Einsatz von Machine Learning zur Evaluierung der Qualität von Next-Generation-Sequencing-Daten

Forschungsprojekt Teil B

Alexander Hinzer
Alexander.Hinzer@student.htw-berlin.de

Prüfer: Prof. Dr.-Ing. Piotr Wojciech Dabrowski

Studiengang: Angewandte Informatik (Master)
Fachbereich 4
Hochschule für Technik und Wirtschaft Berlin

Wintersemester 2021/22

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Fortsetzung des Projekts

In Teil A des vorliegenden Forschungsprojekts wurde gezeigt, wie die Evaluierung der Qualität von Next-Generation-Sequencing-Daten durch Machine Learning (ML) umgesetzt werden kann. Die implementierten ML-Modelle sind in der Lage, die Qualität der gegebenen Sequenzierungsdaten beinahe perfekt zu klassifizieren. Da die Klassen der vorliegenden Daten jedoch besonders eindeutig voneinander zu unterscheiden sind, liegt die Vermutung nahe, dass die Performance der ML-Modelle für komplexere Datensätze mit mehr Grenzfällen weitaus schlechter ausfallen könnte. Der Grund für diese Vermutung ist, dass die verwendeten Features der Modelle besonders einfach sind und einen Großteil der Informationen, die in einer Qualitätsanalyse der FastQC-Software [1] entstehen, nicht darstellen.

Des Weiteren wurde im Teil A des Projekts bisher nicht untersucht, wie die entstandene Implementierung in die automatisierten Abläufe der Verarbeitungsprozesse (Pipeline) von Sequenzierungsdaten integriert werden kann. Dies ist, hinsichtlich der Zielsetzung die manuelle Arbeit der Biologen in der Qualitätseinschätzung zu ersetzen, ein wichtiger Schritt.

1.2 Zielstellung

Somit liegt das Ziel des Teils B der Forschungsarbeit zum einen in der Verfeinerung der Features der ML-Modelle und zum anderen in der Implementierung einer Pipeline, mit welcher sich diese Features in bestehende bioinformatische Prozesse integrieren lassen. Die Pipeline soll mit Nextflow [2] umgesetzt werden und in der Lage sein, für die Eingabe einer FASTQ-Sequenzierungsdatei deren Qualitätsbewertung auszugeben. Zusätzlich soll der Prozess der ML-Modell-Erstellung in einem parametrisierten Python-Skript ausführbar sein, inklusive der Datenextraktion, des Feature Engineerings, des Modell-Trainings und dessen Exportierung.

In Teil A des Projekts wurden die einzelnen FastQC-Module nur durch ihre automatisierte Bewertung als Features repräsentiert. Das Ziel dieses Teils der Forschungsprojekts soll sein, aus den in jedem Modul enthaltenen komplexeren Daten, neue Features für das Training des Modells zu erstellen.

2 Feature Engineering

2.1 Hintergrund

Entscheidend für die Performance eines ML-Modells ist die Qualität der Features mit denen es trainiert wird. Feature Engineering beschreibt die Vorverarbeitung der Rohdaten eines Datensatzes zu sinnvollen und aussagekräftigen Features, die geeignet sind um die gewählten ML-Algorithmen umzusetzen. Rohdaten können kategorische oder komplexe Daten enthalten, wie zum Beispiel Listen oder Tabellen (Matrizen). Für diese muss eine geeignete Normalisierung und zahlenmäßige Repräsentation (in Form von Features) gefunden werden, bevor ihre Informationen im ML-Modell verarbeitet werden können. [3]

Das Feature Engineering für kategorische Daten wurde bereits in Projektteil A umgesetzt, indem die Modul-Status als Zahlen kodiert wurden. In diesem Projekt sollen nun auch die Inhalte der Module als Features implementiert werden. Die Module enthalten jeweils komplexe Daten in Form von Matrizen in der Größenordnung von bis zu mehreren hundert Zeilen. Um daraus bedeutungsvolle Features extrahieren zu können, ist Fachwissen über die Inhalte der Module notwendig, um die richtigen mathematischen beziehungsweise statistischen Operationen anwenden zu können.

2.2 Feature Engineering kontinuierlicher Daten

Um kontinuierliche Daten, wie sie in den gegebenen Matrizen vorliegen, in prägnanten Features zusammenfassen, bieten sich verschiedene statistische oder mathematische Mittel an. Die günstigste Umsetzung hängt von der genauen Struktur und den zugrundeliegenden fachlichen Informationen ab.

Eine Möglichkeit kontinuierliche Daten zu beschreiben, ist deren Häufigkeitsverteilung. Diese kann in einem Histogramm dargestellt werden. Das Histogramm teilt die Häufigkeiten der Werte in definierbare Klassen (engl.: Bins) von Wertebereichen. Diese Bins können direkt als Features verwendet werden. Solch ein Feature repräsentiert damit die Häufigkeit der Datenwerte, die in dem definierten Bereich vorkommen. Dabei ist eine Abwägung zu treffen zwischen der Breite des Wertebereichs jedes Bins und dem aus der Zusammenfassung resultierenden Informationsverlust. In der Regeln entstehen aus dieser Herangehensweise zu viele Features. [3]

Die Verteilung lässt sich über ihre Parameter in wesentlich weniger Features darstellen. Dafür muss abgeschätzt werden, um welche Art der Verteilung es sich bei den Daten handelt. Handelt es sich um eine Normalverteilung, so sind die Parameter der Mittelwert μ und die Varianz σ^2. Bei einer Beta-Verteilung sind die Parameter α und β [4]. Damit kann eine Verteilung in jeweils zwei Features dargestellt werden. Welche Verteilung am besten zu den Daten passt, muss in der Daten Exploration untersucht werden. Eine weitere Vorgehensweise kontinuierliche Daten zusammenzufassen, ist die Anpassung einer Polynomfunktion für die Daten [5]. Die daraus resultierenden Parameter können als Features verwendet werden. Die Anzahl der entstehenden Features ist um eins größer als der Grad des verwendeten Polynoms.

3 Feature Engineering der FastQC-Module

3.1 Überblick

FastQC fasst die Ergebnisse der Analyse der Sequenzierungsdaten in 11 Modulen zusammen. Im folgenden Abschnitt werden die einzelnen Module vorgestellt und vorgeschlagen, durch welche Features sie dargestellt werden können. Alle in den Modulen visualisierten Werte werden von FastQC auch in tabellarischer Form für die Weiterverarbeitung bereitgestellt.

3.2 Basic Statistics

3.2.1 Inhalt des Moduls

Das Modul „Basic Statistics“ enthält Grundlegende Informationen der analysierten FASTQ-Dateien in tabellarischer Form. Diese sind in der Tabelle 3.1 gelistet und kurz erklärt.

Information	Erklärung
Filename	Name der analysierten Datei
File type	Angabe, ob die Daten Base Calls enthält oder Farbraum-Daten, die konvertiert werden mussten
Encoding	Verwendete ASCII-Kodierung der Datei
Total Sequences	Anzahl der enthaltenen Sequenzen
Filtered Sequences	Anzahl der Sequenzen, die aufgrund schlechter Qualität in der weiteren Analyse ignoriert werden
Sequence Length	Angabe der kürzesten und längsten enthaltenen Sequenzlänge
%GC	Gesamter GC-Anteil in allen Sequenzen

Tabelle 3.1: Informationen in „Basic Statistics“[1]

3.2.2 Feature Engineering

Das Feature Engineering für dieses Modul wurde bereits im Teil A des Projekts umgesetzt. Die folgenden Informationen werden aus dem Modul extrahiert und können direkt als Features verwendet werden: Anzahl der Sequenzen (*total_sequences*), GC-Anteil (*percent_gc*), kürzeste Sequenzlänge (*min_sequence_length*) und längste Sequenzlänge (*max_sequence_length*).

