NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

RICARDO DUARTE CARDOSO BERNARDINO

BSc in Computer Science and Engineering

MACHINE LEARNING FOR ANOMALY DETECTION IN DNA SANGER SEQUENCING

APPLIED TO 96-WELL PLATE DATA IN HIGH-THROUGHPUT GENOMIC WORKFLOWS

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon

Draft: *September 1, 2025*

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MACHINE LEARNING FOR ANOMALY DETECTION IN DNA SANGER SEQUENCING

APPLIED TO 96-WELL PLATE DATA IN HIGH-THROUGHPUT GENOMIC WORKFLOWS

RICARDO DUARTE CARDOSO BERNARDINO

BSc in Computer Science and Engineering

Adviser: Orfeu Flores
CEO, STAB VIDA

Co-advisers: Carmen Morgado
Assistant Professor, NOVA University Lisbon

André Lamúrias
Assistant Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon

Draft: *September 1, 2025*

Abstract

Since the dawn of mankind, our pursuit to understand the world has driven innovation and shaped the course of scientific discovery. Among the most profound achievements of this quest is the field of genomics, which has revolutionized our understanding of life itself. By decoding the genetic blueprint of organisms, genomics has unlocked insights into health, disease, evolution, and biodiversity, paving the way for transformative advancements in medicine, agriculture, and biotechnology.

Sanger sequencing remains a cornerstone of DNA analysis, yet its alignment results often necessitate labor-intensive manual verification to identify anomalies and inaccuracies. This process is not only time-consuming but also susceptible to human error, presenting significant challenges for laboratories handling high-throughput sequencing data. Addressing these limitations is critical for improving the efficiency and reliability of genomic workflows.

This master's thesis introduces an AI-driven tool designed to automate the verification of DNA sequence alignment results generated by the ABI3730xl sequencer. Leveraging machine learning models trained on real-world data from 96-well plates, the tool identifies anomalies, detects sequence homologies, and evaluates the acceptability of results with high precision. To ensure accessibility, the tool is deployed on an online platform, enabling remote usage by laboratories worldwide.

The proposed solution significantly enhances laboratory workflows by streamlining the verification process and reducing reliance on manual oversight. This not only minimizes errors but also accelerates data processing, supporting advancements in precision genomics and biotechnology. Additionally, the tool's capability to detect inaccurately labeled wells further highlights its practical utility in high-throughput DNA analysis.

By addressing critical limitations in current practices, this work demonstrates the transformative potential of integrating artificial intelligence into genomic research. The tool contributes to the field by enabling more efficient and accurate DNA sequencing, paving the way for future innovations in genomics and biotechnology.

Keywords: Sanger Sequencing, Artificial Intelligence, DNA Verification, Machine Learning, Genomic Workflows, High-Throughput Sequencing, Automation, Precision Genomics

Resumo

Desde os primórdios da humanidade, a nossa busca para compreender o mundo tem impulsionado a inovação e moldado o curso da descoberta científica. Entre os feitos mais profundos desta busca está o campo da genómica, que revolucionou a nossa compreensão da vida em si. Ao decifrar o código genético dos organismos, a genómica revelou insights sobre saúde, doença, evolução e biodiversidade, abrindo caminho para avanços transformadores na medicina, agricultura e biotecnologia.

O sequenciamento Sanger é um método fundamental na análise de DNA, mas os seus resultados de alinhamento exigem frequentemente uma verificação manual, um processo moroso para identificar anomalias e imprecisões. Este processo não só consome tempo, como também está sujeito a erros humanos, representando um desafio significativo para laboratórios que lidam com dados de sequenciação de alto débito. Superar estas limitações é essencial para melhorar a eficiência e a fiabilidade dos fluxos de trabalho genómicos.

Esta tese de mestrado apresenta uma ferramenta baseada em inteligência artificial (IA) concebida para automatizar a verificação dos resultados de alinhamento de sequências de DNA gerados pelo sequenciador ABI3730xl. Utilizando modelos de aprendizagem automática treinados com dados reais de placas de 96 poços, a ferramenta identifica anomalias, deteta homologias de sequências e avalia a aceitabilidade dos resultados com elevada precisão. Para garantir acessibilidade, a ferramenta foi implementada numa plataforma online, permitindo a sua utilização remota por laboratórios em todo o mundo.

A solução proposta melhora significativamente os fluxos de trabalho laboratoriais ao agilizar o processo de verificação e reduzir a dependência de supervisão manual. Isto não só minimiza erros, como também acelera o processamento de dados, contribuindo para avanços na genómica de precisão e na biotecnologia. Além disso, a capacidade da ferramenta para identificar poços incorretamente rotulados reforça a sua utilidade prática na análise de DNA de alto débito.

Ao abordar limitações críticas das práticas atuais, este trabalho demonstra o potencial transformador da integração da inteligência artificial na investigação genómica. A ferramenta contribui para o campo ao possibilitar uma sequenciação de DNA mais eficiente e

precisa, abrindo caminho para futuras inovações na genómica e na biotecnologia.

Palavras-chave: Sequenciação Sanger, Inteligência Artificial, Verificação de DNA, Aprendizagem Automática, Fluxos de Trabalho Genómicos, Sequenciação de Alto Débito, Automação, Genómica de Precisão

Contents

List of Figures viii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Context . . . 2
  - 1.2.1 Introduction to STAB VIDA . . . 2
  - 1.2.2 Challenges in the Current Workflow . . . 3
  - 1.2.3 The Need for Automation at STAB VIDA . . . 3
  - 1.2.4 AI-Powered Solutions for Sequencing Workflows . . . 4
- 1.3 Objectives . . . 5
- 1.4 Structure . . . 5

2 State of the Art & Related Work 7
- 2.1 Genomics . . . 7
  - 2.1.1 Introduction to DNA . . . 8
  - 2.1.2 DNA Sequencing and Sanger Sequencing . . . 9
- 2.2 Artificial Intelligence . . . 10
  - 2.2.1 Data Science . . . 11
  - 2.2.2 Machine Learning and Deep Learning . . . 12
  - 2.2.3 Considerations for Model Selection & Development . . . 15
- 2.3 Related Work . . . 19
  - 2.3.1 Isolation Forest: Unsupervised Anomaly Detection . . . 19
  - 2.3.2 DBSCAN: Density-Based Clustering and Anomaly Detection . . . 20
  - 2.3.3 Comparison with Thesis Approach . . . 21
  - 2.3.4 Other Relevant Algorithms . . . 21
  - 2.3.5 Conclusion . . . 22

3 Proposal 23
- 3.1 Laboratory Processes . . . 23
  - 3.1.1 Sample Preparation and Processing . . . 23

3.1.2 The ABI3730xl DNA Sequencer . . . 25
3.1.3 Data Generation & Collection . . . 26
3.1.4 Manual Analysis . . . 27
3.2 Initial Proposed Architecture . . . 28
3.2.1 Database Structure . . . 29
3.2.2 Data Preprocessing (Feature Engineering) . . . 30
3.2.3 Model Selection & Development . . . 32
3.2.4 Evaluation (Planned) . . . 34
3.2.5 Frameworks, Languages & Tools . . . 34
3.3 Limitations of the Proposal (Why We Pivoted) . . . 35

4 Implementation **36**
4.1 Design Evolution at a Glance . . . 36
4.2 Co-Design with the Laboratory (Requirements Drift) . . . 37
4.3 System Overview and Orchestration . . . 37
4.4 From Proposal to Implementation: Stepwise Evolution . . . 38
4.5 Data Access & Collection (Implemented) . . . 38
4.6 Data and Final Per-Well Schema . . . 41
4.7 Preprocessing . . . 41
4.7.1 Quality Trimming and Ambiguity Handling . . . 41
4.7.2 Minimum Length and Short-Read Path . . . 41
4.7.3 Sequence Representation and Single-Base Sensitivity . . . 41
4.7.4 Plate-Local Similarity and Clustering Signals . . . 42
4.7.5 Curation Notes . . . 42
4.7.6 Per-Step Inputs and Outputs (I/O Contracts) . . . 42
4.8 Per-Plate Conv1D Autoencoder . . . 43
4.9 Classifier and Evaluation Protocol . . . 43
4.9.1 Retired Configurations . . . 43
4.9.2 Tabular Classifier . . . 44
4.9.3 Train/Test Split and Leakage Control . . . 44
4.9.4 Feature Engineering and Leakage Mitigation . . . 44
4.10 Post-hoc Rule Layer . . . 44
4.10.1 Rule inventory (excerpt) . . . 45
4.10.2 Fusion With Model Probability . . . 45
4.11 Limitations and Failure Modes . . . 45
4.12 Reproducibility . . . 45

