Anwendung der Fourier-Transformation, als auch verschiedene Korrelations- und Klassifizierungsverfahren erörtert.

3.1 Datensatz

Die Daten für dieses Projekt stammen aus einer nicht öffentlich zugänglichen Sammlung von Patientenproben. Der Datensatz ist strukturiert und in verschiedene Läufe (sogenannte „Runs“) aufgeteilt. Jeder Lauf ist mit einem Barcode gekennzeichnet, der wichtige Metainformationen über die sequenzierte Probe enthält.

Zu beachten ist, dass der Datensatz sowohl „PASS“-Runs als auch „FAIL“-Runs enthält. Jeder Run besteht aus mehreren Fast5-Dateien, die die Rohdaten der Sequenzierungen und zusätzliche Metadaten enthalten.

Ein besonderes Merkmal dieses Datensatzes ist, dass die meisten Proben mit CT-Werten gekennzeichnet sind. Dies ermöglicht es, jeder Fast5-Datei und somit jedem Sequenzierungs-Read einen spezifischen CT-Wert zuzuordnen. Für die Ziele dieses Projekts sind Daten ohne zugewiesenen CT-Wert nicht von Interesse.

Insgesamt umfasst der Datensatz 7119 Fast5-Dateien. Allerdings lässt sich nur für 3705 dieser Dateien ein CT-Wert zuordnen. Diese setzen sich zusammen aus 3297 „PASS“- und 408 „FAIL“-Dateien. Alle Dateien zusammen beinhalten 14.158.977 Reads. Zusammenfassend existieren im kompletten Datensatz 66 verschiedene diskrete CT-Werte. Diese liegen in einem Bereich zwischen 12 und 37.

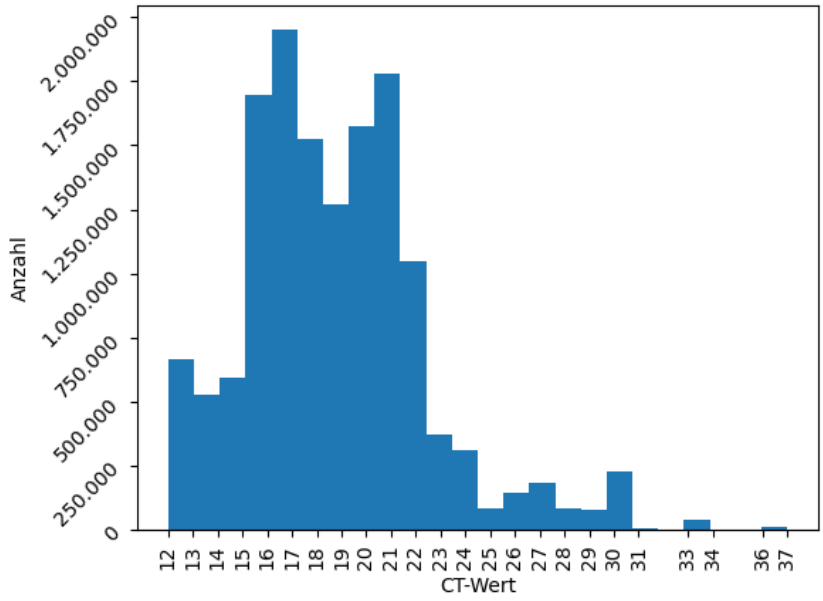


Abbildung 3.1: Verteilung der Reads auf CT-Werte

3.2 Reduzierung des Datensatzes

Aufgrund der Größe des Datensatzes ist es sinnvoll, den Umfang des Datensatzes zu reduzieren und lediglich einen repräsentativen Ausschnitt der Daten zu verwenden. Diese Herangehensweise verringert nicht nur die Komplexität des Projekts, sondern minimiert auch die benötigte Rechenzeit und den Ressourcenbedarf.

Darüber hinaus wird der Datensatz zur einfacheren Ableitung des Verdünnungsgrades in zwei Kategorien unterteilt: starke Verdünnung und geringe Verdünnung. Hierzu werden zwei Bins erstellt: Ein Bin enthält Reads mit starken Verdünnungen (CT-Werte von 0 bis 14), und der andere Bin enthält Reads mit geringen Verdünnungen (CT-Werte von 21 bis 37).

Die bewusste Entscheidung, eine Lücke zwischen den beiden Bins zu belassen, soll dabei helfen, eine klarere Unterscheidung zwischen den beiden Kategorien zu schaffen. Dies könnte es maschinellen Lernalgorithmen ermöglichen, eine präzisere Kategorisierung vorzunehmen.

Anschließend wird die erforderliche Stichprobengröße berechnet. Hierfür wird die Standardformel für den Stichprobenumfang verwendet:

$$n_{unendlich} = \frac{Z^2 \cdot P \cdot (1 - P)}{E^2} \tag{3.1}$$

$$n_{endlich} = \frac{N \cdot n_{unendlich}}{N + n_{unendlich}} \tag{3.2}$$

- n die Stichprobengröße
- N die Gesamtgröße des Datensatzes (14.158.977)
- Z der Z-Wert (z.B. 2,58 für 99 % Konfidenzniveau)
- P die erwartete Proportion (wenn unbekannt, wird 0,5 verwendet, um die maximale Stichprobengröße zu erhalten)
- E die Toleranz oder der Fehler (0,01 für ±1 %)

Die Anwendung dieser Formel mit den angegebenen Werten ergibt eine erforderliche Stichprobengröße von $\sim$ 16.622.

Jedem Read im Datensatz wird basierend auf seinem CT-Wert ein Bin zugewiesen. Um eine ausgeglichene Vertretung beider Kategorien im Datensatz zu gewährleisten, wird eine gleichmäßige Anzahl von Reads pro Bin in den Unterdatensatz aufgenommen. Der für die weiteren Analysen verwendete Unterdatensatz umfasst 25.000 Reads pro Bin und ist somit größer als die statistisch notwendige Stichprobengröße. Dieser reduzierte Datensatz behält die repräsentativen Eigenschaften des ursprünglichen Datensatzes bei und ist zugleich effizienter zu verarbeiten und handhaben.

3.3 Feature Engineering

Im folgenden Schritt des Projekts werden aus den Rohdaten der Reads des reduzierten Datensatzes Features extrahiert, die auch als Features bezeichnet werden. Diese Features entspringen diversen Bereichen wie Mathematik, Statistik und Signalanalyse.

3.3.1 Statistische Features

- Das Minimum des Signals gibt den kleinsten Wert im Signal an: signal_min = min(signal)

- Das Maximum des Signals gibt den größten Wert im Signal an: signal_max = max(signal)
- Der Durchschnittswert des Signals ist die Summe aller Werte im Signal geteilt durch die Anzahl der Werte: signal_mean = $\frac{1}{N} \cdot \sum_{i=1}^{N} \text{signal}[i]$
- Der Median des Signals ist der mittlere Wert, der das Signal in zwei Hälften teilt, wobei die eine Hälfte der Werte kleiner und die andere Hälfte der Werte größer als der Median ist: signal_median = median(signal)
- Die Standardabweichung des Signals gibt an, wie weit die Werte im Signal um den Durchschnitt herum streuen: signal_std = $\sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (\text{signal}[i] - \text{signal\_mean})^2}$
- Die Varianz des Signals misst die durchschnittliche quadratische Abweichung der Werte im Signal vom Durchschnitt: signal_var = $\frac{1}{N} \cdot \sum_{i=1}^{N} (\text{signal}[i] - \text{signal\_mean})^2$
- Der durchschnittliche absolute Unterschied (Average Absolute Difference, AAD) zwischen den Werten des Signals gibt an, wie weit die Werte im Signal im Durchschnitt voneinander entfernt sind: signal_aad = $\frac{1}{N} \cdot \sum_{i=1}^{N} |\text{signal}[i] - \text{signal\_mean}|$
- Die Differenz zwischen dem Maximum und dem Minimum des Signals gibt den Spannungsbereich des Signals an: signal_maxmin_diff = signal_max − signal_min
- Die mittlere absolute Abweichung (Median Absolute Deviation, MAD) des Signals gibt an, wie weit die Werte im Signal im Durchschnitt vom Median abweichen: signal_mad = median $(|\text{signal}[i] - \text{signal\_median}|)$
- Der Interquartilsabstand des Signals ist die Differenz zwischen dem 75. Perzentil und dem 25. Perzentil und gibt an, wie breit die mittlere Hälfte der Werte im Signal ist: signal_iqr = Q3 − Q1, wobei Q1 das 25. Perzentil und Q3 das 75. Perzentil sind.