3.3 Per Base Sequence Quality

3.3.1 Inhalt des Moduls

Das Modul „Per Base Sequence Quality“ aggregiert die Phred-Werte der Base Calls über deren Position im Sequenzierungsvorgang. Dafür werden für jede Position verschiedene Messwerte berechnet und in einem Box-Whisker-Plot, wie in Abbildung 3.1 dargestellt. Die dargestellten Werte sind der Mittelwert (blaue Kurve), Median (rote Linie), das obere und untere Quartil (gelber Kasten) und das 10. und 90. Perzentil (schwarze Whisker). Je höher der Phred-Wert, desto besser die Qualität der Base Calls. Dies ist in der Visualisierung durch den Hintergrund dargestellt, sehr gute Werte liegen im grünen Bereich, akzeptable Werte im gelben und schlechte Werte im roten Bereich. Üblicherweise sinkt die Qualität der Base Calls über die Dauer des Sequenzierungsverfahrens, somit nehmen die Phred-Werte an höherer Position in der Regel ab. [1]

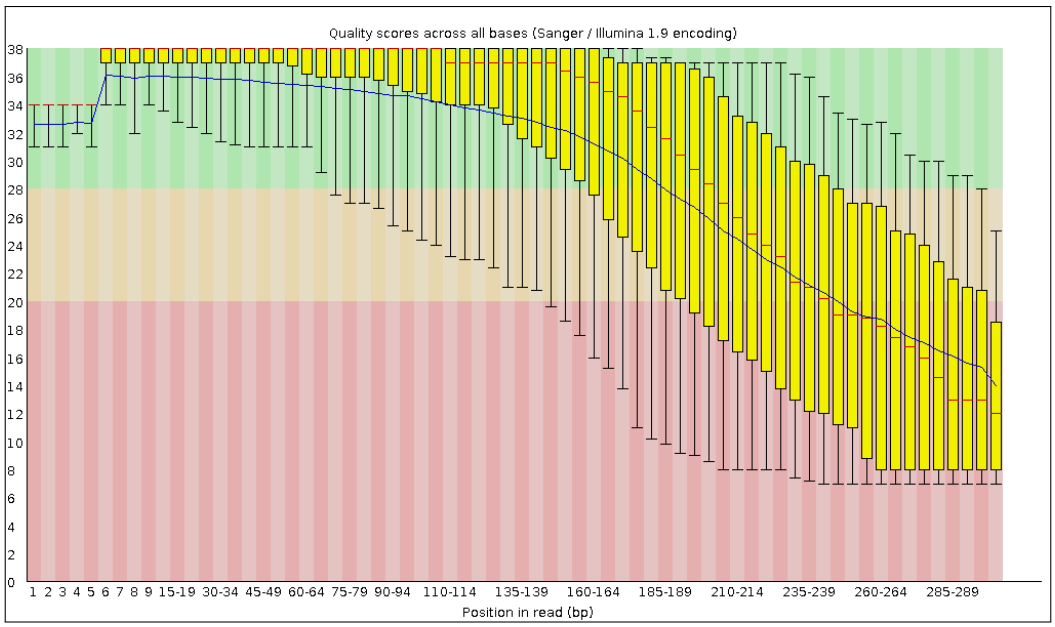

Abbildung 3.1: Beispiel „Per Base Sequence Quality“-Darstellung (erstellt mit [1])

Eine Read guter Qualität verliert in höheren Positionen nur leicht an Qualität, ein schlechter Read beginnt schon in früherer Position abzusinken und fällt zum Ende bis in den gelben oder roten Bereich ab.

Die automatisierte FastQC Bewertung des Moduls ergibt „Warning“, wenn die untere Quartile an einer Position unter einen Phred-Wert von 10 oder der Median an einer Position unter einen Phred-Wert von 25 fällt. Die Einschätzung „Failure“ resultiert bei einer unteren Quartile unter 5 oder einem Median von unter 20. [1]

3.3.2 Feature Engineering

Wie in Abbildung 3.1 dargestellt, werden von FastQC die Positionen ab der 10. Position in Gruppen von fünf Positionen zusammengefasst. Damit eine möglichst korrekte Repräsentation berechnet werden kann, sollten die Daten jedoch gleichmäßig vorliegen. Um dies zu erreichen

kann die von FastQC durchgeführte Berechnung manuell implementiert werden, indem die rohen FASTQC-Daten importiert werden. Dabei wird nur der Mittelwert der Phred-Werte je Position weiterverwendet, um die Anzahl resultierender Features möglichst klein zu halten.

Für das Feature Engineering dieses Moduls wird die Kurve einer Polynomfunktion angepasst. Es konnte festgestellt werden, dass ein Polynom dritten Grades die Kurven ausreichend gut erfasst. Abbildung 3.2 zeigt ein Beispiel der Anpassung. Dargestellt sind die gegebenen Datenpunkte (grün gestrichelte Kurve) sowie die resultierende Polynomfunktion (rote Kurve).

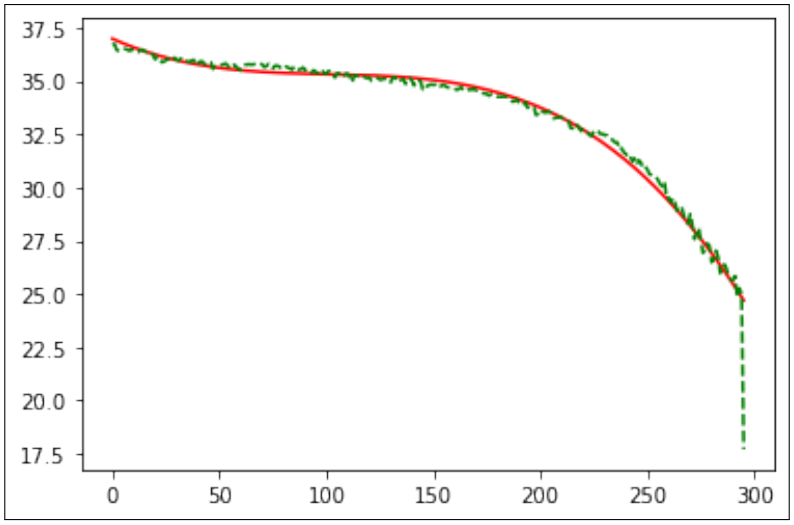

Abbildung 3.2: Beispiel der Anpassung einer Polynomfunktion an die Mittelwerte der Phred-Werte je Position (erstellt mit [6])

Auf Grundlage der Polynomfunktion dritten Grades aus der Formel 3.1 ergeben sich damit vier Features für das Modul „Per Base Sequence Quality“, welche als *module_1_a0*, *module_1_a1*, *module_1_a2* und *module_1_a3* bezeichnet werden.

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{3.1}$$

3.4 Per Tile Sequence Quality

3.4.1 Inhalt des Moduls

Das Modul „Per Tile Sequence Quality“ zeigt Abweichungen in der Durchschnittsqualität der Reads je Kachel (engl.: Tile) der Fließzelle an jeder Position. Diese werden farblich in einer kaltheiß-Skala dargestellt. Dabei ist ein Tile, das dem Durchschnitt entspricht, blau und ein Teil, das sehr stark davon abweicht, rot eingefärbt. Der Zwischenbereich ist dem Verlauf über türkis, grün, gelb und orange entsprechend gefärbt. Eine Sequenzierung guter Qualität sollte vollständig blau sein, eine Sequenzierung schlechter Qualität mit vielen problematischen Tiles ist in Abbildung 3.3 dargestellt. Die FastQC Bewertung des Moduls ergibt „Warning“, wenn ein Tile einen um zwei kleineren Phred-Wert als der durchschnittliche Phred-Wert dieser Position aufweist. Es wird „Failure“ ausgegeben, wenn ein Tile um fünf von dem Durchschnitt der Phred-Werte dieser Position abweicht. [1]