5 Validation & Evaluation **47**
5.1 Experimental Setup . . . 47
5.2 Validation Protocol . . . 47
5.3 Metrics . . . 48

5.4 Evidence for Design Choices 48
5.4.1 Step 1: BLAST-like neighbourhoods vs. DBSCAN@k-mer 48
5.4.2 Step 2: Utility of AE latents 48
5.4.3 Step 3: Classifier comparison and leakage control 49
5.4.4 Step 4: Rule fusion 49
5.5 Overall Results 49
5.6 Per-Plate Results 49
5.6.1 `January_30_2025_p6` 49
5.6.2 `January_31_2025_p1` 49
5.6.3 `March_12_2025_p1_siso` 49
5.6.4 `March_13_2025_p4_siso` 50
5.7 Ablation Study 50
5.8 Error Analysis and Interpretability 50
5.9 Robustness Checks 50
5.10 Runtime & Throughput 50
5.11 Summary 50

6 Planning **51**
6.1 Timeline & Milestones 51

Bibliography **52**

Appendices

Annexes

List of Figures

2.1 Objectives of Human Genome Project: The Human Genome Project was an international scientific endeavor with the aim of finding the base pairs that constitute human DNA as well as of identifying, mapping, and sequencing every gene in the human genome from both a physical and functional perspective [68]. 8
2.2 The structure of the DNA double helix. 8
2.3 Steps involved in Sanger sequencing [63]. 10
2.4 Understanding what Machine Learning and Deep Learning are, as subsets of Artificial Intelligence [57]. 12
2.5 Illustration of the main types of Machine Learning. 13
2.6 A deep neural network consists of an input layer, multiple hidden layers and an output layer, all consisting of nodes [61]. 14
2.7 A node in a hidden layer of a deep neural network takes an input, performs a calculation, and passes on an output to nodes in the next layer [61]. 14
2.8 Each neural node in a deep neural network performs a calculation by taking inputs from nodes in the previous layer, multiplying these by weights, and summing all outcomes. The resulting number gets processed by an activation function to allow for more complex calculations. Lastly, the output is passed on to the nodes in the next layer [61]. 14
2.9 The Optimum Regression compared to Underfitting and Overfitting the prediction. [2]. 17
2.10 (a) Graphical illustration of bias and variance. The aim is to obtain a model that has low bias and low variance. (b) Contribution of bias and variance to the total error of the machine learning model (reproduced from the reference [57]). With increasing model complexity, error function reduced significantly for training data set but perform poorly in validation dataset, i.e., higher variance. The task to define a model in such a way that the bias and variance can tackle in validation and blind test data set. [51]. 18

3.1 A 96-well PCR plate. 24

3.2 Structure and Organization of each PCR plate. 24
3.3 A closed ABI3730xl capillary electrophoresis DNA analyzer. 25
3.4 An opened ABI3730xl capillary electrophoresis DNA analyzer. 25
3.5 Visualization of noise in the beginning of the DNA sequence, using 4peaks. 31
3.6 Visualization of noise in the end of the DNA sequence, using 4peaks. 31

6.1 Gantt Chart of the Project Timeline 51

1

Introduction

In this initial chapter, the motivation for automating critical aspects of the Sanger sequencing pipeline is explored in detail. We discuss the operational challenges faced by laboratory personnel in manually verifying DNA sequencing results, the potential for artificial intelligence (AI) to alleviate these challenges, and the overall significance of introducing automated solutions in the context of genomic research.

The chapter also establishes the context in which this thesis was developed, explaining the collaboration between the company and the university to address specific problems in sequencing workflows. The importance of minimizing errors, optimizing throughput, and improving accuracy in genomic sequencing is emphasized, setting the stage for the problem addressed in this thesis.

Following the context, the defined objectives of the thesis are outlined, detailing the deliverables and results expected from this research. Finally, a brief overview of the thesis document's structure is provided to guide the reader through the subsequent chapters, ensuring a clear understanding of the overall thesis plan.

1.1 Motivation

DNA sequencing has become a cornerstone of modern science, driving advancements in fields such as medicine, agriculture, and evolutionary biology. The ability to decode genetic information has transformed our understanding of life, enabling breakthroughs in personalized medicine, disease diagnosis, and the development of targeted therapies [27]. In agriculture, sequencing has facilitated the engineering of crops with improved yield, resilience, and nutritional value, addressing critical challenges such as food security [76]. Similarly, in evolutionary biology, sequencing has provided unprecedented insights into the genetic diversity and history of species [69].

Despite its transformative potential, DNA sequencing faces significant challenges. The process of generating and interpreting sequencing data is complex, labor-intensive, and prone to human error. Sanger sequencing, a widely used method known for its accuracy, involves multiple manual steps that can introduce inefficiencies and inaccuracies

into the workflow. Laboratory personnel must meticulously verify sequencing results, a task that is both time-consuming and susceptible to oversight. These challenges are exacerbated by the growing demand for high-throughput sequencing, which places additional pressure on laboratory resources and increases the risk of errors [10].

The consequences of these inefficiencies are far-reaching. In medicine, for example, errors in sequencing can lead to misdiagnoses or inappropriate treatments, potentially jeopardizing patient outcomes. In agricultural biotechnology, inaccuracies in sequencing data can delay the development of improved crop varieties, hindering efforts to address global food security. Similarly, in evolutionary biology, errors in sequencing can distort our understanding of genetic relationships and evolutionary processes [46].

Addressing these challenges is critical to unlocking the full potential of DNA sequencing. By improving the accuracy, efficiency, and scalability of sequencing workflows, we can accelerate scientific discoveries and enhance the practical applications of genomics. Emerging technologies, such as artificial intelligence (AI), offer promising avenues for addressing these challenges. AI has demonstrated its potential to automate complex tasks, reduce human error, and enhance decision-making across various domains, including healthcare and biotechnology [19, 67]. Furthermore, artificial intelligence (AI) exhibits significant potential in domains characterized by large-scale datasets, particularly when such data contains discernible patterns and diverse variations. It's the ability of extracing meaningful insights from complex and heterogeneous data that enhances AI's value in addressing intricate challenges across various fields. In the context of DNA sequencing, AI-driven solutions could streamline the verification of sequencing results, enabling faster and more reliable analyses.

1.2 Context

Given that AI has non-stop increased relevance in the market, many companies seek to leverage it's capabilities for specific services, not only creating market share, but also providing competition, innovation, and bolstering the scienfitic advancement of the field. This thesis was conducted within the context of a collaboration between STAB VIDA, a leading company in genomic sequencing, and the university. The partnership aims to address specific inefficiencies in Sanger sequencing workflows, with a focus on improving accuracy, scalability, and operational efficiency. By combining academic research with industry expertise, this collaboration seeks to explore and implement existing AI mechanisms that can transform the field of genomics.

1.2.1 Introduction to STAB VIDA

STAB VIDA is a dedicated company in the field of genomic sequencing, specializing in high-quality services for DNA and RNA analysis. With a strong focus on precision and reliability, STAB VIDA offers state-of-the-art sequencing technologies, including Sanger

sequencing, which remains one of the most trusted methods for accurate DNA analysis [65]. The company caters to a wide range of clients, from academic researchers to pharmaceutical and agricultural industries, providing critical data for projects related to personalized medicine, evolutionary biology, and biotechnological innovations [23].

1.2.2 Challenges in the Current Workflow

Despite the advanced technology used by STAB VIDA, the manual processes associated with DNA sequencing present significant challenges. While the company's clients provide the DNA samples to be sequenced, the preparation of these samples often requires additional human intervention before the sequencing process can begin. Depending on the type of service requested, some samples must be mixed with specific primers, while others require adjustments to their concentration levels. These preparatory steps, though critical for ensuring successful sequencing, introduce opportunities for human error that can compromise the quality of the final results. For example, samples with high concentration levels can contaminate neighboring samples during the sequencing process. This contamination not only affects the reliability of the sequencing results but also necessitates repeat analyses, further delaying the workflow and increasing costs. These manual interventions exacerbate the challenges faced by laboratory personnel. In addition to these preparatory steps, the verification of DNA sequencing results in 96-well plates is another error-prone and time-consuming task. Laboratory personnel must manually inspect each well for abnormalities, such as abnormal signal scores, cross-well contamination, or unexpected DNA sequences that could indicate potential errors in the sequencing process [9]. These manual checks, while essential for ensuring accurate and high-quality sequencing results, introduce bottlenecks into the workflow. With the increasing demand for rapid sequencing services, the time and effort required for manual verification grow proportionally, raising the likelihood of human error [3]. Such errors can compromise the reliability of sequencing results delivered to clients, impacting both client satisfaction and the company's reputation [45].