3.3.2 Features zur Amplitudenanalyse:

- Die Anzahl der negativen Werte im Signal gibt an, wie viele Werte im Signal negativ sind: signal_neg_count = $\sum_{i=1}^{N} \text{signal}[i] < 0$
- Die Anzahl der positiven Werte im Signal gibt an, wie viele Werte im Signal positiv sind: signal_pos_count = $\sum_{i=1}^{N} \text{s}[i] > 0$
- Die Anzahl der Werte im Signal, die über dem Durchschnitt liegen, gibt an, wie viele Werte im Signal über dem Durchschnittswert liegen: signal_above_mean = $\sum_{i=1}^{N} \text{s}[i] >$ signal_mean

- Die Anzahl der Spitzen im Signal gibt an, wie viele lokale Maxima oder Minima im Signal vorhanden sind.

3.3.3 Verteilungsbasierte Features

- Die Schiefe der Verteilung des Signals gibt an, ob die Verteilung asymmetrisch ist und in welche Richtung sie abweicht.
- Die Kurtosis (Wölbung) der Verteilung des Signals gibt an, wie stark die Verteilung im Vergleich zur Normalverteilung abgeflacht oder spitzer ist.

3.3.4 Energie-Features

- Die Energie des Signals gibt an, wie viel Energie insgesamt im Signal enthalten ist. Sie kann als Summe der quadrierten Amplitudenwerte berechnet werden: signal_energy $= \frac{1}{100} \cdot \sum_{i=1}^{N} \mathrm{s}[i]^2$
- Die Signal-Root-Mean-Square (RMS) Energie ist ein Maß für die effektive Energie des Signals: signal_rms $= \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} \mathrm{s}[i]^2}$

3.3.5 Fourier-Transformation

Die Fourier-Transformation bildet ein zentrales Werkzeug in der Signalanalyse. Mit ihrer Hilfe kann ein Signal, das in der Zeitdomäne repräsentiert ist, in die Frequenzdomäne überführt werden. Dies ermöglicht detaillierte Untersuchungen, indem die verschiedenen Frequenzen bzw. Schwingungen aufgedeckt werden, die das ursprüngliche Signal formen. Dabei zerlegt die Fourier-Transformation das Signal in seine Frequenzkomponenten.

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(\tau) \cdot e^{-i \cdot \omega \cdot \tau} dt \tag{3.3}$$

Die Fourier-Transformation, wie sie ursprünglich konzipiert wurde, funktioniert mit kontinuierlichen Signalen, welche in der Praxis allerdings selten vorkommen.

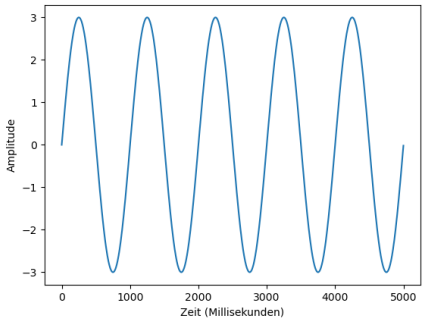
Diskrete Fourier-Transformation

In der realen Welt sind die meisten Signale diskret, d.h., sie bestehen aus einer Folge von diskreten Punkten oder Messwerten. Für solche Signale wird die Diskrete Fourier-

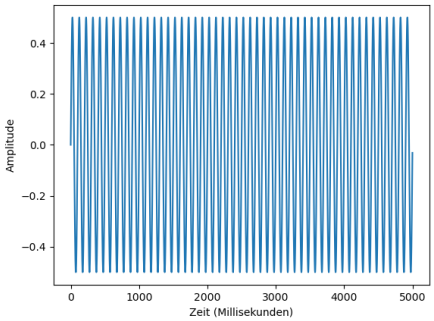
Transformation (DFT) verwendet. Sie erweitert das Konzept der Fourier-Transformation auf diskrete, also in der Praxis messbare, Signale.

Fast Fourier-Transformation

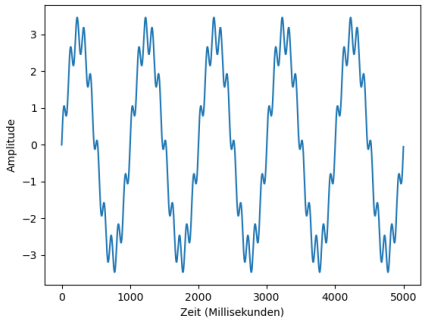
Die Diskrete Fourier-Transformation (DFT) ist für die Zerlegung von Signalen in elementare Frequenzbestandteile unerlässlich. Die Fast Fourier-Transformation (FFT), eine effizientere DFT-Variante, ist jedoch bei umfangreichen Datensätzen von Vorteil. Sie ermöglicht die Generierung von auf Frequenzdomänen basierenden Features, welche für weiterführende Prozesse wie maschinelles Lernen und Datenanalyse von großer Bedeutung sind.



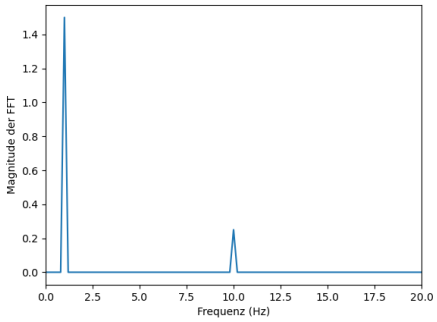
(a) $\sin(2\pi \cdot \frac{1}{1000})$



(b) $0.5 \cdot \sin(2\pi \cdot \frac{10}{1000})$



(c) Signal (a) + Signal (b)



(d) Ausgabe der Frequenzkomponenten der FFT

Abbildung 3.2: Beispiel einer Fourier Transformation

Die Resultate einer FFT sind komplexe Zahlen, die sich in eine Magnitude und eine Phase unterteilen lassen.

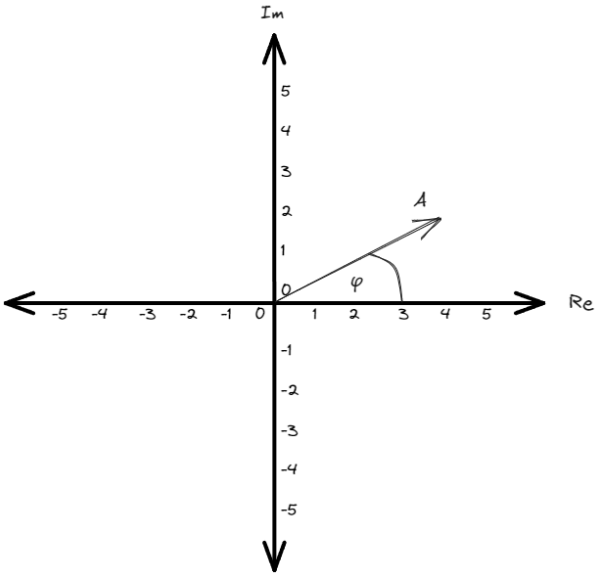


Abbildung 3.3: Skizze einer Komplexen Zahl

Magnitude/Amplitude: Die Magnitude (A) oder Amplitude eines Signals repräsentiert die Intensität oder Stärke der jeweiligen Frequenzkomponente im Signal. Sie stellt den Beitrag der jeweiligen Frequenzkomponente zur Gesamtheit des Signals dar. Sie kann berechnet werden durch:

$$A = \sqrt{Re^2 + Im^2} \tag{3.4}$$

Phase: Die Phase (φ) eines Signals veranschaulicht den zeitlichen Versatz einer spezifischen Frequenzkomponente, wodurch sie angibt, an welchem Punkt im Zyklus die Welle beginnt. Sie gibt eine relative Verschiebung eines Signals im Vergleich zu einem Referenzsignal an. Sie kann berechnet werden durch:

$$\varphi = \arctan\left(\frac{Im}{Re}\right) \tag{3.5}$$

Hierbei sind Re und Im der Realteil und der Imaginärteil der komplexen Zahl, die das Resultat der FFT darstellt.

Durch eine weiterführende Analyse von Phase und Magnitude lassen sich zusätzliche statistische Features ableiten. Im Kontext des vorliegenden Projekts werden die FFT-

Ergebnisse zur Generierung von Features genutzt, die die Verteilung, Varianz, Kurtosis und Skewness der Frequenzkomponenten beschreiben.