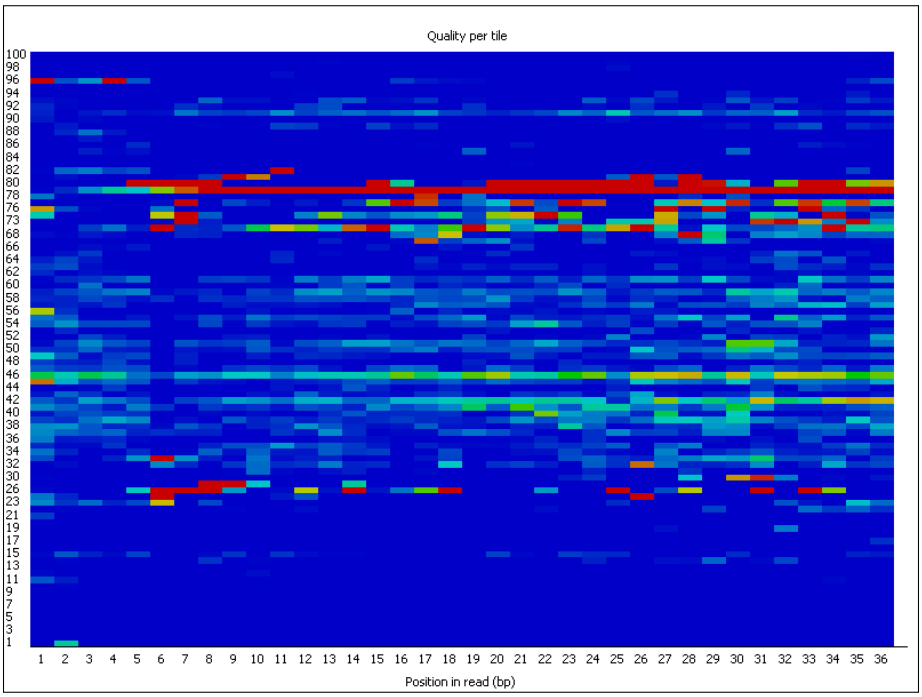

Abbildung 3.3: Beispiel „Per Tile Sequence Quality“-Darstellung [1]

3.4.2 Feature Engineering

Um die Dimensionalität der Informationen dieses Moduls zu reduzieren, werden die Abweichungen pro Tile gemittelt. Da die Aufstellung der Daten durch FastQC wieder mit einer Zusammenfassung von 5 Positionen ab der 10. Position einhergeht, muss dies bei der Durchschnittsbildung durch Anwenden eines gewichteten Durchschnitts berücksichtigt werden. In der Daten Exploration konnte festgestellt werden, dass die resultierenden Werte eine bimodale Normalverteilung aufweisen. Da die Abweichungen gleichermaßen negativ, als auch positiv auftreten können, werden die beiden Verteilungen durch den Nullpunkt voneinander getrennt. Durch Feature Testing konnte festgestellt werden, dass sich die jeweiligen Verteilungen am günstigsten durch ihre Standardabweichung repräsentieren lassen. Damit konnten die Features auf die positive und negative Standardabweichung der pro Tile Phred-Wert-Abweichungen reduzieren lassen. Diese Features werden im ML-Modell als *module_2_std_pos* und *module_2_std_neg* betitelt.

3.5 Per Sequence Quality Scores

3.5.1 Inhalt des Moduls

Das Modul „Per Sequence Quality Scores“ fasst die Anzahl der entsprechenden Sequenzen für jeden durchschnittlichen Phred-Wert zusammen. Es zeigt also die Häufigkeitsverteilung der Mittelwerte der Phred-Werte je Sequenz. Der Großteil der Reads sollte hier in einem guten Phred-Wert Bereich liegen. FastQC gibt ein „Warning“ für dieses Modul, wenn der häufigste Durchschnittliche Phred-Wert unter 27 liegt. Dies entspricht einer Fehlerrate von 0,2%. Ein „Failure“ wird ausgegeben, wenn der häufigste Phred-Wert unter 20 liegt, äquivalent zu einer Fehlerrate von 1%. [1]

Ein Beispiel für die Visualisierung dieses Moduls ist in Abbildung 3.4 dargestellt.

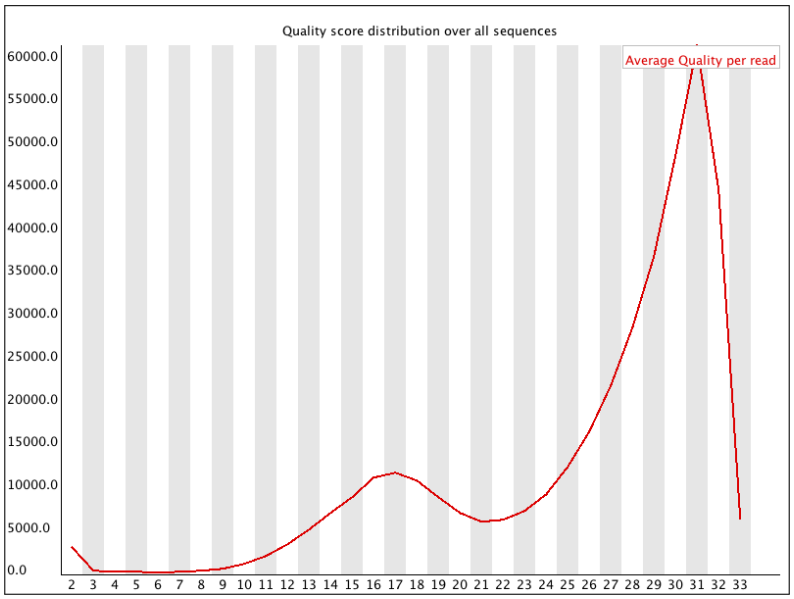

Abbildung 3.4: Beispiel „Per Sequence Quality Scores“-Darstellung [1]

3.5.2 Feature Engineering

Da es sich bei diesen Daten um eine Häufigkeitsverteilung handelt, kann für eine Repräsentation die Art der Verteilung abgeschätzt werden. Es kann festgestellt werden, dass sie einer Beta-Verteilung entspricht. Für die Extraktion der Features werden die Werte daher der Beta-Verteilungsfunktion aus Formel 3.3 angepasst. Sie ist eine Erweiterung der Gamma-Verteilung $\Gamma(z)$ aus Formel 3.2 [7].

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{3.2}$$

$$f(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta) x^{\alpha-1} (1-x)^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)} \tag{3.3}$$

Damit sind die resultierenden Features die α und β Parameter der angepassten Funktion. Diese werden in der Implementation als *module_3_alpha* und *module_3_beta* bezeichnet.

3.6 Per Base Sequence Content

3.6.1 Inhalt des Moduls

Das Modul „Per Base Sequence Content“ zeigt den Anteil der jeweiligen Nukleinbasen Adenin (A), Cytosin (C), Guanin (G) und Thymin (T) der Base Calls für die Positionen der Sequenzierung, dargestellt in Abbildung 3.5. In einer unauffälligen Sequenzierung sollten diese Anteile parallel zueinander verlaufen und dem erwarteten Gesamtanteil der Nukleinbasen für die sequenzierte DNA entsprechen. Die FastQC-Bewertung für dieses Modul ist „Warning“, wenn der Unterschied

zwischen A und T oder G und C über 10% liegt. „Failure“ liegt bei einer Abweichung von über 20% vor. [1]

