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

Independent Coursework 1

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 Zielstellung

Die vorliegende Arbeit dokumentiert die Entwicklung einer Lösung zur Exploration des Einsatzes von Machine Learning (ML) zur Evaluierung der Qualität von Next-Generation-Sequencing-Daten. Dafür sollen zunächst gegebene FASTQ-Dateien durch die Software FastQC [1] analysiert werden. Die Ergebnisse müssen mit Metadaten der gegebenen Dateien verknüpft und in geeigneten Features zusammengefasst werden. Diese sollen dann zum Training verschiedener Machine Learning Modelle eingesetzt werden, damit untersucht werden kann, welches am besten für die Aufgabenstellung geeignet ist.

1.2 Entwicklungsumgebung

Die Implementierung wurde auf einem Linux-System (Pop!_OS 21.10, 64-bit) umgesetzt. Die Entwicklung wurde mit Python 3 in Jupyter Notebooks durchgeführt [2]. Angegebene Ausführungszeiten in dieser Arbeit beziehen sich auf den im Entwicklungssystem vorliegenden Intel® Core™i7-7700HQ Prozessor.

Zur Versionsverwaltung wurde Git verwendet und durch ein Remote-Repository auf GitHub gesichert und Stakeholdern zur Verfügung gestellt. Dies kann unter folgender Adresse abgerufen werden: `https://github.com/AlexHTW/ngs_quality_control`.

1.3 Software Voraussetzungen

Für die Analyse der FASTQ Dateien wird die Software FastQC (0.11.9) verwendet [1]. Diese kann von der Webseite heruntergeladen werden und ist nach dem Entpacken einsatzbereit. FastQC setzt eine Java Runtime Environment voraus. Im Projekt wird openjdk (11.0.14.1) verwendet.

Die Programmierung wird mit Python (3.9.7) umgesetzt. Die benötigten Pakete sind in der Tabelle 1.1 aufgelistet. Sie können über pip installiert werden ('`pip install PAKET==VERSION`'). Pip kann mit '`sudo apt install python3-pip`' installiert werden.

Paket	Version	Paket	Version
notebook	6.4.0	matplotlib	3.5.1
regex	2022.3.15	fastqcparser	1.1
numpy	1.22.1	scikit-learn	1.0.2
pandas	1.4.1	seaborn	0.11.2

Tabelle 1.1: Benötigte Python Pakete

2 Datenextraktion

2.1 Vorliegende Daten

Für die vorliegende Untersuchung wurden Sequenzierungsdaten von Bakterienproben vom Robert Koch-Institut bereitgestellt. Diese liegen als komprimierte FASTQ Dateien (.fastq.gz) vor. Insgesamt handelt es sich dabei um 184 Dateien, bestehend aus 88 Enterococcus faecium (E. faecium), 18 Escherichia coli (E. coli) und 78 Staphylococcus aureus (S. aureus) DNA Sequenzierungen. Die Dateien haben eine Gesamtgröße von 37,7 Gigabyte, wobei die Größe einzelner Dateien im Bereich von 96,4 Kilobyte bis 1,5 Gigabyte liegt.

2.2 Metadaten

Die Dateien sind in einer Verzeichnisstruktur angelegt, die weitere Informationen über die Sequenzierungsdaten enthalten. Die erste Ebene gibt Aufschluss über den Organismus, bezeichnet als „Sau“ (für S. auereus), „Efcm“ (für E. faecium) oder „Ecoli“ (für E. coli). Die zweite Ebene enthält die Qualitätseinschätzung der Sequenzierungen, bezeichnet als „good“ (für gute Qualität) oder „ugly“ (für schlechte Qualität). Die dritte Ebene beinhaltet das verwendete Illumina Sequenzierungs-System - „MS“ (für MiSeq-Systeme), „HS“ (für HiSeq-Systeme) oder „NX“ (für NextSeq-Systeme) [3]. In der vierten Ebene befinden sich schließlich die entsprechenden Sequenzierungs-Dateien. Der Dateiname beinhaltet eine Angabe über die Sequenzierungsrichtung bei Paired-End-Reads, R1 für vorwärts und R2 für rückwärts [3]. Die Verzeichnisstruktur lässt sich folgendermaßen zusammenfassen:

`Organismus/Qualität/Sequenzierungs-System/Sequenzierung_Richtung.fastq.gz`.

Diese Informationen werden im Weiteren zusammenfassend als Metadaten bezeichnet. Sie sind für die Analyse der Daten und das Training der ML-Modelle relevant und müssen mit den Inhalten der FASTQ-Dateien verknüpft werden. Dafür werden die Metadaten im ersten Notebook *01_extract_metadata.ipynb* ausgelesen und in einem Pandas Dataframe gespeichert [4]. Dabei enthält jede Zeile den Dateinamen einer Sequenzierungs-Datei und die entsprechenden Metadaten. Abbildung 2.1 zeigt die ersten fünf Zeilen des entstandenen Dataframes.