3.3.6 Automatische Feature-Generierung mit Autoencodern

Mit neuronalen Netzwerken ist es möglich, Daten mithilfe sogenannter Autoencoder zu kodieren und eine Repräsentation zu erzeugen, die eine geringere Dimensionalität aufweist als die Ausgangsdaten. Dabei kommt das Konzept der Autoencoder zur Anwendung. Diese kodieren die Daten zunächst in eine ausgewählte Dimension und versuchen dann, die Ausgangsdaten aus den kodierten Daten wiederherzustellen. Sowohl die originalen als auch die rekonstruierten Daten werden hierfür herangezogen. Für das Training wird die Mean-Squared-Error Loss Funktion verwendet.

Die Dimension des latenten Raums (der kodierten Repräsentation) bestimmt die Anzahl der Features, die der Autoencoder aus den Daten extrahiert. Eine geringere Dimension führt zu weniger, aber möglicherweise informativeren Features, während eine größere Dimension zu mehr Features führen kann, aber auch das Risiko eines Overfittings erhöht.

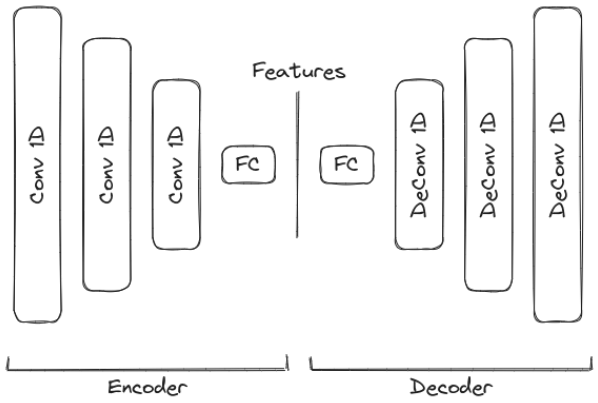


Abbildung 3.4: Netzwerkarchitektur des Autoencoders

Der in diesem Projekt eingesetzte Autoencoder besteht im Encoder-Teil aus jeweils drei aufeinanderfolgenden Conv1D-Layern, jeweils mit einer Kernel-Größe von 3 und 32 Kernels. Lediglich der letzte Layer verwendet nur einen Kernel. Darauf folgt ein Fully-Connected-Layer, der die Daten auf die jeweils ausgewählte Anzahl an Features reduziert. Mit einem Stride von 2 und einem Padding von 1 wird die Länge der Daten nach jedem Conv-Layer halbiert, sodass der Fully-Connected-Layer bei einer Länge der Eingangsdaten von 8192 nur noch 1024 Eingangsneuronen besitzt.

Der Decoder arbeitet in umgekehrter Weise. Zuerst erhöht ein Fully-Connected-Layer

die Dimensionalität auf 1024, gefolgt von drei aufeinanderfolgenden TransposedConv1D-Layern. Mit einem Stride von 2, einem Padding von 1 und einem Output-Padding von 1 skalieren diese die Daten wieder auf ihre ursprüngliche Größe hoch. Die Transposed Convolution ist dabei eine spezielle Art von Convolution, die die Länge der Daten verdoppeln kann. Allen Layern im Encoder und Decoder folgt die Aktivierungsfunktion LeakyReLU(negative_slope=0.01), lediglich die allerletzte Aktivierungsfunktion ist eine Sigmoid-Funktion.

Da neuronale Netzwerke eine einheitliche Länge der Daten voraussetzen, werden kürzere Signale bis zu einer Länge von 8192 mit Nullen aufgefüllt und längere Signale nach 8192 Werten abgeschnitten. Außerdem werden die Signale über die Funktion $f(x) = \frac{x-min(x)}{max(x)-min(x)}$ auf Werte zwischen 0 und 1 normalisiert, was dem Wertebereich der Sigmoid-Funktion entspricht, die als letzte Aktivierungsfunktion beim Autoencoder eingesetzt wird.

3.4 Korrelationsfindung

Im Zuge der Feature-Extraktion und -Selektion ist die Identifizierung von Zusammenhängen zwischen den einzelnen Features und dem Zielattribut, hier der CT-Wert, von entscheidender Bedeutung. Die Ermittlung solcher Korrelationen ermöglicht die Identifikation von Features, die eine signifikante Beziehung zum Zielattribut aufweisen. Hierfür dient die Erstellung einer Korrelationsmatrix. Eine Korrelationsmatrix ist eine quadratische Matrix, die die Korrelationskoeffizienten zwischen jedem Paar von Features darstellt. Diese Koeffizienten, die Werte zwischen -1 und 1 annehmen können, messen die statistische Beziehung zwischen den Features. Ein Wert nahe 1 bedeutet eine starke positive Korrelation, während ein Wert nahe -1 auf eine starke negative Korrelation hinweist. Ein Wert nahe 0 bedeutet, dass keine lineare Korrelation besteht.

3.4.1 Nach Pearson

Die Pearson-Korrelation ist eine weit verbreitete Methode zur Messung der Korrelation zwischen Variablen. Sie ermittelt den linearen Zusammenhang zwischen zwei Datensätzen. Die Pearson-Korrelation kann insbesondere zur Überprüfung verwendet werden, ob und in welchem Maße die extrahierten Features mit dem CT-Wert korrelieren.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.6}$$

- r_{xy}: Pearson-Korrelationskoeffizient zwischen X und Y
- n: Anzahl der Beobachtungen
- x_i und y_i: Einzelbeobachtungen von X und Y
- $\bar{x}$ und $\bar{y}$: Durchschnitt der Beobachtungen von X bzw. Y

3.5 Klassifizierungsverfahren

Im abschließenden Schritt der explorativen Analyse erfolgt die Modellierung. Hierbei wird ein Modell erstellt, dessen Ziel es ist, die erstellten CT-Wert-Bins auf Basis der generierten Features vorherzusagen. Verschiedene Klassifizierungsverfahren eignen sich für diese Aufgabe. Im Folgenden werden zwei weit verbreitete Verfahren - Entscheidungsbäume und Random Forest - detaillierter vorgestellt.

3.5.1 Entscheidungsbäume

Entscheidungsbäume sind eine Form von überwachten Lernalgorithmen, die sich durch ihre einfache Interpretierbarkeit und Visualisierung auszeichnen. Sie bilden eine Baumstruktur, in der jeder innere Knoten eine Entscheidungsregel repräsentiert, die auf einer oder mehreren Eingabevariablen basiert. Jeder Zweig repräsentiert das Ergebnis dieser Regel und jeder Endknoten (oder Blatt) repräsentiert eine Klassifikationsentscheidung. Die Entscheidungen von der Wurzel bis zu einem Blatt stellen einen Pfad dar, der zu einer bestimmten Klassifikation führt.

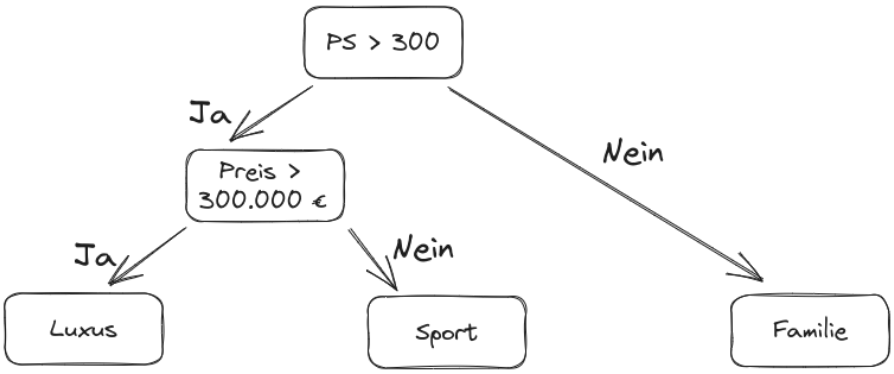


Abbildung 3.5: Beispiel eines Entscheidungsbaumes zur Klassifizierung in eines PKW-Typen

Ein entscheidender Vorteil von Entscheidungsbäumen ist ihre intuitive Natur - die Entscheidungen können einfach visualisiert und verstanden werden. Sie können jedoch anfällig für Overfitting sein, besonders wenn der Baum zu tief ist, und sie können instabil sein, da kleine Änderungen in den Daten zu großen Änderungen im Baum führen können.

3.5.2 Random Forest

Random Forest stellt eine Erweiterung des Entscheidungsbäume-Verfahrens dar. Im Wesentlichen handelt es sich um ein Ensemble-Lernverfahren, das eine Menge von Entscheidungsbäumen (den „Wald“) erzeugt und ihre Vorhersagen kombiniert. Jeder Baum im Random Forest wird unabhängig von den anderen Bäumen erzeugt, wobei ein zufälliges Subset der Trainingsdaten und ein zufälliges Subset der Features für jede Teilung ausgewählt wird.