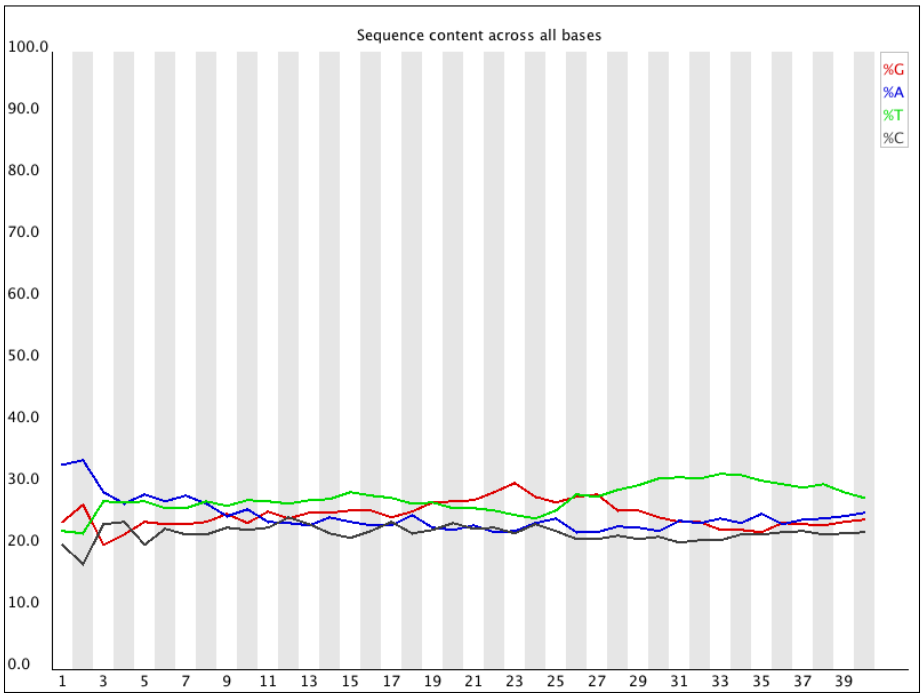

Abbildung 3.5: Beispiel „Per Base Sequence Content“-Darstellung [1]

3.6.2 Feature Engineering

Um die Beziehungen und Verläufe der Anteile der Nukleinbasen prägnant zusammenzufassen, wird von einer direkten Abbildung dieses Moduls abgesehen. Stattdessen wird die absolute Anteilsveränderung jeder Base von einer Position zur nächsten berechnet und für alle vier Nukleinbasen summiert. Diese Zusammenfassung der Anteilsveränderungen aller Basen weist eine Normalverteilung auf und kann durch ihren Durchschnitt und ihre Standardabweichung repräsentiert werden. Dies sind die Features die verwendet werden können, im ML-Modell als *module_4_diff_mean* und *module_4_diff_std* bezeichnet.

3.7 Per Sequence GC Content

Das Modul „Per Sequence GC Content“ zeigt die Verteilung des GC-Anteils, also die Anzahl der Sequenzen je Prozentzahl des Anteils der Summe aus G und C Nukleinbasen. Diese soll im Idealfall normalverteilt sein, dazu wird in der Darstellung, wie in Abbildung 3.6, zusätzlich eine Normalverteilung zum Vergleich als blaue Kurve visualisiert. FastQC liefert eine „Warning“-Bewertung, wenn mehr als 15% der Reads von der Normalverteilung abweichen und eine „Failure“-Bewertung, wenn mehr als 30% der Reads davon abweichen. [1]

In dieser Arbeit wird angenommen, dass der Informationsgehalt dieses Moduls schon in seinem Status und den Features des Moduls „Per Base Sequence Content“ ausreichend repräsentiert wird. Daher werden keine weiteren Features für dieses Modul erstellt.

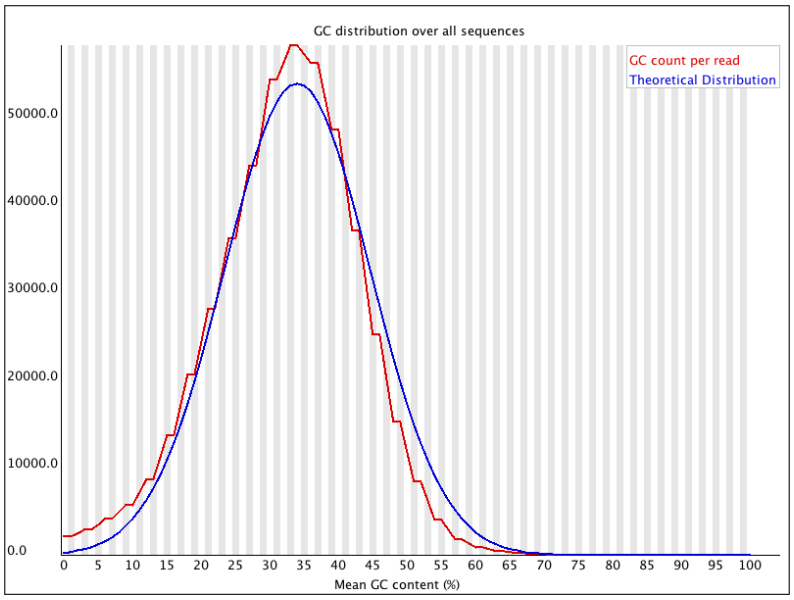

Abbildung 3.6: Beispiel „Per Sequence GC Content“-Darstellung [1]

3.8 Per Base N Content

3.8.1 Inhalt des Moduls

Sollte im Sequenzierungsprozess eine Base nicht identifiziert werden können, wird sie in die Sequenz mit dem Buchstaben N eingetragen. Das Modul „Per Base N Content“ stellt den Prozentanteil der N Eintragungen je Position dar. In Sequenzierungen guter Qualität sollte dieser Anteil an jeder Position möglichst nahe bei 0 liegen. Ein Anstieg an einer oder mehreren Positionen deutet auf Probleme beim Sequenzierungsprozess hin. Daher gibt die FastQC-Bewertung für den Status des Moduls ein „Warning“ aus, wenn eine Position einen N-Anteil von über 5% hat und ein „Failure“, wenn eine Position einen N-Anteil von 20% aufweist. [1]

3.8.2 Feature Engineering

Dieses Modul wird in einem Feature zusammengefasst, welches bei der Importierung der Rohdaten, nicht direkt aus den Modulergebnissen, extrahiert wird. Dies ist der gesamte Prozentanteil an N Base Calls. Das Feature wird im ML-Modell *module_6_n_content* genannt.

3.9 Sequence Length Distribution

3.9.1 Inhalt des Moduls

Das Modul „Sequence Length Distribution“ zeigt die Verteilung der Sequenzlängen aller Reads gruppiert für zusammengefasste Längenbereiche, wie in Abbildung 3.7 dargestellt. Der Status dieses Moduls wird als „Warning“ eingestuft, wenn Sequenzen unterschiedlicher Länge vorliegen. Es wird als „Failure“ bewertet, wenn Sequenzen mit einer Länge von 0 enthalten sind. [1]

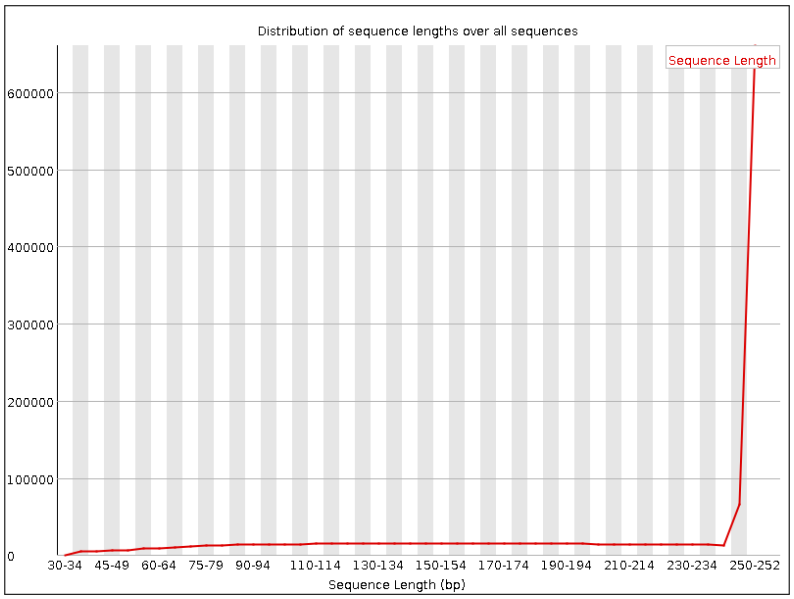

Abbildung 3.7: Beispiel „Sequence Length Distribution“-Darstellung (erstellt mit [1])

3.9.2 Feature Engineering

Ähnlich dem Modul „Per Sequence Quality Scores“ handelt es sich bei den Daten dieses Moduls um eine Häufigkeitsverteilung, die als Beta-Verteilung eingeordnet werden kann. Daher wird hier das selbe Verfahren verwendet, um das Modul durch die Parameter der Beta-Verteilung α und β darzustellen. Abbildung 3.8 zeigt ein Beispiel der resultierenden Anpassung der Kurve an die Datenpunkte. Die grün gestrichelte Kurve sind die Datenpunkte, die rote Kurve die Darstellung der Beta-Verteilung mit den berechneten Parametern.