The combination of manual sample preparation and post-sequencing verification creates a workflow that is not only labor-intensive but also vulnerable to inconsistencies. Addressing these challenges is critical to improving the accuracy, efficiency, and scalability of STAB VIDA's sequencing services.

1.2.3 The Need for Automation at STAB VIDA

Given the increasing complexity and volume of sequencing tasks, STAB VIDA recognizes the urgent need to automate certain aspects of the workflow. Specifically, the company seeks to leverage artificial intelligence to enhance the detection of anomalies in sequencing data [38]. AI has the potential to significantly improve the speed and accuracy of this process by automatically flagging wells that exhibit unusual patterns.

By automating the flagging process, STAB VIDA aims to:

- **Reduce manual labor**: Laboratory personnel will be freed from repetitive tasks and be able to focus on higher-order tasks that require human expertise, such as resolving flagged anomalies [82].
- **Increase accuracy and throughput**: AI-driven anomaly detection will reduce human error, improving the overall accuracy of sequencing results delivered to the client and enabling faster turnaround times [11].
- **Improve scalability**: Automation will allow STAB VIDA to handle larger volumes of sequencing data, facilitating growth and meeting the increasing demand for sequencing services [62].

The collaboration between STAB VIDA and the university has thus emerged as a critical step in addressing these challenges. This thesis, which focuses on the study and application of existing AI tools for anomaly detection in sequencing data, aligns directly with STAB VIDA's goals of operational efficiency and enhanced service quality [70].

1.2.4 AI-Powered Solutions for Sequencing Workflows

The implemented solution leverages existing AI mechanisms to integrate seamlessly into STAB VIDA's existing sequencing workflows. By analyzing sequencing data from the company's ABI3730xl sequencers, the AI tool automatically flags wells that deviate from expected patterns based on specific criteria, such as signal quality or sequence consistency [25]. These flagged wells will then be quarantined for human review, where laboratory personnel can investigate and resolve any issues that might arise.

Key benefits of this AI-driven solution include:

- **Automated anomaly detection**: The AI tool will automatically identify and flag wells exhibiting abnormalities, such as contamination, poor signal quality, or unusual DNA sequences [81].
- **Detailed reporting**: For each flagged well, the system will provide detailed explanations and reasons for the anomaly, enabling more efficient review and resolution [83].
- **Integration with existing workflows**: The tool will be designed to work within STAB VIDA's current sequencing infrastructure, ensuring minimal disruption to ongoing operations and enabling smooth adoption by laboratory staff [12].

This thesis will contribute to STAB VIDA's mission to improve sequencing workflows, reduce operational inefficiencies, and enhance the quality and reliability of its sequencing services. By focusing on the study and implementation of existing AI mechanisms, the proposed solution has the potential to set a new standard for accuracy and efficiency in genomic sequencing.

1.3 Objectives

The primary objective of this thesis is to develop an AI-powered tool that automates the verification of alignment results in Sanger sequencing. This tool will address inefficiencies in the manual processes currently employed and enhance the overall reliability of sequencing workflows. Specific objectives include:

- **Data Exploration and Preprocessing**: Use data science techniques to explore, determine, and select the relevant data for model training, ensuring the data is preprocessed effectively.
- **Model Investigation**: Research an already existing robust AI algorithm fit and capable of detecting anomalies in the sequencing data, including signal inconsistencies, contamination, and unexpected DNA sequences.
- **Model Training and Validation**: Train and validate the AI algorithm using real-world datasets obtained from ABI3730xl sequencers, ensuring its applicability in practical laboratory settings.
- **Performance Evaluation**: Compare the AI tool's accuracy and efficiency against traditional manual verification methods, demonstrating its effectiveness in a real-world context.
- **Deployment Strategy**: Deploy the AI tool, ensuring seamless integration into existing laboratory workflows and maximizing user adoption.

By achieving these objectives, this research will not only improve the efficiency of STAB VIDA's sequencing workflows but also contribute to the broader adoption of AI in genomics, paving the way for future innovations in the field. *Note on scope.* The implemented system is a local pipeline integrated with the laboratory's workflow. A networked service wrapper (e.g., web API) is considered future work rather than part of the deployed artifact.

1.4 Structure

This document is organized into six main chapters, each focusing on a critical aspect of the thesis:

- **Chapter 1 – Introduction**: Motivation, context, objectives, and structure.
- **Chapter 2 – State of the Art & Related Work**: Key background in genomics and AI; related approaches to clustering and anomaly detection.
- **Chapter 3 – Proposal**: The original (baseline) design explored at project start and its assumptions.

- **Chapter 4 – Implementation**: Describes the actual development of the final pipeline, including orchestration, preprocessing, modeling, and rule fusion.
- **Chapter 5 – Validation & Evaluation**: Experimental setup, metrics, baselines, ablations, and per-plate case studies.
- **Chapter 6 – Conclusion & Future Work**: Summary of contributions, limitations, and directions ahead.

The Proposal (Chapter 3) is treated as a baseline for comparison; the final system is specified in Chapter 4 and validated in Chapter 5. Each chapter is designed to guide the reader through the research journey, from identifying the problem to proposing and validating a solution. This structured approach ensures clarity and coherence, making the thesis accessible to both technical and non-technical audiences. The following chapter will delve into the state of the art, reviewing existing literature and technologies relevant to this research.

2

State of the Art & Related Work

This chapter provides a comprehensive review of the current state of the art in genomics, emphasizing the transformative role of artificial intelligence (AI) in modern DNA sequencing and analysis. It begins with an introduction to genomics as a scientific discipline, exploring the structure, function, and evolution of genomes, along with the foundational advancements enabled by projects such as the Human Genome Project.

The chapter delves into DNA sequencing techniques, with a primary focus on Sanger sequencing, highlighting its historical significance, methodology, and key components.

Core concepts of machine learning and deep learning are introduced in latter sections, emphasizing their application in automating and enhancing genomic analysis tasks. Important topics such as overfitting, bias-variance tradeoff, and model optimization are thoroughly discussed, reflecting the relevance of these challenges in the context of AI-driven genomics research. This chapter aims to establish a clear understanding of the foundational and emerging methodologies that define the field today.

2.1 Genomics

Genomics is the study of the structure, function, and evolution of genomes—the complete set of DNA within an organism. It encompasses various aspects, including gene identification, genome sequencing, and the analysis of genetic interactions. The field has advanced rapidly since the completion of the Human Genome Project, which mapped the entire human genome, enabling breakthroughs in personalized medicine, evolutionary biology, and disease research [47, 78, 35].

Modern genomics leverages high-throughput sequencing technologies, such as next-generation sequencing (NGS), which generate vast amounts of data at unprecedented speeds. These technologies, combined with computational tools, have transformed our ability to explore genetic information on a massive scale. Genomics not only provides insights into fundamental biological processes but also underpins critical applications in biotechnology, agriculture, and healthcare [55, 52].

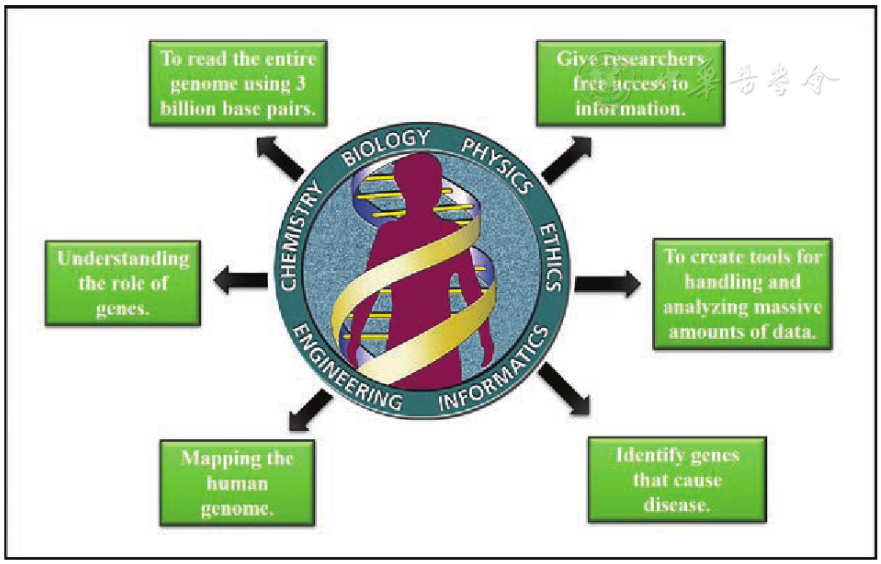

Figure 2.1: Objectives of Human Genome Project: The Human Genome Project was an international scientific endeavor with the aim of finding the base pairs that constitute human DNA as well as of identifying, mapping, and sequencing every gene in the human genome from both a physical and functional perspective [68].