Die endgültige Klassifikationsentscheidung wird dann durch Abstimmung aller Bäume des Waldes getroffen, wobei die häufigste Klassifikation ausgewählt wird. Der Random Forest-Algorithmus kann das Overfitting-Problem, das bei einzelnen Entscheidungsbäumen auftreten kann, reduzieren und ist oft genauer. Er ist jedoch rechenintensiver und die Ergebnisse sind schwieriger zu interpretieren als die eines einzelnen Entscheidungsbauems. Des Weiteren ist es mit dem Random-Forest verfahren möglich eine Regression zu erstellen.

3.5.3 Cross-Validation

Cross-Validation ist eine statistische Methode zur Evaluierung der Leistungsfähigkeit von Modellen. Sie dient dazu, die Generalisierbarkeit eines Modells auf unbekannte Daten zu schätzen und Overfitting zu vermeiden. Bei Overfitting passt sich das Modell zu stark an die Trainingsdaten an und versagt bei der Generalisierung auf neue, unbekannte Daten.

In einer typischen Cross-Validation wird das vorhandene Datenset in k Teilmengen, sogenannte „Folds“, aufgeteilt. Anschließend wird das Modell k Mal trainiert und validiert. In jedem Durchlauf dient ein Fold als Testdatensatz, während die restlichen $k - 1$ Folds als Trainingsdatensatz verwendet werden. Nach Abschluss aller k Durchläufe werden die Evaluationsmetriken gemittelt, um eine robuste Schätzung der Modellleistung zu erhalten.

Ein häufig verwendeter Ansatz ist die k-Fold-Cross-Validation, wobei k oft auf Werte wie 5 oder 10 gesetzt wird.

Cross-Validation bietet den Vorteil einer besseren Generalisierbarkeit im Vergleich zu einfachen Train-Test-Splits, da das Modell an verschiedenen Teilmengen des Datensatzes

getestet wird. Dies trägt zu einer robusteren Modellbewertung bei und kann helfen, ein besseres Verständnis für die Stärken und Schwächen des Modells zu entwickeln.

3.5.4 Metriken für Klassifizierungsverfahren

Für die Evaluierung des trainierten Random-Forest-Klassifikators werden verschiedene Metriken herangezogen, die im Folgenden näher erläutert werden:

Accuracy (Genauigkeit)

Die Metrik „Accuracy“ gibt den Anteil der korrekt klassifizierten Beispiele im Verhältnis zu allen Beispielen an. Sie ist eine der am häufigsten verwendeten Metriken. Allerdings ist sie nicht immer aussagekräftig, insbesondere bei unausgewogenen Datensätzen.

$$\text{Accuracy} = \frac{\text{Anzahl der korrekten Vorhersagen}}{\text{Gesamtanzahl der Beobachtungen}} \tag{3.7}$$

ROC-AUC (Receiver Operating Characteristic - Area Under Curve)

Die Metrik „ROC-AUC“ bewertet die Fähigkeit des Modells, zwischen den Klassen zu unterscheiden. Der Wertebereich liegt zwischen 0 und 1. Ein Wert von 0,5 deutet auf ein Zufallsmodell hin, während ein Wert von 1 eine perfekte Unterscheidungsfähigkeit anzeigt.

Precision (Genauigkeit)

Die Metrik „Precision“ gibt den Anteil der korrekt positiv klassifizierten Beispiele im Verhältnis zu allen als positiv klassifizierten Beispielen an.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.8}$$

Recall (Sensitivität)

Die Metrik „Recall“ gibt den Anteil der korrekt positiv klassifizierten Beispiele im Verhältnis zu allen tatsächlich positiven Beispielen an.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.9}$$

F1-Score

Der „F1-Score“ ist das harmonische Mittel aus Precision und Recall und dient als Kompromiss zwischen den beiden. Diese Metrik wird insbesondere in Szenarien verwendet, in denen beide Metriken gleichermaßen wichtig sind.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.10}$$

Die Verwendung mehrerer dieser Metriken ermöglicht eine umfassende Beurteilung der Leistungsfähigkeit eines Klassifikationsmodells, da jede Metrik ihre eigenen Stärken und Schwächen hat.

3.5.5 Metriken für Regressionsverfahren

Zur Evaluierung von trainierten Regressionsmodellen, wie beispielsweise dem Random-Forest-Regressor, werden unterschiedliche Metriken verwendet, die im Folgenden detailliert erläutert werden.

Mittlerer absoluter Fehler (Mean Absolute Error, MAE)

Der mittlere absolute Fehler gibt den Durchschnitt der absoluten Differenzen zwischen den vorhergesagten und den tatsächlichen Werten an. Diese Metrik ist einfach zu interpretieren und reicht von 0 bis unendlich, wobei ein Wert von 0 ein perfektes Modell anzeigt.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_{\text{true},i} - y_{\text{pred},i}| \tag{3.11}$$

Mittlerer quadratischer Fehler (Mean Squared Error, MSE)

Der mittlere quadratische Fehler ist eine weitere häufig verwendete Metrik, die die durchschnittliche Quadratdifferenz zwischen den beobachteten tatsächlichen Ausgangswerten und den Werten, die vom Modell vorhergesagt werden, misst.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_{\text{true},i} - y_{\text{pred},i})^2 \tag{3.12}$$

R-Quadrat (R^2)

Der Determinationskoeffizient, oft als R^2 bezeichnet, gibt an, welcher Anteil der Varianz in den abhängigen Variablen vorhergesagt werden kann, und reicht von 0 bis 1. Je höher R-Quadrat, desto besser ist das Modell an die Trainingsdaten angepasst.

$$R^2 = 1 - \frac{\text{Summe der quadratischen Fehler}}{\text{Totale Summe der Quadrate}} \tag{3.13}$$

Explained Variance Score

Der Explained Variance Score misst den Anteil der gesamten Varianz, der durch das Modell erklärt wird. Ein Wert von 1 würde bedeuten, dass das Modell die abhängige Variable perfekt vorhersagt, während ein Wert von 0 darauf hindeutet, dass das Modell nicht besser als ein einfaches Mittelwertmodell ist.

$$\text{Explained Variance Score} = 1 - \frac{\text{Varianz der Fehler}}{\text{Varianz der tatsächlichen Werte}} \tag{3.14}$$

4 Durchführung

In diesem Kapitel wird der Fokus auf die praktische Durchführung der vorgesehenen Experimente gelegt. Es wird eine Darstellung der technischen Umsetzung sowie der angewandten Methoden und Verfahren gegeben, die für die Umsetzung der Experimente von Bedeutung waren.

4.1 Technische Implementierung

Die technische Umsetzung der Experimente erfolgte mithilfe der Programmiersprache Python. Alle zugehörigen Dateien sind im folgenden GitHub-Repository verfügbar:

`https://github.com/christophstach/covid-springs-explorative-analysis`

Für die Erstellung der Diagramme und Visualisierungen wurde die Matplotlib-Bibliothek[1] verwendet, um eine anschauliche Darstellung zu gewährleisten. Weitere skizzenhafte Darstellungen wurden über die Software Excalidraw[2] realisiert. Die Pandas-Bibliothek[3] kam beim Erzeugen von Dataframes zum Einsatz, um die Handhabung der Features zu erleichtern. Zudem bietet Pandas Funktionen zur Berechnung von Korrelationen (z.B. Pearson) und zur Gruppierung von Daten.

Die Zwischenergebnisse sowie Teildatensätze wurden im leistungsstarken Feather-Dateiformat[4] gespeichert, das nahtlos mit Pandas interagiert und somit die Effizienz steigert.

Die Berechnung der Features erfolgte mithilfe von NumPy[5]. Zusätzliche Features wurden, wie bereits erwähnt, mittels Autoencodern generiert, die mithilfe der Bibliotheken PyTorch[6] und PyTorch-Lightning[7] trainiert wurden.

[1] Matplotlib: `https://matplotlib.org/`
[2] Excalidraw: `https://excalidraw.com/`
[3] Pandas: `https://pandas.pydata.org/`
[4] Apache Arrow (Feather): `https://arrow.apache.org/`
[5] NumPy: `https://numpy.org/`
[6] PyTorch: `https://pytorch.org/`
[7] PyTorch-Lightning: `https://www.pytorchlightning.ai/`

Abschließend wurde SkLearn[8] für das Training der Klassifikationsmodelle in Form von Entscheidungsbäumen eingesetzt.