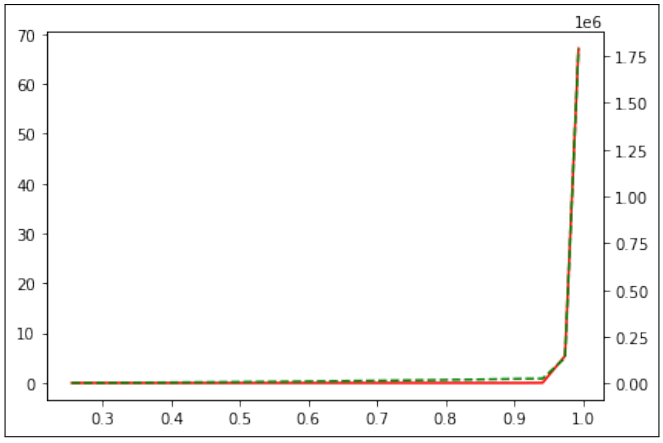

Abbildung 3.8: Beispiel der Anpassung einer Beta-Verteilung an die Verteilung der Sequenzlängen (erstellt mit [6])

Die Features für dieses Modul werden im ML-Modell als *module_ 7_ alpha* und *module_ 7_ beta* bezeichnet.

3.10 Sequence Duplication Levels

3.10.1 Inhalt des Moduls

Das Modul „Sequence Duplication Levels“ beschreibt die Verteilung der gefundenen Wiederholungen von Sequenzen. Dafür werden die Anzahl der Wiederholungen einer Sequenz auf der x-Achse abgebildet, wobei diese Anzahl ab 10 in unterschiedlich großen Gruppen zusammengefasst wird, siehe Abbildung 3.9. Die y-Achse stellt den Prozentanteil der Sequenzen mit dieser Anzahl dar. Diese Informationen werden für alle Sequenzen in der blauen Kurve dargestellt und für deduplizierte Sequenzen in der roten Kurve. Zusätzlich wird über der Visualisierung die verbleibende Prozentzahl an Sequenzen nach einer Deduplizierung ausgegeben. FastQC bewertet den Status dieses Moduls mit einem „Warning“, wenn mehr als 20% aller Sequenzen nicht einzigartig sind. Es wird mit einem „Failure“ bewertet, wenn mehr als 50% aller Sequenzen nicht einzigartig sind. [1]

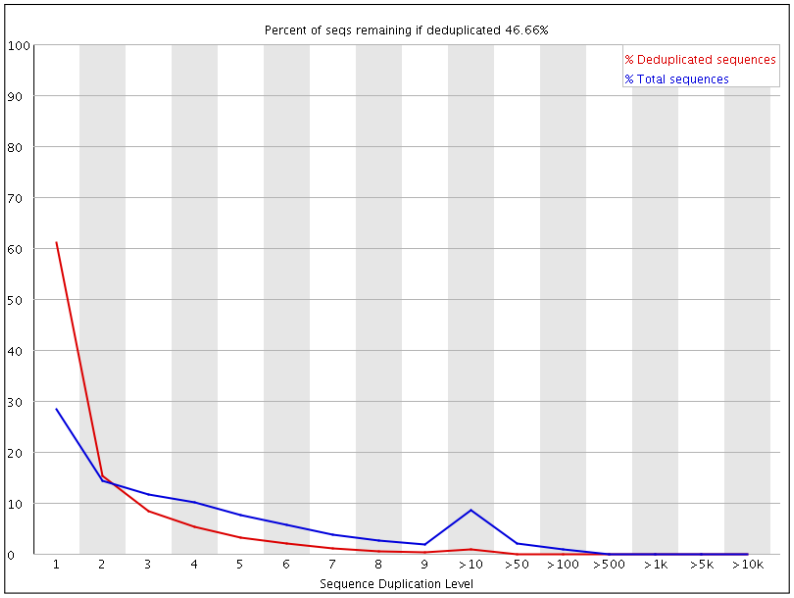

Abbildung 3.9: Beispiel „Sequence Duplication Levels“-Darstellung [1]

3.10.2 Feature Engineering

Bei den Daten dieses Moduls handelt es sich um eine Häufigkeitsverteilung, für die nach der selben Vorgehensweise wie in den Modulen „Per Sequence Quality Scores“ und „Sequence Length Distribution“ eine Beta-Verteilungsfunktion angepasst werden kann. Daraus ergeben sich für dieses Modul die Features *module_8_alpha* und *module_8_beta*.

3.11 Overrepresented Sequences

Das Modul „Overrepresented Sequences“ enthält keine Visualisierung, sondern eine Tabelle von allen Sequenzen (mit einer Mindestlänge von 20 Basenpaaren), die so häufig wiederholt vorliegen, dass sie mehr als 0,1% aller Sequenzen ausmachen. Der Status wird als „Warning“ ausgegeben,

wenn mindestens eine solche Sequenz gefunden wurde. Ein „Failure“ resultiert bei Vorliegen mindestens einer Sequenz, die mehr als 1% aller Sequenzen darstellt. [1]

Es wird angenommen, dass die Informationen dieses Moduls bereits ausreichend gut durch dessen Status und die Features des Moduls „Sequence Duplication Levels“ repräsentiert wird. Daher werden hierfür keine weiteren Features implementiert.

3.12 Adapter Content

Das Modul „Adapter Content“ visualisiert den Anteil der gefundenen Adaptersequenzen für jede Position. FastQC bewertet das Modul mit „Warning“, wenn eine Adaptersequenz in mehr als 5% aller Reads auftritt. Es wird mit „Failure“ bewertet, wenn eine Adaptersequenz mehr als 10% aller Reads ausmacht. [1]

In den vorliegenden Daten konnten keine aussagekräftigen Informationen aus diesem Modul gewonnen werden, daher wird von einer Implementierung von Features für dieses Modul abgesehen.

4 Implementierung einer Pipeline

4.1 Python-Skript und Module

Der gesamte ausgearbeitete Prozess bin hin zur Erstellung der ML-Modelle wird in einem Python-Skript zusammengefasst. Dafür werden die einzelnen Verarbeitungsschritte in 4 Python-Module ausgelagert.

Das erste Modul (*fastq_ extract*) dient der Extraktion der Sequenzierungsdaten aus den FASTQ-Dateien. Hier werden die FASTQ-Dateien durch FastQC ausgewertet und importiert. Außerdem werden auch Daten aus den rohen FASTQ-Dateien importiert, da die FastQC-Ergebnisse für einige Module nicht ausreichen, um die entsprechenden Features zu konstruieren. Darüberhinaus müssen auch die Informationen zur Gesamteinschätzung der Qualität durch Biologen und dem sequenzierten Organismus aus der Datenstruktur extrahiert weden. Das Modul *fastq_ extract* fasst all diese Daten zusammen, speichert sie zwischen, um bei wiederholtem Import der selben Dateien Zeit einsparen zu können und stellt sie für die Weiterverarbeitung bereit.