2.1.1 Introduction to DNA

DNA (deoxyribonucleic acid) is the molecular blueprint for all living organisms, carrying the genetic instructions necessary for growth, development, and reproduction. The structure of DNA was famously elucidated by Watson and Crick in 1953, a discovery that revolutionized molecular biology [80]. DNA consists of two complementary strands coiled into a double helix, with nucleotide bases (adenine, thymine, cytosine, and guanine) forming the rungs of the helix [1].

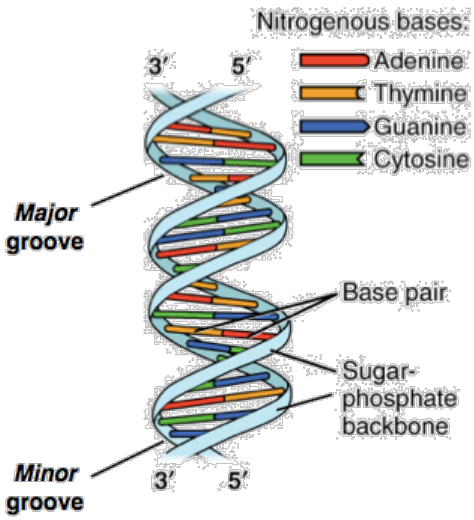

Figure 2.2: The structure of the DNA double helix.

2.1.2 DNA Sequencing and Sanger Sequencing

DNA sequencing is the process of determining the precise order of nucleotides within a DNA molecule. Among the sequencing methods, Sanger sequencing, developed by Frederick Sanger in 1977, has long been the gold standard due to its high accuracy and reliability [66]. This method, also known as the chain termination method, involves the selective incorporation of chain-terminating dideoxynucleotides (ddNTPs) during DNA replication, resulting in DNA fragments of varying lengths that can be analyzed to determine the DNA sequence [54].

While Sanger sequencing remains indispensable, next-generation sequencing (NGS) technologies have emerged in recent years. These technologies offer higher throughput and faster sequencing capabilities, enabling the sequencing of entire genomes quickly and cost-effectively [55]. However, Sanger sequencing retains its significance for specific applications such as sequencing small, targeted regions of the genome, validating NGS results, and scenarios where longer read lengths are advantageous [52].

Despite advancements in NGS, Sanger sequencing continues to be favored for its simplicity, cost-effectiveness in low-throughput projects, and unparalleled accuracy. Its enduring role in genomic research and clinical diagnostics highlights its importance, particularly for applications requiring meticulous validation of genetic data [69].

This section explores the principles and methodologies of Sanger sequencing, emphasizing its relevance and applications in modern genomic studies. Sequencing DNA is fundamental to understanding genetic information, enabling applications ranging from disease diagnosis and treatment to evolutionary biology [53]. By deciphering the nucleotide order, scientists can identify mutations, study gene expression, and uncover genetic diversity.

To better understand the components involved in the Sanger Sequencing process:

- **DNA Template**: The single-stranded DNA to be sequenced.
- **Primer**: A short single-stranded oligonucleotide complementary to the target sequence, serving as the starting point for DNA synthesis.
- **DNA Polymerase**: An enzyme that synthesizes new DNA strands by adding nucleotides to the primer.
- **Deoxynucleotides (dNTPs)**: Normal nucleotides (A, T, C, G) used to elongate the DNA strand.
- **Dideoxynucleotides (ddNTPs)**: Modified nucleotides (ddATP, ddTTP, ddCTP, ddGTP) that terminate DNA strand synthesis upon incorporation due to the lack of a 3′-OH group.

The steps of Sanger sequencing are enumerated as follows and illustrated in Figure 2.3:

1. **Primer annealing and chain extension**: DNA polymerase synthesizes complementary DNA strands using a primer and a mixture of deoxynucleotides (dNTPs).
2. **ddNTP binding and chain termination**: The incorporation of ddNTPs halts DNA synthesis at specific nucleotides, resulting in DNA fragments of varying lengths.
3. **Fluorescent labeling of DNA fragments**: The terminated DNA strands are fluorescently labeled for detection.
4. **Capillary gel electrophoresis and fluorescence detection**: DNA fragments are separated by capillary gel electrophoresis and identified based on fluorescence signals.
5. **Sequence analysis and reconstruction**: The nucleotide sequence is reconstructed from the fluorescence data and analyzed.

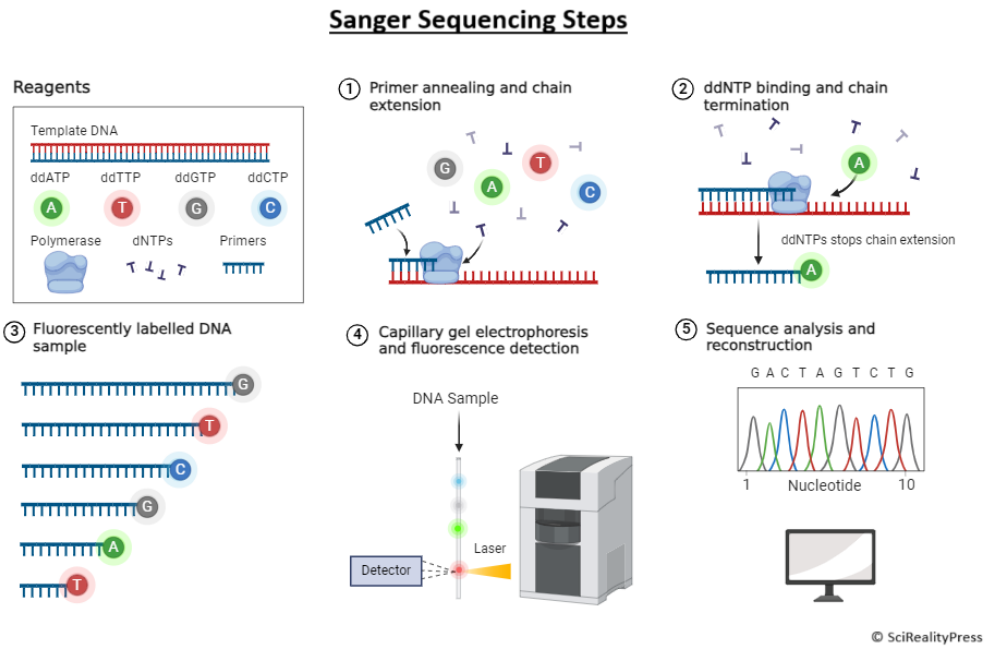

Figure 2.3: Steps involved in Sanger sequencing [63].

2.2 Artificial Intelligence

Artificial intelligence (AI) represents a paradigm shift in how data is analyzed and interpreted across multiple fields, including genomics. AI encompasses a broad range of computational techniques aimed at simulating human intelligence, including learning, reasoning, and self-correction [64]. These capabilities enable AI systems to process and analyze vast amounts of data, uncover hidden patterns, and make informed predictions. The journey of AI began with early algorithms that could perform basic logical reasoning. Over time, advancements in computational power and data availability led to the development of sophisticated methods like **Machine Learning (ML)** and **Deep Learning (DL)**,

which are now at the forefront of AI innovation. The development of AI has been greatly accelerated by advancements in computational power, data availability, and algorithmic improvements [8].

2.2.1 Data Science

Data is the cornerstone of achieving optimal model performance in Artificial Intelligence (AI). The effectiveness of AI models hinges on the quality and relevance of the data used for training. These models learn from the data provided, and their ability to generate accurate responses is directly tied to the insights they extract during this process. In the context of Data Science, this underscores the critical importance of data preparation, including cleaning, transformation, and feature engineering, as these steps lay the foundation for robust and reliable AI systems [26, 60]. Data Science is an interdisciplinary field that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from structured and unstructured data [58]. It serves as the foundation for advanced technologies like Machine Learning (ML) and Deep Learning (DL), offering essential tools and methodologies for data processing and analysis [17]. The importance of Data Science lies in its ability to bridge the gap between raw data and actionable insights. By leveraging techniques such as data cleaning, transformation, and visualization, it enables the identification of patterns, trends, and correlations that would otherwise remain hidden [36]. This process lays the groundwork for training ML and DL models by ensuring that the data is accurate, complete, and relevant.