4.2 Versuchsaufbau

Die durchgeführten Versuche lassen sich in zwei Kategorien unterteilen, die im Folgenden kurz erläutert werden und erfordern bestimmte Voraussetzungen, bevor sie durchgeführt werden können. Zunächst müssen die Features festgelegt werden, die für die jeweiligen Versuche berechnet werden sollen. In der Datei *features.py* im Hauptordner des Repositories sind alle Features definiert, die in den späteren Experimenten Verwendung finden.

Darüber hinaus wurden fünf Autoencoder trainiert. Jeder dieser Autoencoder wurde über zehn Epochen hinweg mit dem vollständigen Datensatz trainiert. Die Anzahl der durch die Autoencoder erzeugten Features beträgt 2, 3, 8, 16 und 32. Die gespeicherten Parameter (Gewichte der Autoencoder) befinden sich im Ordner *checkpoints/*.

4.3 Vorbereitung

Um die Experimente vorzubereiten, sind einige Schritte notwendig. Zunächst müssen bestimmte Vorbereitungen getroffen werden, darunter die Erstellung von Zwischendateien. Mithilfe der Datei *create_file_index.py* lässt sich ein Datei-Index des Datensatzes erstellen. Dieser Index enthält alle fast5-Dateien des Projekts mit den zugeordneten CT-Werten sowie der Anzahl der Reads pro Datei und einer Kennzeichnung, ob die Datei als „passed“ markiert ist. Der vollständige Datei-Index umfasst 3.705 Einträge.

Auf Grundlage dieses Datei-Index kann über die Datei *create_read_index.py* ein Read-Index erstellt werden. Dieser enthält sämtliche Reads innerhalb der im Datei-Index aufgeführten Dateien. Dabei werden der Pfad zur Datei, die Read-ID, die Länge des Reads, der „passed“-Status und der CT-Wert gespeichert. Der gesamte Read-Index umfasst 14.158.977 Einträge.

4.3.1 Training der Autoencoder

Als letzter vorbereitender Schritt müssen die für die späteren Experimente verfügbaren Autoencoder trainiert werden. Dies erfolgt über die Datei *train_read_autoencoder.py*

[8] SkLearn: `https://scikit-learn.org/`

mithilfe der Bibliothek PyTorch Lightning über einen Trainingsverlauf von 10 Epochen mit einer Batch-Size von 64. Hierbei werden sämtliche Reads des gesamten Datensatzes verwendet und in einem Verhältnis von 80 zu 20 in Trainings- und Validierungsdatensätze aufgeteilt. Die Lernrate wird automatisch durch PyTorch-Lightnings *LearningRateFinder()* bestimmt.

4.4 Analyse der Reads

Bei diesen Versuchen liegt der Fokus auf der Analyse der Reads. Hierfür ist zunächst die Erstellung eines Unterdatensatzes erforderlich, da die Analyse lediglich auf einer repräsentativen Probe aller Reads basieren soll. Die Datei *create_binned_subset.py* generiert auf Grundlage des Read-Index einen solchen Unterdatensatz. Dabei können Bins definiert werden, die auf CT-Werten basieren und dem Unterdatensatz hinzugefügt werden. Zur Vereinfachung werden lediglich zwei Bins verwendet: einer für niedrige CT-Werte (0-16) und einer für hohe CT-Werte (21-37). Hierbei wird bewusst eine Lücke gelassen, um die Bestimmung für spätere Experimente und Klassifikationsverfahren zu erleichtern. Jeder Bin enthält 25.000 Reads, sodass die erstellte Datei insgesamt 50.000 Einträge ihrer Signaldaten (Raw Values) umfasst.

Die Datei *create_features.py* wandelt im Anschluss die Rohsignale der Reads aus dem Unterdatensatz in die zuvor definierten Features um. Dabei fließen auch die Features der Autoencoder mit ein. Die Feature-Extraktion umfasst 18 Features, die aus den Rohsignalen abgeleitet werden, 4 Features aus der Phase des FFT-transformierten Rohsignals, weitere 4 Features aus der Magnitude des FFT-transformierten Rohsignals und 61 Features aus verschiedenen Autoencodern. Zusätzlich wird die Länge des Reads als weiteres Feature betrachtet. Insgesamt ergibt sich eine Gesamtmenge von 88 Features.

4.5 Analyse der Dateien

Ein weiterer Ansatz besteht darin, die Beziehung zwischen Gruppen von Reads und dem CT-Wert zu untersuchen. Hierfür erfolgt die Analyse der individuellen fast5-Dateien, da eine solche Datei eine Gruppe von meist 4000 Reads repräsentiert (was die Standard-Einstellung des Sequenzierers ist; einige Dateien haben weniger Reads). Die Datei *create_file_features.py* berechnet dabei alle bereits zuvor erwähnten Features für jeden Read in jeder Datei. Anschließend werden für jedes dieser Features vier statistische Kennzahlen berechnet, was insgesamt zu 352 Features pro fast5-Datei führt.

5 Experimente und Ergebnisse

In diesem Kapitel werden die durchgeführten Experimente und die erzielten Ergebnisse präsentiert. Der Überblick reicht von der Präsentation der Trainingsergebnisse der Autoencoder über die reine Analyse der zuvor definierten Features bis hin zum Training von Random-Forest-Klassifizierungsverfahren zur Bestimmung des CT-Werts innerhalb einzelner Reads sowie ganzer Dateien mit einer Vielzahl von Reads.

5.1 Trainingsergebnisse der Autoencoder

Der erste Schritt beinhaltet einen schnellen Blick auf die Trainingsergebnisse der Autoencoder. Diese wurden im Vorfeld trainiert und dienen als Grundlage für die automatische Feature-Extraktion, die in den weiterführenden Experimenten benötigt wird.

Während des Trainings ergaben sich die folgenden Verläufe für die unterschiedlichen Loss-Werte der Autoencoder **2**, **3**, **8**, **16** und **32**:

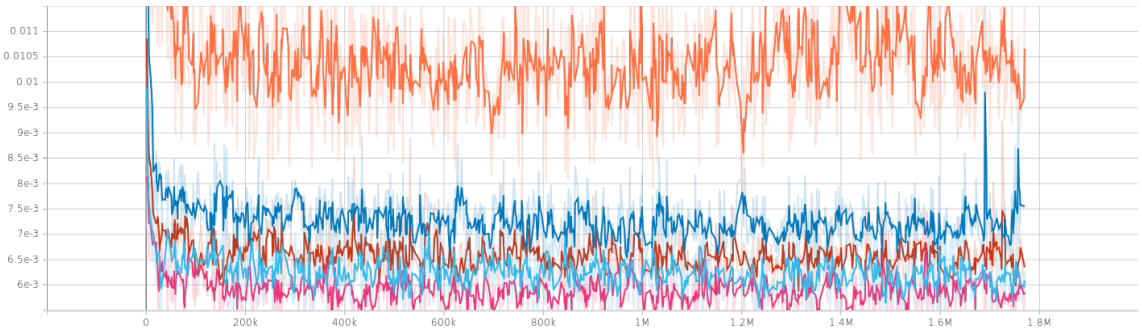


Abbildung 5.1: Trainings-Loss-Kurve der Autoencoder

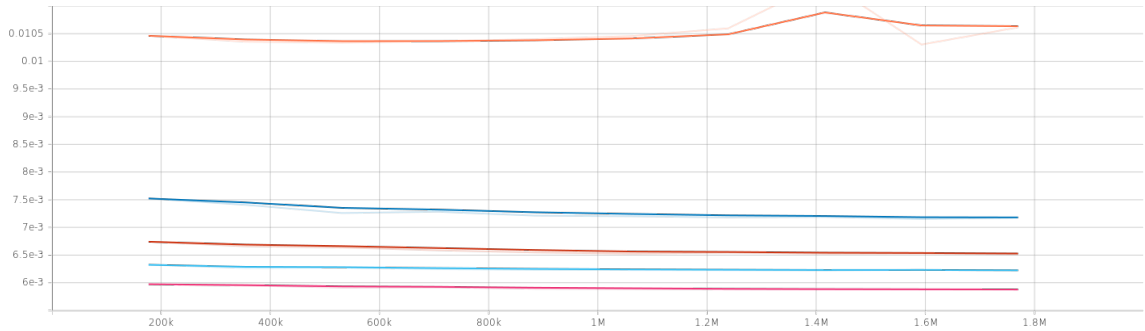
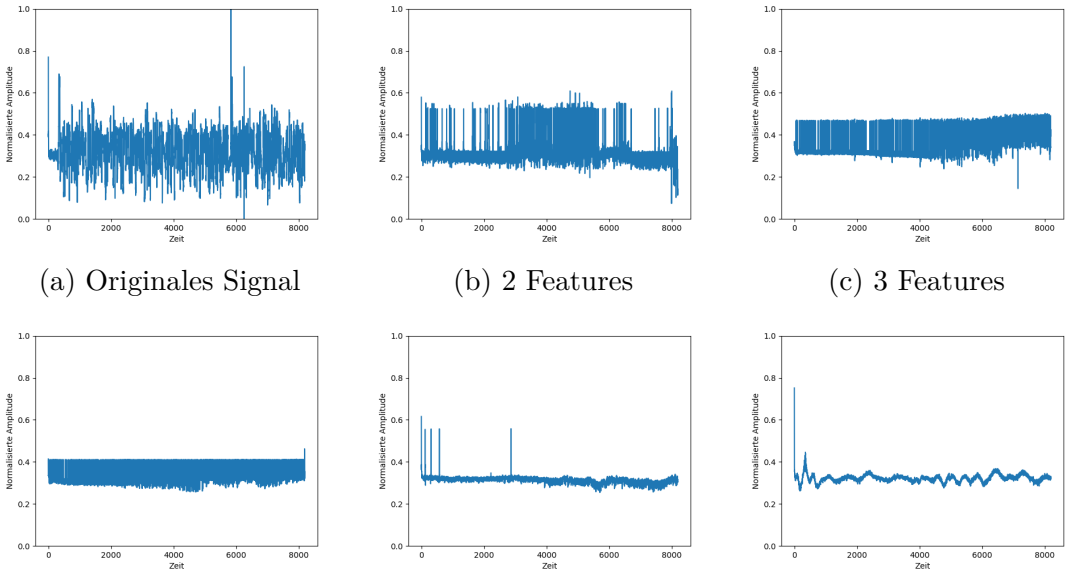


Abbildung 5.2: Validation-Loss-Kurve der Autoencoder

Auf der Y-Achse ist der Fehler (Loss) dargestellt, während die X-Achse die Anzahl der Iterationen repräsentiert. Die Kurven verdeutlichen, dass die gewählte Anzahl von 10 Epochen mehr als ausreichend war. Dies lässt sich daran erkennen, dass bei allen Autoencodern bereits nach kurzer Zeit eine Konvergenz erreicht und der Loss-Wert ein Plateau gefunden hat.

Zudem ist auffällig, dass Autoencoder mit einer größeren Anzahl an Features im Bottleneck besser darin sind, das Ausgangssignal zu rekonstruieren, was sich in einem niedrigeren Loss niederschlägt. Autoencoder mit einer geringeren Anzahl an Features müssen hingegen wichtige Charakteristika stärker extrahieren, um eine adäquate Signalrekonstruktion zu gewährleisten. Diese könnten daher von besonderem Interesse sein.



(a) Originales Signal (b) 2 Features (c) 3 Features

(d) 8 Features (e) 16 Features (f) 32 Features

Abbildung 5.3: Rekonstruktionen der normalisierten Rohsignale der Autoencoder

In den oberen Grafiken ist erkennbar, wie die verschiedenen Autoencoder ein bestehendes Signal zunächst encodieren und anschließend dekodieren, um es somit zu rekonstruieren. Dabei fällt interessanterweise der Autoencoder mit 32 Features besonders auf. Es scheint, als würde er das vorhandene Signal stark glätten.

5.2 Analyse auf Read-Ebene

Eine interessante Fragestellung besteht darin, ob es möglich ist, Reads (und somit die erstellten Features auf der Ebene der Reads) in Beziehung zum CT-Wert zu setzen. Hierzu wird im ersten Schritt versucht, eine Korrelationsmatrix der Features zu erstellen und anschließend einen RandomForest-Klassifizierer zu trainieren. Die Datengrundlage bilden dabei die mit *create_features.py* erstellten Features des Unterdatensatzes (der mit *create_binned_subset.py* erstellt wurde).

5.2.1 Korrelation zwischen Read-Features und CT-Wert

Die folgende Tabelle präsentiert die Korrelationskoeffizienten der einzelnen Features zum CT-Wert. Zur Veranschaulichung der stärksten Korrelationen ist die Tabelle absteigend nach den absoluten Koeffizienten sortiert. Die Darstellung ist auf die ersten 15 Werte limitiert; die vollständige Tabelle ist im zugehörigen Git-Repository abrufbar.

Feature	Korrelationskoeffizient
raw_neg_count	0,118
raw_min	−0,095
raw_maxmin_diff	0,094
raw_max	0,088
autoencoder_16d_12	0,075
fft_magnitude_kurtosis	0,073
raw_kurtosis	0,073
fft_phase_kurtosis	0,073
autoencoder_32d_3	0,069
autoencoder_16d_3	−0,066
autoencoder_16d_10	0,063
fft_phase_skewness	−0,063
fft_magnitude_skewness	−0,063
raw_skewness	−0,063
autoencoder_16d_6	−0,062

Tabelle 5.1: Pearson Korrelationskoeffizienten von CT-Wert zu Read-Features

Die Tabelle bietet einen ersten Einblick in den Zusammenhang zwischen den Reads und dem CT-Wert. Auffällig ist, dass die Pearson-Korrelationswerte durchweg sehr niedrig sind; der höchste erreicht gerade einmal 0,11. In der Pearson-Korrelationsanalyse weisen Werte nahe ± 1 auf eine starke, lineare Korrelation hin, während Werte nahe 0 eine fehlende Korrelation anzeigen. In diesem Kontext ist die geringe Stärke der beobachteten Zusammenhänge besonders bemerkenswert.

Es ist jedoch zu beachten, dass die Pearson-Korrelation lediglich lineare Beziehungen erfasst. Daher soll im nächsten Experiment eine weitergehende Evaluation durchgeführt werden.

5.2.2 Klassifizierung von Reads

Im folgenden Experiment wird ein Random-Forest-Klassifikator trainiert. Dabei wird eine Anzahl von 128 Bäumen als Hyperparameter verwendet. Ziel ist es zu untersuchen, ob der Klassifikator in der Lage ist, Reads mit niedrigen CT-Werten (0-16) von Reads mit hohen CT-Werten (21-37) zu differenzieren. Für die Evaluation wird eine 5-Fold-Cross-Validation verwendet.

Die aus dem Training resultierenden Metriken sind in der folgenden Tabelle aufgeführt:

Metrik	Mittelwert	Standardabweichung	Varianz
Accuracy	0,563 740	0,008 009	0,000 064
ROC-AUC	0,585 914	0,006 765	0,000 046
Precision	0,571 402	0,005 580	0,000 031
Recall	0,509 627	0,014 717	0,000 217
F1-Score	0,538 695	0,010 647	0,000 113

Tabelle 5.2: Trainings Metriken des Random-Forest-Read-Klassifikators

Die in der Tabelle dargestellten Ergebnisse zeigen, dass das Modell in verschiedenen Leistungsmetriken Werte um 0,5 erreicht. Dies ist ein Indikator für eine Klassifikationsleistung, die kaum besser ist als zufälliges Raten. Bei binären Klassifikationsaufgaben entspricht ein Wert von etwa 0,5 in Metriken wie Genauigkeit, ROC-AUC, Präzision, Recall und F1-Score einem zufälligen Ergebnis. Dies legt nahe, dass das Modell möglicherweise nicht optimal auf die Daten abgestimmt ist, die gewählten Merkmale nicht ausreichend informativ sind oder die Daten selbst problematisch sein könnten. Diese Ergebnisse sind ein Anlass zur weiteren Untersuchung.

5.3 Analyse auf Datei-Ebene

Nach der detaillierten Analyse auf Read-Ebene, die gezeigt hat, dass es nur eine geringe Korrelation zwischen den Read-Features und dem CT-Wert gibt, richtet sich der Fokus nun auf eine Analyse auf Datei-Ebene. Die vorausgehenden Ergebnisse werfen die Frage auf, ob eine Aggregation der Daten auf höherer Ebene zu aussagekräftigeren Einsichten führen kann.