Das zweite Modul (*data_ preparation*) säubert die Daten, kodiert kategorische Felder und transformiert alle Datenpunkte so, dass sie für die folgenden Verarbeitungsschritte geeignet sind.

Das dritte Modul (*feature_ engineering*) implementiert alle in Kapitel *3* dargestellten Feature Engineering Transformationen, sodass abschließend die Sequenzierungsdaten in ihren Features und Labels für die ML Implementierung vorliegen. Der resultierende Datensatz wird auch hier zwischengespeichert.

Im vierten Modul (*ml_ model*) wird das ML-Modell trainiert und auf Wunsch evaluiert. Das Modell wird exportiert und kann damit für die Klassifizierung der Qualität von neuen FASTQ-Dateien verwendet werden. Dieses Modul implementiert auch die Anwendung des exportierten ML-Modells auf einen neuen Datensatz.

Ein Python-Skript zum Training setzt die Module mit anpassbaren Parametern ein, um eine Datei des ML-Modells aus einem FASTQ-Datensatz zu erhalten.

Ein Python-Skript zur Klassifikation verwendet ebenso alle Module, um aus einer (oder mehreren) FASTQ-Datei(en) und einem ML-Modell eine Qualitätseinschätzung zu erhalten.

4.2 Nextflow

Nextflow ist ein Workflow Management System, welches für bioinformatische Prozessabläufe eingesetzt werden kann, um einzelne Arbeitsschritte unabhängig von ihrer jeweiligen Implementierung miteinander zu verknüpfen. Dies ermöglicht eine einfache Reproduzierbarkeit, Parallelisierbarkeit und Erweiterbarkeit der bioinformatischen Pipeline. Dafür wird das datenstromorientierte (dataflow) Programmierparadigma umgesetzt. Jeder Prozess ist unabhängig von den anderen und kann ausgeführt werden, sobald er die benötigten Daten über einen Input-Kanal erhält. Daraufhin können die, im Prozess definierten, Datenmanipulationen durch Ausführen beliebiger Skripte (unter anderem Bash, Python, Perl) abgewickelt werden. Die Ergebnisse werden über einen Output-Kanal ausgegeben und stehen damit für andere Nextflow-Prozesse zur Weiterverarbeitung bereit. Die Implementierung wird mit einer domänenspezifischen Sprache (engl.: domain specific languague, DSL), basierend auf Groovy, umgesetzt. [2]

Nextflow wird auch für die bioinformatischen Pipelines in der Analyse von Genomsequenzierungsdaten am Robert Koch-Institut eingesetzt. Daher ermöglicht die Einbindung des, in diesem Projekt erstellten, Evaluierungsprozesses in einem Nextflow-Prozess eine direkte Einbindung in die bestehenden Datenverarbeitungsabläufe des Auftraggebers.

In diesem Projekt wird die Klassifikation der Qualität einer FASTQ-Datei in einem Nextflow-Prozess umgesetzt. Der Input für diesen Prozess sind die Datei, das Modell und ein Dateipfad für die erzeugten Zwischenergebnisse. Der Output ist eine Textdatei mit der resultierenden Bewertung.

5 Ergebnisse

5.1 Übersicht der Features

Für das Training der Modelle wurden alle im Feature Engineering erstellten Features, sowie die in Projektteil A verwendeten Features eingesetzt. Abbidlung 5.1 zeigt alle Features und deren Korrelation miteinander und mit dem Label (*target*). Wie schon im Teil A des Projekts beobachtet, haben weiterhin die Status von Modul 1 und 2 die größte Korrelation mit dem Label. Darüberhinaus weisen die neuen Features der Module „Per Tile Sequence Quality“, „Sequence Length Distribution“, „Per Base Sequence Content“ und „Sequence Duplication Levels“ Korrelationen mit dem Label auf. Zwischen den Features der Module 8, 2 und 4 ist eine starke Korrelation zu beobachten. Außerdem ist auffällig, dass viele der erstellten Features innerhalb eines Moduls eine hohe Korrelation haben. So zu erkennen zwischen den Features der Module „Per Base Sequence Quality“ (*module_1_a0, module_1_a1, module_1_a2, module_1_a3*), „Per Tile Sequence Quality“ (*module_2_std_pos, module_2_std_neg*), „Sequence Length Distribution“ (*module_7_alpha, module_7_beta*) und „Sequence Duplication Levels“ (*module_8_alpha, module_8_beta*). Dies deutet darauf hin, dass einige dieser Features in späteren Iterationen entfernt werden können.

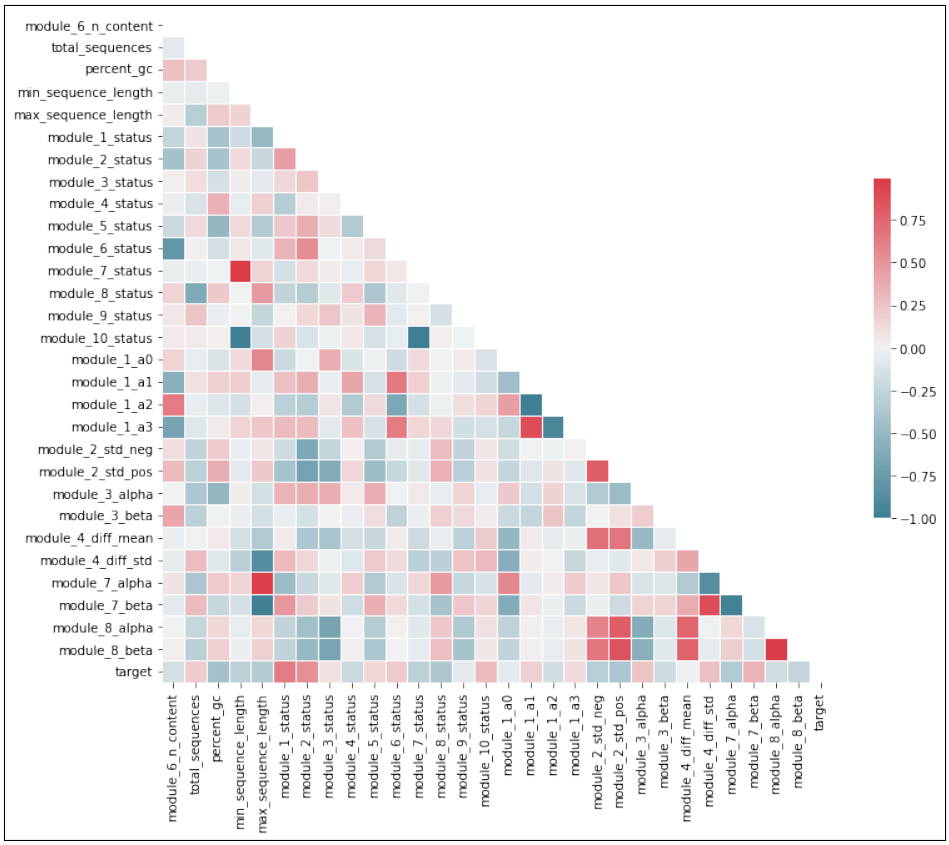

Abbildung 5.1: Heatmap der Feature-Korrelation (erstellt mit [8])

5.2 Ergebnisse der ML-Modelle

Da bereits das Training von ML-Modellen auf Grundlage der einfachen Features in Projektteil A zu sehr guten Ergebnissen geführt hat, können die Kennzahlen der Kreuzvalidierungen nur begrenzt Auskunft über eine Verbesserung des Modells geben. Dafür müssten die Unterschiede der Ergebnisse mit einem größeren und komplexeren Datensatz analysiert werden. Solche Daten könnten ohne weitere Anpassungen gleichermaßen mit der erstellten Pipeline zum Training von ML-Modellen und deren Evaluierung verwendet werden. Nichtsdestotrotz können, durch die Informationen des Decision Trees und der Random Forest Feature Importance, auch von diesem Datensatz Erkenntnisse gewonnen werden, welche Features vielversprechend umgesetzt wurden und welche unbedeutend für die Zielfragestellung sind. Im Folgenden werden die Kennzahlen der Kreuzvalidierung ($k = 10$) für die Modelle, welche durch den gesamten Datensatz trainiert worden sind, dargestellt und mit den äquivalenten Ergebnissen aus Teil A des Projekts verglichen. Abschließend werden die erreichten Kennzahlen für die Bakterientyp spezifischen ML-Modelle diskutiert.