The key components of Data Science include:

- **Data Collection:** Gathering data from diverse sources, such as experiments, databases, or web scraping [30].
- **Data Cleaning:** Addressing issues like missing values, duplicates, and inconsistencies to ensure data quality [44].
- **Exploratory Data Analysis (EDA):** Using statistical techniques and visualization tools to understand the underlying structure of the data [75].
- **Feature Engineering:** Transforming raw data into meaningful features that enhance the performance of ML models [28].
- **Model Evaluation:** Assessing the performance of predictive models using metrics like accuracy, precision, and recall [21].

Data Science is a crucial step for ML and DL because the quality and relevance of data directly impact model performance. Poor data quality can lead to inaccurate predictions, while well-prepared data ensures that models can learn effectively and generate reliable results [5].

2.2.2 Machine Learning and Deep Learning

One of the key approaches within AI is Machine Learning (ML), which has become central to modern AI development. A more specialized area within ML is Deep Learning, which utilizes neural networks with multiple layers to model and solve highly complex problems. Deep Learning is considered a subset of ML, focused on using deep neural networks to automatically learn hierarchical representations of data.

The overall structure of AI can be thought of as a hierarchy, where AI is the overarching field that encompasses various approaches, with ML being a significant subset. Within ML, Deep Learning stands as a specialized technique designed to handle large, unstructured data by leveraging multi-layered networks. This structure allows each level to address different aspects of problem-solving, with Deep Learning excelling in tasks that require high levels of abstraction and complex pattern recognition.

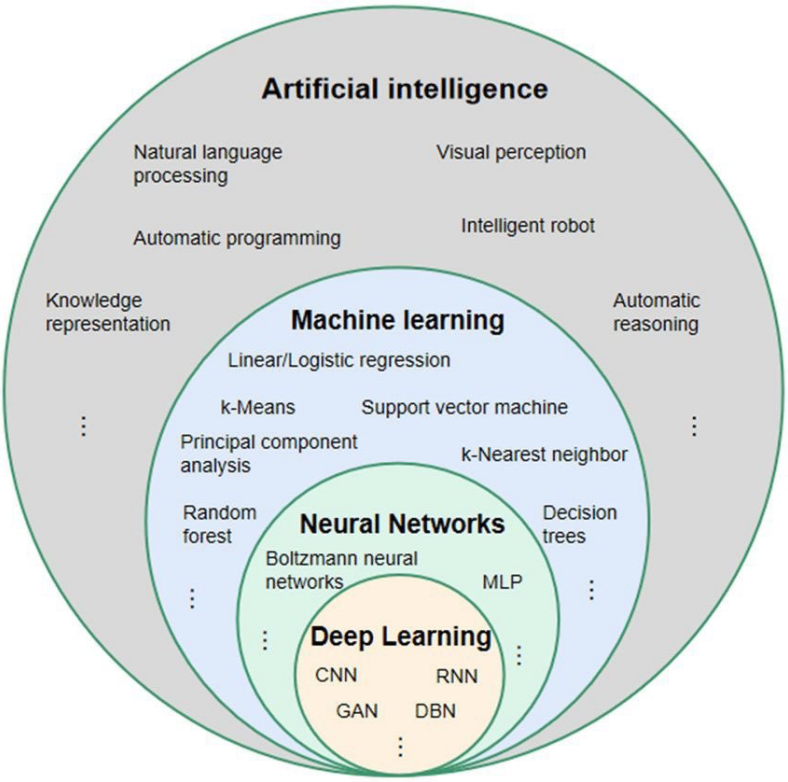

Figure 2.4: Understanding what Machine Learning and Deep Learning are, as subsets of Artificial Intelligence [57].

2.2.2.1 Machine Learning

Machine Learning: Machine Learning (ML) is a subset of AI focused on developing algorithms that enable systems to learn from data and make decisions without explicit programming. In ML, algorithms are trained to recognize patterns and infer insights from large datasets [5]. ML techniques are typically divided into three main categories based on the nature of the training data:

- **Supervised Learning**: The algorithm is trained on labeled data, where the input data is paired with the correct output. The goal is to learn a mapping from inputs to outputs, as seen in classification or regression tasks [56]. **Semi-supervised Learning** uses both labeled and unlabeled data samples

- **Unsupervised Learning**: This method involves training on data that has no labels. The algorithm tries to identify hidden patterns or structures, such as grouping similar data points into clusters [36].

- **Reinforcement Learning**: The algorithm learns by interacting with its environment and receiving feedback through rewards or penalties. It is particularly useful in dynamic environments where decisions have long-term consequences [74].

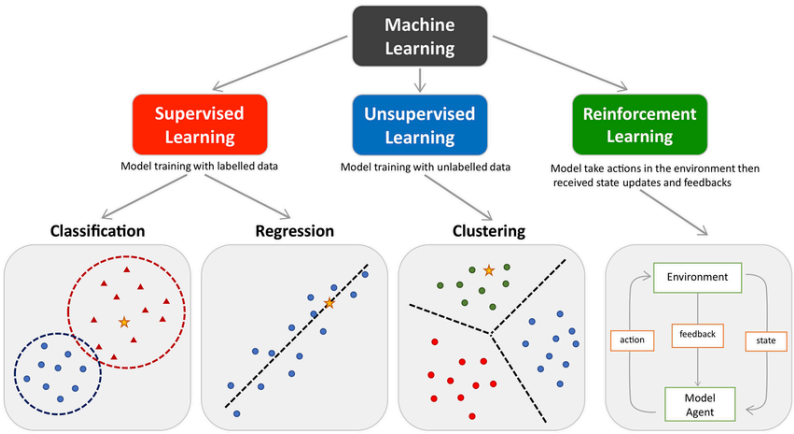

Figure 2.5: Illustration of the main types of Machine Learning.

2.2.2.2 Deep Learning

Deep Learning: Deep Learning, a specialized area of machine learning, leverages artificial neural networks with multiple layers (hence "deep") to model and solve highly complex problems. By simulating the way humans learn from data, deep learning excels in identifying patterns, handling large datasets, and making predictions with high accuracy. Key architectures in deep learning include:

- **Convolutional Neural Networks (CNNs):** CNNs are particularly effective for analyzing data with a spatial structure, such as images or signals. They have found applications in areas like object detection, image classification, and medical imaging [49, 48].

- **Recurrent Neural Networks (RNNs):** RNNs are designed to handle sequential data, making them well-suited for tasks like natural language processing and time series analysis. Advanced variants, such as Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), address challenges like vanishing gradients in longer sequences [34, 14].

- **Transformers:** Transformers revolutionized sequential data processing by introducing self-attention mechanisms, allowing the model to consider relationships between all elements in a sequence simultaneously. Transformers underpin models like BERT and GPT, which have demonstrated state-of-the-art performance in natural language understanding and generation [77, 16].

- **Autoencoders:** Autoencoders are unsupervised learning models that focus on efficient data encoding. They are widely used for dimensionality reduction, anomaly detection, and data denoising [33, 79].

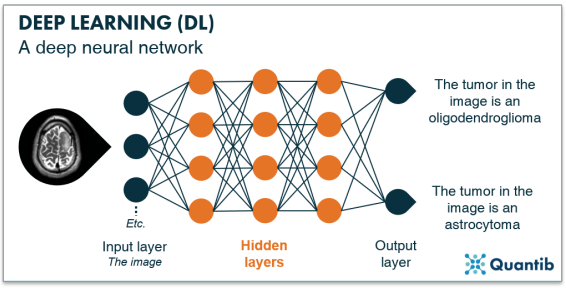

Figure 2.6: A deep neural network consists of an input layer, multiple hidden layers and an output layer, all consisting of nodes [61].

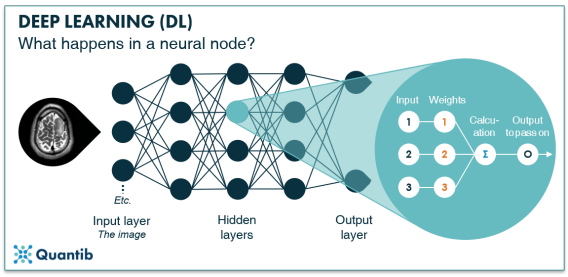

Figure 2.7: A node in a hidden layer of a deep neural network takes an input, performs a calculation, and passes on an output to nodes in the next layer [61].