Die Analyse auf Datei-Ebene verfolgt mehrere Ziele. Erstens soll untersucht werden, ob durch eine Zusammenfassung der Read-Features auf Datei-Ebene stärkere Korrelationen zum CT-Wert erzielt werden können. Zweitens wird die Leistung eines Random-Forest-Klassifikators bei der Vorhersage des CT-Wertes auf dieser Ebene evaluiert. Für beide Schritte werden die aggregierten Daten herangezogen, die mittels des Skripts *create_file_features.py* aus den einzelnen Reads generiert werden. Dieses bildet die Features aller Reads im Datensatz und aggregiert diese dann pro Datei.

5.3.1 Korrelation zwischen Datei-Features und CT-Wert

Zunächst wird eine neue Korrelationsmatrix generiert, die die aggregierten Features auf Datei-Ebene mit dem CT-Wert in Beziehung setzt. Analog zur vorherigen Analyse wird die Darstellung der Ergebnisse auf die stärksten Korrelationen beschränkt, um eine übersichtliche und fokussierte Interpretation zu ermöglichen.

Feature	Korrelationskoeffizient
autoencoder_32d_2_std	0,355
autoencoder_32d_18_std	0,351
autoencoder_8d_4_std	0,343
autoencoder_32d_2_var	0,334
autoencoder_32d_28_std	0,332
autoencoder_8d_4_var	0,330
autoencoder_32d_4_std	0,328
autoencoder_32d_10_var	0,327
autoencoder_32d_18_var	0,325
autoencoder_16d_0_std	0,323

Tabelle 5.3: Pearson Korrelationskoeffizienten von CT-Wert zu Datei-Autoencoder-Features

Die Korrelationsmatrix zeigt, dass bei diesem Versuch bereits vielversprechende Korrelationen gefunden werden können, die weitaus höhere Werte aufweisen als die auf der Read-Ebene. Die stärksten Korrelationen werden, wie in der Tabelle zu sehen ist, durch die Standardabweichung und die Varianz der Autoencoder-Features gebildet. Insbesondere das 32-Feature-Modell zeigt eine gute Performance.

Feature	Korrelationskoeffizient
fft_magnitude_std_std	0,125
fft_phase_std_std	0,125
fft_magnitude_kurtosis_std	0,122
fft_phase_kurtosis_std	0,122
fft_magnitude_skewness_std	0,120

Tabelle 5.4: Pearson Korrelationskoeffizienten von CT-Wert zu Datei-FFT-Features

Die Features, die über die FFT erstellt wurden, weisen jedoch relativ geringe Korrelationskoeffizienten auf und scheinen somit von geringer Relevanz zu sein.

Feature	Korrelationskoeffizient
raw_mad_var	0,252
raw_iqr_var	0,240
raw_mad_std	0,235
raw_iqr_std	0,225
raw_rms_var	0,166

Tabelle 5.5: Pearson Korrelationskoeffizienten von CT-Wert zu Datei-Roh-Features

Einige manuell aus den Rohdaten konstruierte Features scheinen jedoch vielversprechend zu sein. Insbesondere stechen die Varianz und die Standardabweichung der Median Absolute Deviation (MAD) sowie des Interquartilsabstands (IQR) hervor. Diese weisen ähnlich wie die Autoencoder-Features verhältnismäßig hohe Korrelationskoeffizienten auf.

5.3.2 Klassifizierung von Dateien

Die Analyse auf der Read-Ebene verwendet ebenfalls ein Random-Forest-Modell mit 128 Bäumen, das durch eine 5-Fold-Crossvalidierung evaluiert wird.

Metrik	Mittelwert	Standardabweichung	Varianz
Accuracy	0,816 099	0,020 405	0,000 416
ROC-AUC	0,856 206	0,026 783	0,000 717
Precision	0,840 837	0,035 093	0,001 232
Recall	0,914 973	0,007 428	0,000 055
F1-Score	0,875 839	0,016 914	0,000 286

Tabelle 5.6: Trainings Metriken des Random-Forest-Datei-Klassifikators mit allen Features

Die Ergebnisse sind, wie die obere Tabelle unterstreicht, deutlich vielversprechender. Alle analysierten Metriken überschreiten den Wert von 0.8, was darauf hindeutet, dass das Modell effektiv zwischen Dateien mit niedrigen und hohen CT-Werten unterscheiden kann.

Die nachfolgende Analyse der Feature-Importance-Werte nach dem Training des Random-Forest-Modells offenbart, welche Variablen am meisten zur Leistung des Modells beitragen. Hier zeigen sich vor allem die Features der verschiedenen Autoencoder als besonders wichtig. Bei den aus den Rohdaten generierten Features fallen insbesondere die Median Absolute Deviation (MAD) sowie der Interquartilsabstand (IQR) auf, die ebenfalls eine hohe Bedeutung im Modell haben.

5.3.3 Klassifizierung von Dateien auf Untermengen der Features

Wie die Feature-Importance-Werte des mit allen Features trainierten Modells zeigen, verdeutlicht die folgende Tabelle, dass auch ein Modell, das ausschließlich auf Autoencoder-Features trainiert wurde, bereits gute Metriken erzielen kann.

Score	Mittelwert	Standardabweichung	Varianz
Accuracy	0,797 523	0,012 476	0,000 156
ROC-AUC	0,838 027	0,036 478	0,001 331
Precision	0,819 515	0,013 578	0,000 184
Recall	0,918 398	0,014 557	0,000 212
F1-Score	0,865 977	0,007 158	0,000 051

Tabelle 5.7: Trainings Metriken des Random-Forest-Datei-Klassifikators mit Datei-Autoencoder-Features

Anders verhält es sich bei einem Modell, das ausschließlich auf FFT-Features trainiert wurde. Dieses schneidet im Vergleich deutlich schlechter ab.

Score	Mittelwert	Standardabweichung	Varianz
Accuracy	0,690 402	0,034 420	0,001 185
ROC-AUC	0,619 926	0,015 814	0,000 250
Precision	0,739 219	0,033 553	0,001 126
Recall	0,872 812	0,018 862	0,000 356
F1-Score	0,800 214	0,025 658	0,000 658

Tabelle 5.8: Trainings Metriken des Random-Forest-Datei-Klassifikators mit Datei-FFT-Features

Das Modell, das mit Features aus Rohdaten trainiert wurde, schneidet besser ab als das FFT-Modell, jedoch schlechter als das Autoencoder-Modell.

Score	Mittelwert	Standardabweichung	Varianz
Accuracy	0,728 173	0,026 372	0,000 695
ROC-AUC	0,761 801	0,029 494	0,000 870
Precision	0,771 513	0,020 248	0,000 410
Recall	0,878 528	0,036 748	0,001 350
F1-Score	0,821 122	0,020 542	0,000 422

Tabelle 5.9: Trainings Metriken des Random-Forest-Datei-Klassifikators mit Datei-Roh-Features

Die Metriken, die in den vorangegangenen Tabellen dargestellt wurden, korrespondieren gut mit den Ergebnissen der Korrelationsanalyse. Dies unterstreicht die Gültigkeit der gewählten Methodik. Der Random-Forest-Klassifikator, der mit einer vollständigen Kombination aller Features trainiert wurde, zeigt die besten Ergebnisse in allen Bewertungskategorien.

Ein Modell, das nur auf Autoencoder-Features trainiert wurde, folgt in der Rangfolge der Effizienz. Seine Metriken liegen im höheren Bereich. Ein weiteres Modell, das auf Rohdaten-Features basiert, weist akzeptable, wenn auch nicht herausragende Ergebnisse auf. Am unteren Ende findet sich das FFT-basierte Modell, dessen Metriken im Vergleich deutlich abfallen.

5.3.4 Klassifizierung von Dateien mit mehr CT-Kategorien

In den vorangegangenen Experimenten haben wir die CT-Werte in zwei Kategorien eingeteilt: einer mit Werten von 0 bis 16 und einer zweiten mit Werten von 21 bis 37. Ziel war es, eine Unterteilung in niedrige und hohe CT-Werte zu schaffen. Dabei wurde bewusst eine Lücke im Bereich von 16 bis 21 ausgelassen, um dem Modell die Klassifizierung der Daten zu erleichtern. Nun stellt sich die Frage, welche Auswirkungen die Einführung einer dritten Kategorie für mittlere CT-Werte im Bereich von 16 bis 21 haben könnte.

Auch für dieses Modell werden die zuvor festgelegten Einstellungen beibehalten. Es kommt ein Random-Forest-Klassifikator mit 128 Bäumen zum Einsatz, der mittels 5-Fold-Cross-Validation evaluiert wird.