Das Dataframe wird als JSON-Datei exportiert, um in anderen Notebooks oder Skripten weiterverwendet werden zu können.

	filename	organism	technology	read_number	evaluation
0	200709_20-07968_20-00891_S21_L000_R2_001	Sau	MS	2	ugly
1	200204_20-00746_19-03927_S9_L000_R1_001	Sau	MS	1	ugly
2	181002-18-6991-775-18_S1_L001_R1_001	Sau	MS	1	ugly
3	200327_20-04028_20-00328_S25_L000_R1_001	Sau	MS	1	ugly
4	180727-18-5425-18-01680_S8_L001_R2_001	Sau	MS	2	ugly

Abbildung 2.1: Dataframe der Metadaten

2.3 FASTQ-Dateien

Die Grundlage der Extraktion von Features für die ML-Modelle sollen die Ergebnisse einer FastQC-Analyse der FASTQ-Dateien bilden. Daher wird FastQC zunächst für alle Dateien ausgeführt und die Ergebnisse in einem Unterordner gespeichert.

Mit dem Befehl `'fastqc -o fastqc_results --extract *'` können alle FASTQ-Dateien in einem Verzeichnis durch FastQC analysiert und die Ergebnisse in das Zielverzeichnis *fastqc_results* extrahiert werden.

Nachdem dieser Schritt für alle Verzeichnisse durchgeführt wurde, befinden sich die Ergebnisse einer Sequenzierung in der Datei `fastqc_results/DATEINAME/fastqc_data.txt`.

Die Ergebnisdateien aller Sequenzierungen werden daraufhin im Notebook *03_fastqc_data_extraction_and_merge.ipynb* unter Einsatz des Python Pakets *fastqcparser* importiert. Die erhaltenen Daten werden so umgeformt, dass der gesamte Datensatz in einem Dataframe dargestellt wird, wobei jeder Read eine Zeile ist und die entsprechenden FastQC-Modulinhalte in den Spalten hinterlegt werden. Die Modulinhalte sind wiederum Dataframes, sodass insgesamt ein geschachteltes Dataframe entsteht. Darüberhinaus werden die Status aller Module als Liste in einer zusätzlichen Spalte hinzugefügt. Über den gemeinsamen Schlüssel des Dateinamens kann anschließend das Dataframe der Metadaten hinzugefügt werden. Damit enthält das Dataframe alle für die weitere Verarbeitung benötigten Daten und wird als JSON-Datei exportiert. Die Tabelle 2.1 zeigt die vorliegenden Spalten und deren Datentypen für jeden Read.

Bezeichnung	Dateityp
Basic Statistics	Pandas Dataframe
Per base sequence quality	Pandas Dataframe
Per tile sequence quality	Pandas Dataframe
Per sequence quality scores	Pandas Dataframe
Per base sequence content	Pandas Dataframe
Per sequence GC content	Pandas Dataframe
Per base N content	Pandas Dataframe
Sequence Length Distribution	Pandas Dataframe
Sequence Duplication Levels	Pandas Dataframe
Overrepresented sequences	Pandas Dataframe
Adapter Content	Pandas Dataframe
Module Statuses	Liste
organism	String
technology	String
read_number	Integer
evaluation	String

Tabelle 2.1: Inhalt des vollständigen Dataframes

3 Datenaufbereitung

3.1 Datenexploration

Um die sinnvollste Weiterverarbeitung der Daten zu ermitteln, soll zunächst ein Überblick über die aggregierten Daten gewonnen werden, dargestellt im Notebook *04_data_exploration.ipynb*.

Das Modul „Basic Statistics“ enthält folgende Informationen zur Sequenzierung: Dateiname, Dateityp, Codierung, Sequenzen mit schlechter Qualität, Bereich der Sequenzlängen und GC-Anteil.

Die darauf folgenden Module enthalten Tabellen unterschiedlicher Form, welche die einzelnen Base Call Qualitäten oder Basenanteile unter verschiedenen Gesichtspunkten zusammenfassen.

Die Spalte für Modul Status enthält die von FastQC ermittelten Status der einzelnen Module („pass“, „warn“ oder „fail“) entsprechend der Reihenfolge der vorhergehenden Spalten. Der erste Eintrag jeder Liste, also der Status für das Modul „Basic Statistics“, ist stets „pass“, kann also in den folgenden Betrachtungen weggelassen werden.