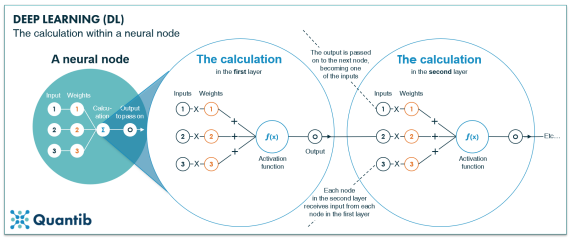

Figure 2.8: Each neural node in a deep neural network performs a calculation by taking inputs from nodes in the previous layer, multiplying these by weights, and summing all outcomes. The resulting number gets processed by an activation function to allow for more complex calculations. Lastly, the output is passed on to the nodes in the next layer [61].

2.2.3 Considerations for Model Selection & Development

Despite the vast array of deep learning techniques available, the choice of model should always align with the problem's context. Not all tasks require complex, deep networks; in fact, for simpler problems, simpler models may be more effective.

2.2.3.1 Overfitting

Overfitting is a common issue in machine learning, occurring when a model learns not only the underlying patterns in the training data but also the noise. This results in poor generalization to new, unseen data. Overfitting is particularly problematic when using overly complex models for relatively simple tasks.

For example, handwritten digit recognition tasks (such as the MNIST dataset) can be effectively solved using shallow neural networks or simpler models like Support Vector Machines (SVMs). Using a deep architecture, such as a Convolutional Neural Network (CNN), which is more suitable for complex datasets like ImageNet, would add unnecessary complexity and increase the risk of overfitting [31, 32].

To understand overfitting, it is essential to differentiate between training error and test error.

Training Error refers to the error a model makes on the data it was trained on. A model that memorizes the training set perfectly will have a very low training error, but this does not necessarily mean it will perform well on new data.

Test Error , on the other hand, measures the model's performance on unseen data. A well-generalized model maintains a balance between training error and test error. However, an overfitted model will exhibit a large gap between the two: it performs exceptionally well on training data but poorly on test data [24, 5].

Causes of Overfitting Overfitting is not solely caused by model complexity. Several factors contribute to this issue:

- **Model Selection:** Choosing a model that is too complex for the given dataset can lead to overfitting. For instance, using a deep neural network for a simple classification problem may result in unnecessary memorization rather than pattern recognition [17].

- **Insufficient Training Data:** When the dataset is too small, the model may not capture the true distribution of the data and instead learn noise as patterns. Expanding the dataset through data augmentation, synthetic data generation, or collecting more samples can mitigate this issue [71].

- **Lack of Regularization:** Regularization techniques such as L1/L2 regularization (ridge and lasso regression), dropout in neural networks, and early stopping help prevent overfitting by constraining the model's complexity [73].

- **Training for Too Many Epochs:** Over-training a model without early stopping can cause it to learn noise from the training data, reducing generalization to new examples [59].

- **High Feature Dimensionality:** If a model has too many features relative to the number of training samples, it may fit noise instead of meaningful relationships. Feature selection and dimensionality reduction techniques, such as Principal Component Analysis (PCA), can help mitigate this risk [39].

Mitigating Overfitting To reduce overfitting and improve generalization, various strategies can be employed:

- **Cross-validation:** Splitting data into multiple folds for training and validation helps estimate a model's performance on unseen data [43].

- **Early stopping:** Monitoring validation loss and stopping training before overfitting occurs prevents unnecessary memorization of noise [59].

- **Regularization:** Techniques like dropout, weight decay, and batch normalization help prevent overfitting by introducing constraints [73].

- **Data augmentation:** Applying transformations such as rotation, scaling, and flipping to training samples increases dataset diversity and improves model robustness [71].

- **Reducing model complexity:** Choosing a simpler model that still captures essential patterns prevents unnecessary parameterization [17].

By understanding and addressing the causes of overfitting, models can be optimized to balance training and test performance, ensuring their reliability in real-world applications.

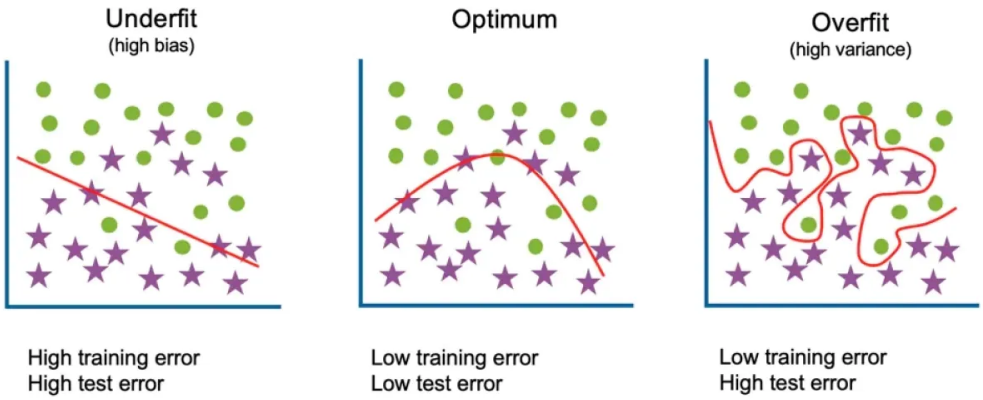

Figure 2.9: The Optimum Regression compared to Underfitting and Overfitting the prediction. [2].

2.2.3.2 Balancing Bias and Variance

Bias and variance are two fundamental sources of error in machine learning models, and understanding their trade-off is crucial for building models that generalize well to new data [24].

Bias: refers to the error introduced by approximating a real-world problem with a simplified model. A model with high bias makes strong assumptions about the data and may fail to capture the underlying patterns, leading to underfitting [22]. This means that the model performs poorly on both training and test data. For example, using a linear regression model to fit a dataset with a highly nonlinear relationship would result in high bias. The model would fail to capture the complexity of the data, leading to systematic errors in its predictions [22].

Variance: refers to the model's sensitivity to small fluctuations in the training data. A model with high variance captures noise in the training set instead of just the underlying patterns, leading to overfitting [5]. This results in good performance on training data but poor generalization to unseen data. For instance, a deep neural network with a large number of parameters trained on a small dataset may memorize the training data but fail to generalize. The model's high variance means it is too dependent on the specifics of the training data, leading to poor performance on unseen samples [5].

The Trade-Offs: The trade-off between bias and variance is fundamental when selecting a model. Simpler models, such as linear regression or decision trees, generally exhibit higher bias but lower variance. These models are often preferable for tasks with limited data or when interpretability is critical. On the other hand, more complex models like Gradient Boosted Machines (GBMs) or neural networks tend to have lower bias but higher variance. Regularization techniques are often employed to control overfitting in these

cases [5, 17]. The goal of machine learning is to find a balance between bias and variance to minimize total error. A model with high bias and low variance tends to underfit, while a model with low bias and high variance tends to overfit. The optimal trade-off ensures that the model has just the right complexity to generalize well [17]. Regularization techniques, cross-validation, and careful model selection help manage this trade-off effectively.

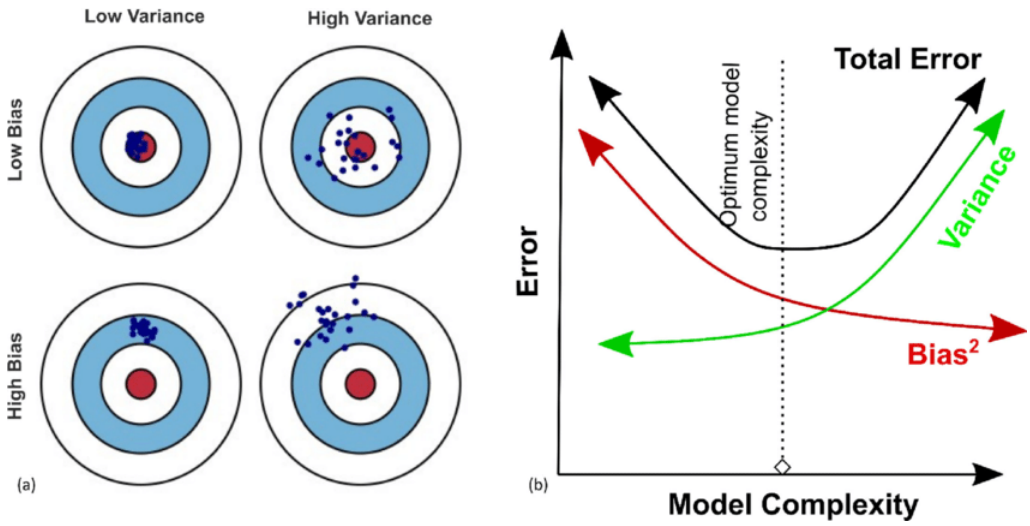

Figure 2.10: (a) Graphical illustration of bias and variance. The aim is to obtain a model that has low bias and low variance. (b) Contribution of bias and variance to the total error of the machine learning model (reproduced from the reference [57]). With increasing model complexity, error function reduced significantly for training data set but perform poorly in validation dataset, i.e., higher variance. The task to define a model in such a way that the bias and variance can tackle in validation and blind test data set. [51].