Score	Mittelwert	Standardabweichung	Varianz
Accuracy	0,672 874	0,017 793	0,000 317
ROC-AUC	0,768 471	0,015 877	0,000 252
Precision	0,669 248	0,020 846	0,000 435
Recall	0,672 874	0,017 793	0,000 317
F1-Score	0,651 121	0,019 941	0,000 398

Tabelle 5.10: Trainings Metriken des Random-Forest-Datei-Klassifikators mit drei Kategorien

Die Metriken wurden unter Verwendung der Einstellung **average='weighted'**[1] in der scikit-learn-Bibliothek kalkuliert. Diese Gewichtung ist besonders relevant, da sie die Verteilung der einzelnen Klassen im Datensatz berücksichtigt, um einen gewichteten Durchschnittswert zu ermitteln. Diese Anpassung ist sinnvoll, insbesondere wenn die Klassen im Datensatz ungleich verteilt sind.

Für die ROC-AUC-Metrik kam die Methode **multi_class='ovr'**[2] (One-vs-Rest) zum Einsatz. Diese Berechnungsmethode erlaubt eine spezifische Auswertung der Fähigkeit des Modells, zwischen den verschiedenen Klassen zu unterscheiden. Dabei wird für jede Klasse die AUC individuell berechnet, um daraus einen gewichteten Durchschnitt zu bilden.

Da es sich bei dem untersuchten Problem nicht mehr um eine binäre Klassifikation handelt, ist die Verwendung gewichteter Mittelwerte bei den Metriken erforderlich. Diese Methodik erlaubt eine differenzierte und aussagekräftige Beurteilung der Leistungsfähigkeit des Modells im Kontext einer Multiklassenklassifikation.

Die Ergebnisse deuten darauf hin, dass die Einführung einer zusätzlichen Kategorie für mittlere CT-Werte eine signifikante Verschlechterung der Metriken zur Folge hat. Diese Beobachtung könnte auf die erhöhte Komplexität des Klassifizierungsproblems zurückzuführen sein. Mit der Hinzufügung einer weiteren Klasse steigen die Anforderungen an das Modell, die vorhandenen Informationen effektiv zu nutzen und adäquate Vorhersagen zu treffen.

Es wird deutlich, dass das ursprüngliche Modell, welches nur niedrige und hohe CT-Werte unterscheidet, in der Lage ist, zuverlässigere Ergebnisse zu erzielen. Die Kompliziertheit einer Multiklassenklassifikation kann die Leistungsfähigkeit des Modells negativ beeinflussen.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html`
[2] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html`

5.3.5 Training eines Regression-Models

In einem finalen Experiment wird der Fokus von der Klassifikation zur Regression verlagert. Das Ziel besteht darin, ein Regressionsmodell zu entwickeln, das in der Lage ist, den CT-Wert basierend auf den Datei-Features vorherzusagen. Ähnlich wie im Klassifikationskontext wird hierfür ein Random-Forest-Modell eingesetzt, das aus 128 Bäumen besteht.

Dieser Ansatz könnte besonders wertvoll sein, wenn es nicht nur darum geht, die CT-Werte in bestimmte Kategorien einzuteilen, sondern eine kontinuierliche Vorhersage von Interesse ist. Durch die Verwendung eines Regressionsmodells könnten genauere und nuanciertere Einschätzungen der CT-Werte möglich werden, die weit über eine bloße Kategorisierung hinausgehen.

Score	Mittelwert	Standardabweichung	Varianz
Mean Absolute Error	−2,121 221	0,073 200	0,005 358
Mean Squared Error	−7,439 045	0,545 917	0,298 026
R-Quadrate R^2	0,413 408	0,048 358	0,002 339
Explained Variance Score	0,414 324	0,048 243	0,002 327

Tabelle 5.11: Trainings Metriken der Random-Forest-Datei-Regression

Die Tabelle stellt erste Metriken zur Verfügung, die jedoch schwierig zu interpretieren sind und in erster Linie als Vergleichsbasis für zukünftig entwickelte Modelle dienen sollen. Um ein detaillierteres Verständnis der Modellqualität zu erhalten, kann eine optische Visualisierung aufschlussreicher sein.

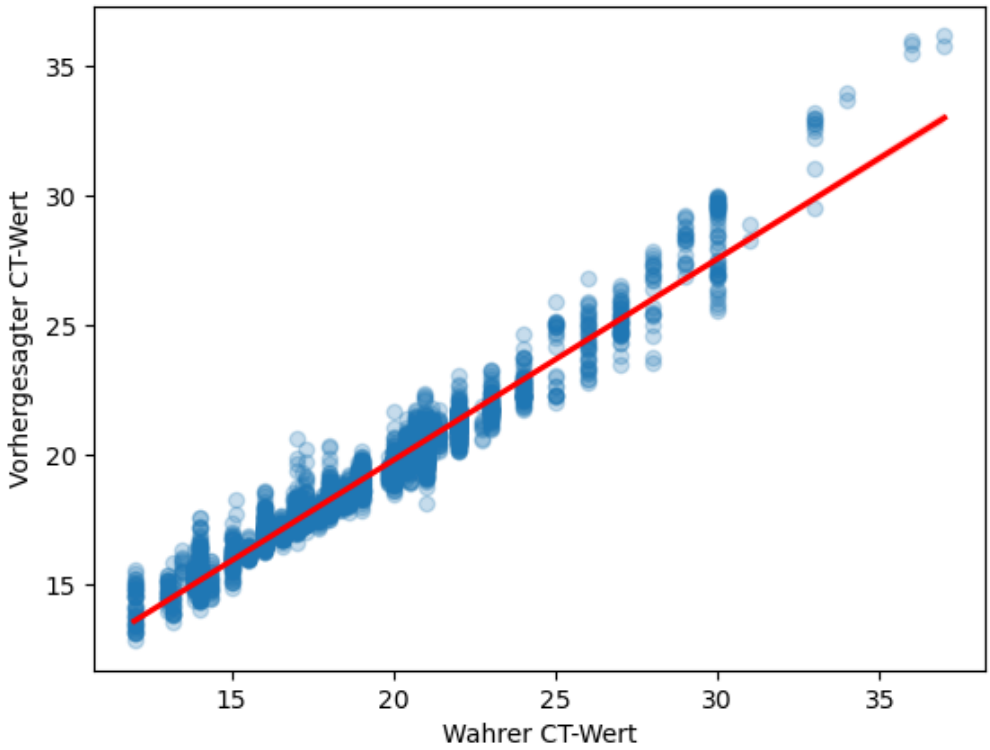


Abbildung 5.4: Regression: Vorhergesagter- vs. Wahrer CT-Wert

Die dargestellte Grafik zeigt, dass das Modell bereits in der Lage ist, auf Basis der Datei-Features eine Regression durchzuführen. Obwohl die Ergebnisse nicht exakt sind, demonstriert die Grafik die Möglichkeit, eine ungefähre Vorhersage des CT-Werts zu treffen. Dies legt nahe, dass mit weiterer Feinabstimmung und Anpassung des Modells eine immer genauere Vorhersage erreichbar sein könnte.

6 Diskussion

6.1 Interpretation der Trainingsergebnisse

6.2 Kritik

6.3 Verbesserungsvorschlaege

6.4 Ausblick

7 Zusammenfassung

Literature references

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts und P. Walter, *Molecular Biology of the Cell*, 6th. Garland Science, 2014.

[2] H. Lodish, A. Berk, C. A. Kaiser u. a., *Molecular Cell Biology*, 8th. W. H. Freeman, 2016.

[3] T. Strachan und A. P. Read, *Human Molecular Genetics*, 4th. Garland Science, 2010.

[4] R. L. Nussbaum, R. R. McInnes und H. F. Willard, *Thompson & Thompson Genetics in Medicine*, 8th. Elsevier, 2016.

[5] M. L. Metzker, „Sequencing technologies - the next generation," *Nature Reviews Genetics*, Jg. 11, Nr. 1, S. 31–46, 2010.

[6] S. Goodwin, J. D. McPherson und W. R. McCombie, „Coming of age: ten years of next-generation sequencing technologies," *Nature Reviews Genetics*, Jg. 17, Nr. 6, S. 333–351, 2016.

[7] E. van Dijk, H. Auger, Y. Jaszczyszyn und C. Thermes, „Next-generation sequencing technologies and their impact on microbial genomics," *Briefings in functional genomics*, Jg. 12, Nr. 5, S. 440–453, 2017.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, 10.09.2023
Christoph Stach