5.2.1 Support Vector Machine Ergebnisse

Die Support Vector Machine (SVM) konnte durch die Erweiterung mit den neuen Features keine nennenswerte Verbesserung erzielen. Die Tabelle 5.1 zeigt die resultierenden Kennzahlen im Vergleich mit denen aus Projektteil A (in Klammern dargestellt).

	Accuracy	Precision	Recall	F_1-Score	AUC
Durchschnitt	0.69 (0.68)	0.68 (0.67)	0.97 (0.95)	0.79 (0.78)	0.6 (0.74)
Standardabweichung	0.12 (0.14)	0.12 (0.11)	0.08 (0.14)	0.07 (0.10)	-

Tabelle 5.1: Ergebnisse der SVM Kreuzvalidierung (gesamter Datensatz) mit den Werten aus Projektteil A in Klammern

5.2.2 Decision Tree Ergebnisse

Für den Decision Tree ist mit der erhöhten Dimensionalität eine leichte Verschlechterung der Performance festzustellen. Tabelle 5.2 zeigt die Kennzahlen im Vergleich mit denen aus Projektteil A in Klammern.

	Accuracy	Precision	Recall	F_1-Score	AUC
Durchschnitt	0.89 (0.93)	0.89 (0.93)	0.95 (0.98)	0.91 (0.95)	0.94 (0.96)
Standardabweichung	0.09 (0.08)	0.10 (0.11)	0.09 (0.05)	0.07 (0.06)	-

Tabelle 5.2: Ergebnisse der Decision Tree Kreuzvalidierung (gesamter Datensatz) mit den Werten aus Projektteil A in Klammern

Der in Abbildung 5.2 dargestellte Decision Tree zeigt jedoch, dass in der Klassifizierung durch dieses Modell auch einige der neuen Features von Bedeutung waren. So sind nur drei von 10 Feature Tests die benötigt werden, um vollständig reine Blätter zu erreichen, Status-Features aus Projektteil A. Die anderen sieben wurden aus den Features gewählt, die in diesem Projekt

erstellt wurden. Am einflussreichsten war dabei *module_2_std_neg*, welches als Wurzelknoten und direkt darauf als zweiter Knoten die Daten an zwei verschiedenen Werten besonders gut voneinander teilen konnte. Dies relativiert die ernüchternden Ergebnisse der Kennzahlen, da gezeigt werden kann, dass die kreierten Features gleichwertig fähig sind, die Klassenunterschiede zu identifizeren.

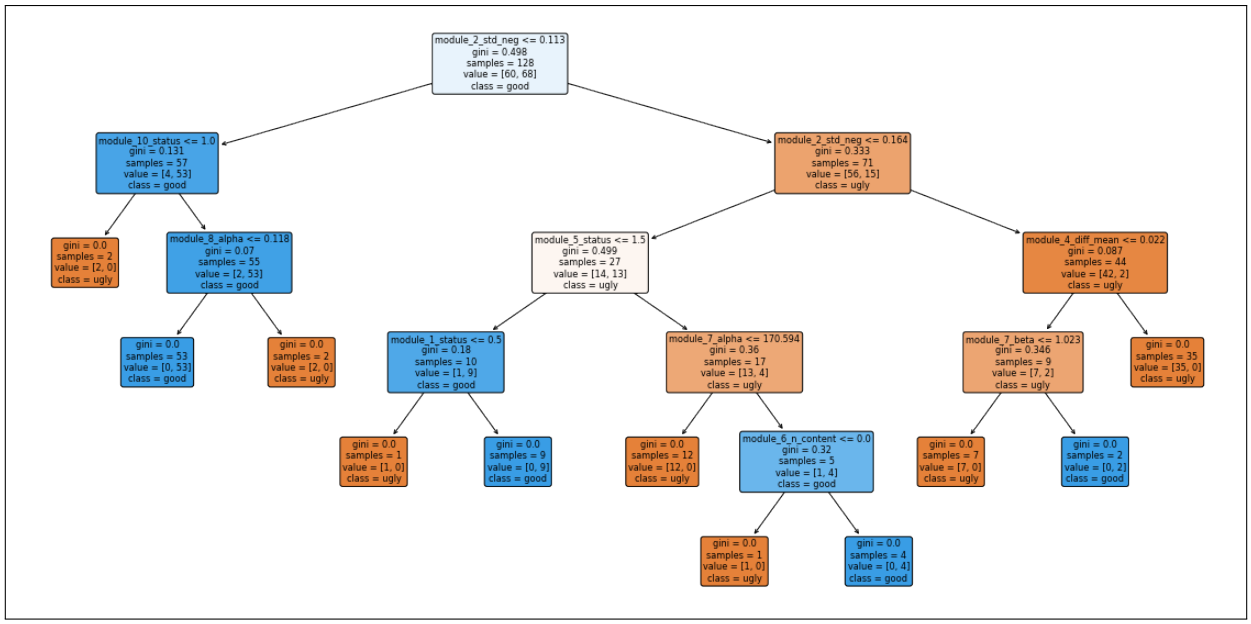

Abbildung 5.2: Visualisierung des Entscheidungsbaums (erstellt mit [9])

5.2.3 Random Forest Ergebnisse

Auch bei den Ergebnissen des Random Forest Modells sind leichte Verschlechterungen in der Kreuzvalidierung festzustellen, dargestellt in Tabelle 5.3.

	Accuracy	Precision	Recall	F_1-Score	AUC
Durchschnitt	0.93 (0.96)	0.93 (0.96)	0.96 (0.98)	0.94 (0.97)	1.0 (1.0)
Standardabweichung	0.08 (0.06)	0.09 (0.07)	0.08 (0.05)	0.06 (0.05)	-

Tabelle 5.3: Ergebnisse der Random Forest Kreuzvalidierung (gesamter Datensatz) mit den Werten aus Projektteil A in Klammern

Die Feature Importance des trainierten Random Forest Modells, dargestellt in Abbildung 5.3, gibt einen aufschlussreichen Einblick in die Klassifikationsfähigkeit der konstruierten Features. So sind die beiden Features des Moduls „Per Tile Sequence Quality“ diejenigen mit der größten Teilungsfähigkeit der Klassen im Random Forest. Auch viele andere der neuen Features sind von Bedeutung. So sind die sieben Features mit dem geringsten Einfluss alle Features aus dem Projektteil A. Von besonderem Interesse ist auch die Analyse, welche Module von ihren, in diesem Projekt erstellten, Features bedeutsamer sind als die Status dieser Module. Das trifft auf die Module „Per Tile Sequence Quality“, „Sequence Duplication Levels“, „Per Sequence Quality Scores“, „Per Base Sequence Content“ und „Sequence Length Distribution“ zu. Daraus kann

geschlussfolgert werden, dass die Repräsentation dieser Module durch die konstruierten Features relevanter für die Klassifizierung der Qualität ist als deren FastQC-Statusbewertung.