2.2.3.3 Occam's Razor for Model Selection & Simplicity

Occam's razor [7] serves as a guiding principle: among competing models, the simplest one that adequately explains the data should be preferred. For instance, Pedro Domingos highlights the importance of understanding when complexity adds value and when it merely adds noise [17]. By adhering to this principle, one can avoid unnecessary complexity and focus on building models that are both effective and efficient.

2.2.3.4 Conclusion

In the context of machine learning and deep learning, choosing the right model depends on the specific problem, available data, and computational resources. While complex models like deep neural networks are powerful, they should be used judiciously. For many tasks, simpler models may offer equal or better performance with fewer risks of overfitting, lower computational costs, and greater interpretability. A thoughtful approach to model selection ensures that the chosen algorithm is both efficient and appropriate for the task at hand.

2.3 Related Work

The integration of Artificial Intelligence (AI) in genomics has revolutionized the field, enabling breakthroughs in DNA sequencing, variant calling, and anomaly detection. For this thesis, which aims to automate the detection and quarantining of anomalous DNA wells in a 96-well plate format, AI offers powerful tools to address the challenges of manual verification and error-prone workflows. Specifically, unsupervised and semi-supervised learning techniques are highly relevant, as they can identify patterns and anomalies in unlabeled sequencing data.

This section explores two state-of-the-art AI algorithms that appear to be best suited for the task. A comparative analysis is presented to evaluate their suitability for automating the quarantining process of DNA wells.

2.3.1 Isolation Forest: Unsupervised Anomaly Detection

Isolation Forest is an unsupervised anomaly detection algorithm that isolates anomalies instead of profiling normal data points. It works by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Anomalies are identified as data points that require fewer splits to be isolated [50].

Methodology Isolation Forest operates by constructing an ensemble of isolation trees. Each tree is built by recursively partitioning the data space until all data points are isolated. Anomalies, which are typically few and different, are isolated closer to the root of the tree, while normal points require more splits to be isolated. The anomaly score for each data point is calculated based on the average path length from the root to the leaf across all trees.

Strengths

- **Anomaly Detection**: Isolation Forest is specifically designed for anomaly detection, making it highly effective for identifying outliers in sequencing data.
- **Handles Mixed Data Types**: While traditionally used for numerical data, Isolation Forest can be adapted to handle categorical data (e.g., primer types) and sequence data by encoding them appropriately.
- **Explainability**: The algorithm provides anomaly scores, which can be used to explain why a well was flagged (e.g., "High concentration level detected" or "Unexpected primer used").
- **Scalability**: Isolation Forest is computationally efficient and scales well to large datasets, making it suitable for high-throughput sequencing workflows.

- **Robustness to Noise**: The algorithm is robust to noisy data, which is important given the potential for errors in sequencing data.

Limitations

- **Feature Engineering**: The algorithm requires careful feature engineering to handle mixed data types effectively.
- **Parameter Tuning**: The performance of Isolation Forest depends on the choice of parameters, such as the number of trees and the contamination rate, which may require tuning.

Relevance to Thesis Isolation Forest is highly relevant to the unsupervised nature of the task in this thesis. Its ability to detect anomalies in mixed data types (e.g., numerical and categorical) aligns well with the goal of automating the quarantining of DNA wells. Additionally, its explainability and scalability make it a strong candidate for high-throughput sequencing workflows.

2.3.2 DBSCAN: Density-Based Clustering and Anomaly Detection

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an unsupervised clustering algorithm that groups data points based on their density. It can identify clusters of arbitrary shapes and mark outliers as noise [18]. DBSCAN is particularly effective for detecting anomalies in datasets with irregular structures.

Methodology DBSCAN works by defining clusters as dense regions of data points separated by sparse regions. The algorithm requires two parameters: the minimum number of points (minPts) and the maximum distance (epsilon) between points to be considered part of the same cluster. Points that do not belong to any cluster are marked as noise (anomalies).

Strengths

- **Anomaly Detection**: DBSCAN explicitly identifies outliers as noise, making it suitable for anomaly detection tasks.
- **Handles Irregular Clusters**: Unlike K-Means, DBSCAN can detect clusters of arbitrary shapes, which is useful for complex genomic data.
- **No Need for Predefined Clusters**: DBSCAN does not require the number of clusters to be specified in advance, making it more flexible than K-Means.
- **Robustness to Noise**: DBSCAN is robust to noisy data, which is important for sequencing workflows.

Limitations

- **Parameter Sensitivity**: The performance of DBSCAN depends on the choice of parameters (epsilon and minPts), which may require tuning.
- **Scalability**: While DBSCAN is efficient for moderate-sized datasets, it can struggle with extremely large datasets due to its computational complexity.
- **Handling Mixed Data Types**: DBSCAN is traditionally designed for numerical data and may require adaptations to handle categorical or sequence data.

Relevance to Thesis DBSCAN is highly relevant to the unsupervised nature of the task in this thesis. Its ability to detect anomalies and handle irregular cluster shapes aligns well with the goal of automating the quarantining of DNA wells. However, its parameter sensitivity and scalability limitations may require careful consideration.

2.3.3 Comparison with Thesis Approach

Table 2.1: Comparison of Isolation Forest, DBSCAN, and Thesis Approach

Aspect	**Isolation Forest**	**DBSCAN**	**Thesis Approach**
Algorithm	Unsupervised Anomaly Detection	Density-Based Clustering	Anomaly Detection & Similarity-Based Quarantining
Data Type	Mixed (Numerical & Categorical)	Numerical (with adaptations for mixed data)	Mixed (Numerical & Sequence Data)
Supervision	Unsupervised	Unsupervised	Unsupervised & Semi-Supervised
Output	Anomaly Scores	Cluster Assignments & Noise (Anomalies)	Cluster and Anomaly Predictions
Scalability	High	Moderate	High
Computational Cost	Moderate	Moderate	Moderate

2.3.4 Other Relevant Algorithms

In addition to Isolation Forest and DBSCAN, several other algorithms and techniques have been developed for genomic data analysis. These methods represent a mix of supervised,

unsupervised, and transformer-based approaches, providing a broader context for the field.

Random Forest for Classification and Anomaly Detection Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their predictions to improve accuracy and robustness. By leveraging multiple trees, Random Forest reduces overfitting and enhances interpretability, making it a reliable choice for structured genomic data analysis [6].

DNABERT: Transformer-Based Genomic Sequence Analysis DNABERT is a transformer-based model designed for genomic sequence analysis. It leverages the self-attention mechanism of transformers to capture long-range dependencies in DNA sequences, enabling tasks such as sequence classification, motif discovery, and variant calling [37]. DNABERT is pre-trained on large-scale genomic datasets and fine-tuned for specific tasks, achieving state-of-the-art performance in several benchmarks.

Nucleotide Transformer: Pre-trained Models for Genomics The Nucleotide Transformer is another transformer-based model that has been pre-trained on diverse genomic datasets. It is designed to handle a wide range of genomic tasks, including sequence alignment, variant calling, and gene expression prediction [15]. Like DNABERT, the Nucleotide Transformer demonstrates the potential of pre-trained models for genomics, particularly in supervised learning tasks.

Autoencoders for Anomaly Detection Autoencoders are a type of neural network used for unsupervised learning, particularly for dimensionality reduction and anomaly detection. They work by compressing the input data into a lower-dimensional representation (encoding) and then reconstructing the original data from this representation (decoding). Wells with high reconstruction errors can be flagged as anomalies [41].

Graph Neural Networks (GNNs) for Relationship Modeling Graph Neural Networks (GNNs) can model relationships between entities in a graph, making them suitable for tasks like gene clustering or pathway analysis. In your case, GNNs could be used to model relationships between wells based on shared features like primer usage, DNA ID, or client information [42].

2.3.5 Conclusion

The review of related work highlights the strengths and limitations of already existing AI algorithms in genomics. By levereging this knowledge with the specific domain goals of this thesis, we can now pursue and detail a proposal.