Insgesamt liegen in den gegebenen Daten 110 Reads guter Qualität und 74 Reads schlechter Qualität vor. Die Sequenzierungs-Systeme werden als *technology* bezeichnet. Im Datensatz befinden sich 80 Sequenzierungen von MiSeq-Systemen, 50 Sequenzierungen von HiSeq-Systemen und 54 Sequenzierungen von NextSeq-Systemen. Diese sollen im vorliegenden Projekt jedoch nicht weiter untersucht werden. Ebenso wird die Sequenzierungsrichtung (*read_number*) in dieser Arbeit nicht weiter verarbeitet.

3.2 Features und Label

Im folgenden Verarbeitungsschritt, implementiert in dem Notebook *05_data_preparation_ml.ipynb*, werden die Informationen ausgewählt, welche als Features für die ML-Modelle verwendet werden können und in eine geeignete numerische Repräsentation überführt.

Aus dem Modul „Basic Statistics“ werden dafür vier Werte extrahiert: Anzahl der Sequenzen (*total_sequences*), GC-Anteil (*percent_gc*), kürzeste Sequenzlänge (*min_sequence_length*) und längste Sequenzlänge (*max_sequence_length*).

Darüber hinaus sollen die verbleibenden Module in diesem Teil des Projekts nur durch ihren Status repräsentiert werden. Dafür werden sie zunächst in numerische Form überführt. Da sie eine eindeutige Wertigkeit zueinander aufweisen, können sie durch aufeinander folgende Integer repräsentiert werden: „fail“ durch *0*, „warn“ durch *1* und „pass“ durch *2*. Daraufhin werden sie aus der inneren Liste der Spalte „Module Statuses“ in durchnummerierte Spalten überführt. Beginnend mit *module_1_status* für den Status des Moduls „Per Base Sequence Quality“ bis hin zu *module_10_status* für den Status des Moduls „Adapter Content“, entsprechend der Reihenfolge aus Tabelle 2.1.

Ebenso müssen die Label für die ML-Modelle in Zahlenform überführt werden. Die Label sind die Qualitätsbewertungen in der Spalte *evaluation*. Hierbei wird „ugly“ als *0* und „good“ als *1* dargestellt.

4 Machine Learning

4.1 Implementierung der Machine Learning Algorithmen

Der aus der Datenaufbereitung resultierende Datensatz verfügt über die folgenden Features: *total_sequences, percent_gc, min_sequence_length, max_sequence_length, module_1_status, module_2_status, module_3_status, module_4_status, module_5_status, module_6_status, module_7_status, module_8_status, module_9_status* und *module_10_status*. Das Label ist *evaluation*. Der Eintrag der Spalte *organism* wird nicht als Feature verwendet, er dient dazu den Datensatz nach Organismus trennen zu können, um eigene ML-Modelle für jeden Organismus trainieren und evaluieren zu können. Die folgenden Modelle werden für den gesamten Datensatz in dem Notebook *06_ngs_quality_machine_learning-all-reads.ipynb* implementiert. Das Notebook kann ohne Veränderung auf die Teildatensätze je Organismus angewendet werden. Dafür muss nur der Datenimport ausgetauscht werden. Im Repository liegen die speziellen Ausführungen und deren Ergebnisse in den folgenden Notebooks vor:

- *06_ngs_quality_machine_learning-ecoli.ipynb*
- *06_ngs_quality_machine_learning-efcm.ipynb*
- *06_ngs_quality_machine_learning-sau.ipynb*.

Die Daten werden für das Training und Testen gesplittet, wobei die Größe des Testdatensatzes 30% beträgt. Dadurch lassen sich die ROC-Kurven für die Ergebnisse darstellen und die gewählten Kennzahlen berechnen. Aufgrund der geringen Anzahl an Daten wird für eine zuverlässigere Berechnung der Kennzahlen auch eine 10-fache Kreuzvalidierung für jedes Modell durchgeführt.

Unter Einsatz der Python Bibliothek scikit-learn werden die ML-Modelle Support Vector Machine (SVM), Decision Tree und Random Forest trainiert und evaluiert. Die SVM wurde mit einem linearen Kernel implementiert. Der Random Forest wurde mit 100 Decision Trees umgesetzt.

4.2 Ergebnisse

Die detaillierten Ergebnisse der Kreuzvalidierungen können den entsprechenden Notebooks oder der wissenschaftlichen Arbeit des Forschungsprojekts Teil A entnommen werden. Zusammenfassend ist festzustellen, dass die SVM eine wesentlich schlechtere Performance erzielt als der Decision Tree und der Random Forest. Dies gilt sowohl für den gesamten Datensatz als auch für die Teildatensätze. Die Verwendung der Teildatensätze konnte durch die Spezialisierung für einen Organismus bessere Ergebnisse erzielen als der Gesamtdatensatz.