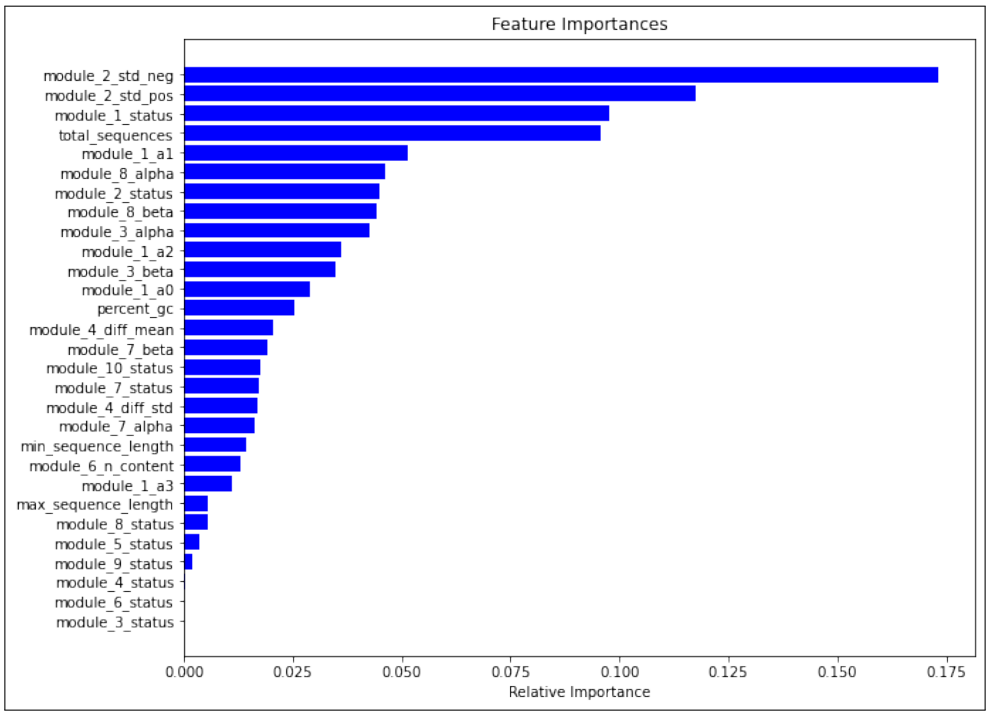

Abbildung 5.3: Feature Importance des Random Forests (erstellt mit [6])

5.2.4 ML-Modelle je Organismus

Die Implementierung der ML-Algorithmen sollte auch für eine Separierung der Bakterienarten Enterococcus faecium (E. faecium), Escherichia coli (E. coli) und Staphylococcus aureus (S. aureus) in eigene ML-Modelle untersucht werden. Für E. coli lagen zu wenig Daten vor, um eine sinnvolle Kreuzvalidierung durchzuführen. Die Angaben hierzu sind nur zur Vollständigkeit aufgeführt und sollten nicht verwendet werden, um Schlussfolgerungen zu ziehen. Die Tabelle 5.4 zeigt die Kreuzvalidierungsergebnisse aller untersuchten Modelle und die äquivalenten Ergebnisse aus Teil A des Projekts in Klammer dahinter.

Die einzige Verbesserung gegenüber den Werten des Projektteils A konnte im SVM-Modell des E. faecium festgestellt werden, alle anderen weisen eine etwas schlechtere Performance auf.

Weiterhin erweisen sich die ML-Modelle für einzelne Organismen als besser geeignet für eine Klassifizierung der Qualitäten als die Modelle der gesamten Daten, da sie sich durch spezialisiertere Gewichtung der Features möglichen Eigenheiten des DNA-Typs besser anpassen können.

Modell	Accuracy	Precision	Recall	F_1-Score	AUC
Gesamt (SVM)	0.69 (0.68)	0.68 (0.67)	0.97 (0.95)	0.79 (0.78)	0.6 (0.74)
E. faecium (SVM)	0.78 (0.86)	0.81 (0.86)	0.92 (0.98)	0.82 (0.91)	0.65 (0.6)
S. aureus (SVM)	0.81 (0.62)	0.83 (0.73)	0.93 (0.62)	0.87 (0.63)	0.69 (0.69)
E. coli (SVM)*	1.0 (1.0)*	0.2 (0.4)*	0.2 (0.4)*	0.2 (0.4)*	-
Gesamt (Decision Tree)	0.89 (0.93)	0.89 (0.93)	0.95 (0.98)	0.91 (0.95)	0.94 (0.96)
E. faecium (Decision Tree)	0.91 (0.98)	0.94 (1.0)	0.92 (0.96)	0.92 (0.97)	0.95 (1.0)
S. aureus (Decision Tree)	0.95 (0.96)	0.98 (0.97)	0.94 (0.98)	0.96 (0.97)	0.97 (1.0)
E. coli (Decision Tree)*	0.9 (1.0)*	0.1 (0.2)*	0.1 (0.2)*	0.1 (0.2)*	-
Gesamt (Random Forest)	0.93 (0.96)	0.93 (0.96)	0.96 (0.98)	0.94 (0.97)	1.0 (1.0)
E. faecium (Random Forest)	0.95 (1.0)	0.95 (1.0)	0.98 (1.0)	0.96 (1.0)	(1.0)
S. aureus (Random Forest)	0.96 (0.99)	0.97 (1.0)	0.98 (0.98)	0.97 (0.99)	0.92 (1.0)
E. coli (Random Forest)*	0.9 (1.0)*	0.0 (0.2)*	0.0 (0.2)*	0.0 (0.2)*	-

Tabelle 5.4: Vergleich der Ergebnisse aller ML-Modelle
mit den Werten aus Projektteil A in Klammern

5.3 Zusammenfassung

Im Teil B des Projekts konnte dargestellt werden, wie die einzelnen FastQC-Module durch Feature Engineering so repräsentiert werden können, dass sie sich für das Training von ML-Modellen zur Klassifizierung der Qualität von Sequenzierungsdaten eignen. Dabei wurde auch gezeigt, dass ein Großteil dieser Features in der Lage sind, die Zielklassen besser voneinander zu unterscheiden als die Status der FastQC-Module.

Die Gesamtperformance der ML-Modelle fiel etwas schlechter aus als bei der Implementation der Features aus Teil A des Projekts. Dies kann durch ein ungünstiges Verhältnis der Anzahl der Features zu der Anzahl der Daten begründet werden. Nichtsdestotrotz zeigt die Analyse der Modelle, dass eine weitere Exploration komplexerer Features von Nutzen sein kann. Insbesondere sollten dafür größere Datensätzen verwendet werden, damit die Bedeutung der einzelnen Features genauer herausgearbeitet werden kann.

Das Random Forest Modell zeigte die beste Performance und wurde deshalb in einer Pipeline zum Training von ML-Modellen für die Qualitätseinschätzung von FASTQ-Dateien implementiert. Darüberhinaus wurde ein Nextflow Prozess implementiert, der dieses Modell verwenden kann, um die Qualität neuer FASTQ-Dateien zu evaluieren. Dieser kann in bestehende Pipelines zur Analyse von Sequenzierungsdaten eingebaut werden und damit Biologen unterstützen die Qualität von Sequenzierungen zu bewerten.

Literatur

[1] Simon Andrews. *Babraham Bioinformatics - FastQC Documentation*. URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ (besucht am 30. 03. 2022).

[2] Paolo Di Tommaso u. a. „Nextflow enables reproducible computational workflows“. In: *Nature Biotechnology* 35.4 (Apr. 2017), S. 316–319. DOI: 10.1038/nbt.3820. URL: https://nextflow.io.

[3] Pablo Duboue. *The Art of Feature Engineering: Essentials for Machine Learning*. 1. Aufl. Cambridge University Press, 2020.

[4] Richard J. Rossi. *Mathematical statistics: An introduction to likelihood based inference*. 1. Aufl. Wiley, 2018.

[5] Müller Andreas C. und Sarah Guido. *Introduction to machine learning with python: A guide for data scientists*. 2. Aufl. O'Reilly Media, Inc., 2017.

[6] 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.

[7] Pauli Virtanen u. a. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“. In: *Nature Methods* 17 (2020), S. 261–272. DOI: 10.1038/s41592-019-0686-2.

[8] 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.

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