Auch bei der Laufzeit des Trainings der Modelle sind der Decision Tree und der Random Forest der SVM deutlich überlegen. Die Tabelle 4.1 zeigt die Laufzeiten des Trainings eines Modells, mit 70% der Gesamtdaten, ohne Kreuzvalidierung.

Das Training der SVM ist um ein 40.000-faches langsamer als das Training eines Decision Trees.

Modell	Laufzeit
Support Vector Machine	42,8s
Decision Tree	1,06ms
Random Forest	91ms

Tabelle 4.1: Laufzeiten des Modell-Trainings

5 Zusammenfassung

5.1 Entstandene Implementierung

Die Anforderungen des Projekts wurden in acht Jupyter Notebooks implementiert. Die Tabelle 5.1 listet die Bezeichnungen der Notebooks im Repository und die entstandenen Ergebnisse auf. Resultierende Datenexporte sind kursiv gekennzeichnet. Die Exporte sind im Verzeichnis *notebooks/exported_datasets* zu finden und ermöglichen die Ausführung der Notebooks (außer der ersten beiden) ohne über die umfangreichen Rohdaten der Sequenzierungen verfügen zu müssen.

Notebook Bezeichnung	Ergebnisse
01_extract_metadata.ipynb	Extraktion der Metadaten *metadata.json*
03_fastqc_data_extraction_and_merge.ipynb	Import der FastQC Ergebnisse *complete_set.json*
04_data_exploration.ipynb	Einblick in den Datensatz
05_data_preparation_ml.ipynb	Datenaufbereitung für die ML-Modelle *all_simple.json*, *efcm_simple.json*, *sau_simple.json*, *ecoli_simple.json*
06_ngs_quality_machine_learning-all-reads.ipynb	Kreuzvalidierungen der Modelle des vollständigen Datensatzes und Decision Tree Visualisierung
06_ngs_quality_machine_learning-ecoli.ipynb	Kreuzvalidierungen der Modelle des E. coli-Datensatzes und Decision Tree Visualisierung
06_ngs_quality_machine_learning-efcm.ipynb	Kreuzvalidierungen der Modelle des E. faecium-Datensatzes und Decision Tree Visualisierung
06_ngs_quality_machine_learning-sau.ipynb	Kreuzvalidierungen der Modelle des S. aureus-Datensatzes und Decision Tree Visualisierung

Tabelle 5.1: Entstandene Implementierung

5.2 Fazit und Ausblick

Die Zielstellung des Projekts konnte erfolgreich umgesetzt werden. ML-Modelle können für die Evaluierung der Qualität von Next-Generation-Sequencing-Daten eingesetzt werden. Der Einsatz von FastQC zur vorhergehenden Qualitätsanalyse der Daten ermöglicht eine effiziente Zusammenfassung der umfangreichen Sequenzierungsdaten und eine unkomplizierte Extraktion relevanter Features. Die gewählten Features der wesentlichen Statistiken und Modul-Status repräsentieren jedoch nur einen kleinen Teil des Informationsgehalts der FastQC-Analyse. Eine eingehendere Exploration der einzelnen Modul-Inhalte könnte die Performance der ML-Modelle verbessern.

Eine Feinabstimmung der ML-Modelle, wie zum Beispiel die Untersuchung anderer Kernels für die SVM oder anderer Hyperparameter für die Erstellung des Random Forest, wurde nicht umgesetzt. Dies sollte für einen umfangreicheren Datensatz durchgeführt werden, der eine ausgewogenere Repräsentation der Sequenzierungsdaten aus dem tatsächlichen Einsatzfeld darstellt. Nur dadurch kann sichergestellt werden, dass ermittelte Hyperparameter für die ML-Modelle relevant sind.

Die Implementierung wurde in Jupyter Notebooks durchgeführt um die Ergebnisse dieses Ansatzes zu untersuchen und darzustellen. Für einen Einsatz in bioinformatischen Pipelines sollte der Lösungsweg in Python Modulen und Skripten implementiert werden.

Literatur

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

[2] Thomas Kluyver u. a. „Jupyter Notebooks – a publishing format for reproducible computational workflows“. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Hrsg. von F. Loizides und B. Schmidt. IOS Press. 2016, S. 87–90.

[3] Illumina Inc. *An introduction to Next-Generation Sequencing Technology*. URL: https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf (besucht am 18. 04. 2022).

[4] The pandas development team. *pandas-dev/pandas: Pandas*. Version 1.4.1. Feb. 2022. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.