3

Proposal

This chapter presents the original *thesis proposal* that guided the early stages of this work. It outlines the initial plan for developing an AI-driven tool to automate anomaly detection in Sanger sequencing workflows, including the overall vision, technical approach, and model choices that informed the first phase. The subsequent chapters (§4–5) document how the design evolved and why.

The goal of the proposed system was to improve the efficiency and accuracy of well verification in 96-well sequencing plates by reducing reliance on manual review. By leveraging machine learning techniques, the system was expected to automatically categorize DNA samples based on sequence similarity and associated metadata, enabling faster detection of contamination or sequencing errors.

The chapter begins with an overview of laboratory processes and data generation, then introduces the original architecture and preprocessing pipeline (e.g., DBSCAN + Random Forest). This proposal served as the conceptual foundation for later development.

3.1 Laboratory Processes

The laboratory processes described here are central to the generation of the data used in this work. These processes involve the handling of DNA samples, their preparation for sequencing, and the subsequent analysis of the results.

3.1.1 Sample Preparation and Processing

The laboratory workflow begins with the reception of DNA samples from various clients. These samples are organized into 96-well PCR plates, where each well is identified by a unique coordinate system: letters A to H for the rows and numbers 1 to 12 for the columns (Figure 3.1). This organization ensures traceability and efficient handling of multiple samples simultaneously. Samples from the same client are grouped together in neighboring wells, although not all wells need to be occupied for sequencing to proceed.

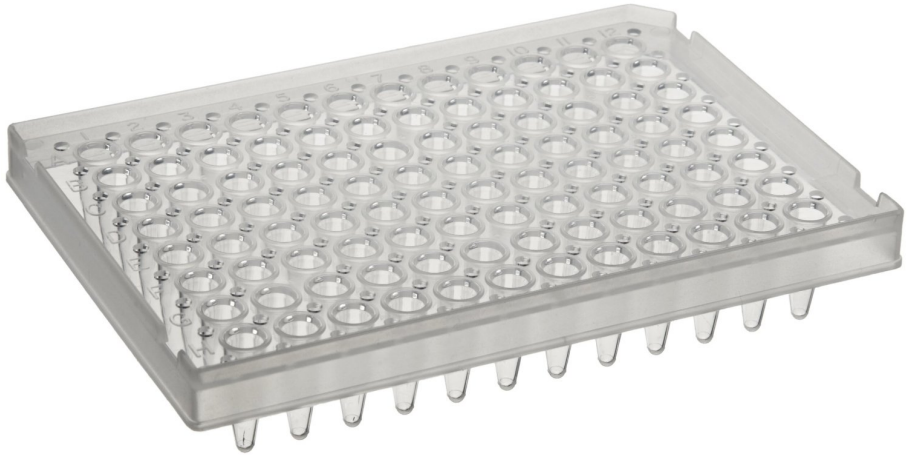

Figure 3.1: A 96-well PCR plate.

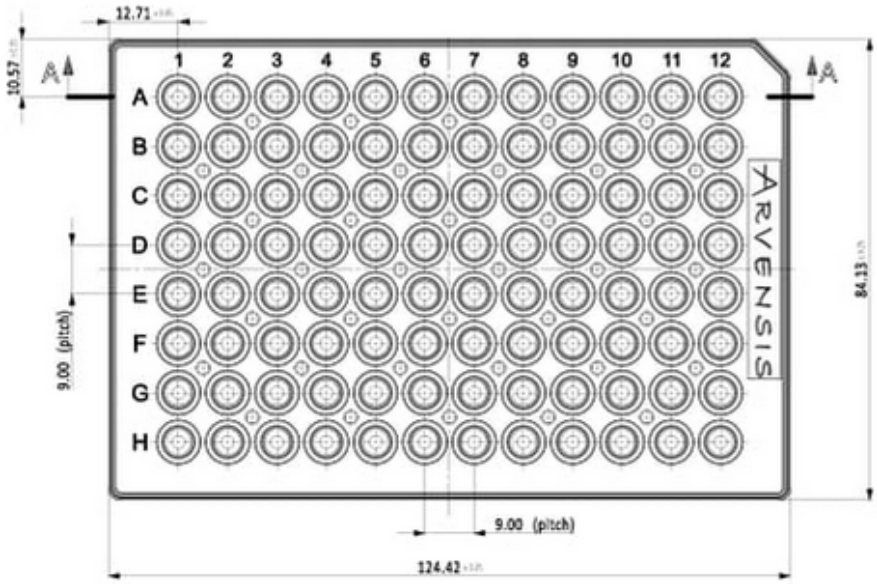

Figure 3.2: Structure and Organization of each PCR plate.

Upon receipt, some samples require human intervention to prepare them for sequencing. For instance, samples with uneven DNA concentrations must be normalized to ensure consistent sequencing quality. Additionally, samples that do not include a primer must have the appropriate primer added by laboratory staff. These steps are critical to ensure the accuracy and reliability of the sequencing results. Each plate also contains 1 well reserved solely for quality control, named control well.

Once prepared, these plates are then loaded into a thermocycler, which includes the steps of DNA denaturation, primer annealing, and extension with fluorescently labeled dideoxynucleotides. Afterwards the electrophoresis is performed on the ABI3730xl sequencer. The ABI3730xl sequencer then generates raw data files containing fluorescence intensity readings, which are subsequently basecalled to produce DNA sequences.

3.1.2 The ABI3730xl DNA Sequencer

The ABI3730xl DNA sequencer is a high-throughput instrument widely used in genomics for Sanger sequencing. It automates the electrophoresis and fluorescence detection steps, corresponding to steps 3 and 4 of the Sanger sequencing process (Figure 2.3), allowing for the simultaneous processing of 96-well plates in a single run [72, 4]. By leveraging this technology, laboratories can achieve precise and efficient DNA sequencing.

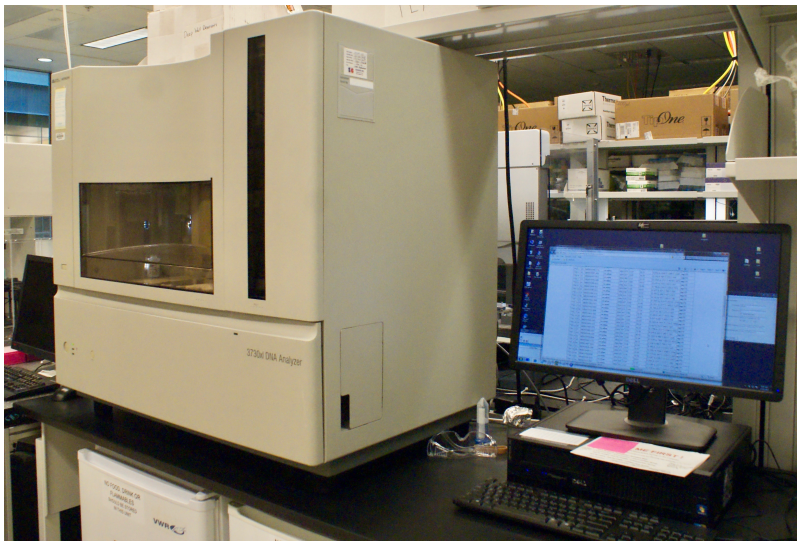

Figure 3.3: A closed ABI3730xl capillary electrophoresis DNA analyzer.

Figure 3.4: An opened ABI3730xl capillary electrophoresis DNA analyzer.

3.1.3 Data Generation & Collection

The effectiveness of any artificial intelligence model relies heavily on the quality and structure of the available data. In this work, the data originates from customer samples sequenced using the ABI3730xl DNA sequencer. Each well in a 96-well PCR plate generates multiple files, including raw fluorescence data, processed DNA sequences, and quality metrics. With each plate containing 96 wells and each well producing 6 distinct files, a single plate yields 576 files. Over time, as multiple plates are processed daily, this results in millions of files, forming a robust dataset for analysis. The primary file types considered in this work include:

- **Ab1 File**: Contains intensity readings for each of the four fluorescent dyes (A, T, C, G) used in Sanger sequencing, resulting in fluorescent traces. This file is optimized for long sequences (more than 500 bases).
- **Phd1 File**: Contains the DNA sequence derived from the fluorescence traces, along with quality scores and scan numbers.
- **Scf File**: Similar to .ab1 files, .scf files include quality information about the base calls, the chromatogram (also called the electropherogram), and the DNA sequence.
- **Seq File**: A plain text file containing the DNA sequence.
- **Qual File**: Includes the quality scores of each base in the DNA sequence.
- **Qualtrace File**: Contains additional information about the sequencing process, such as PeakStart, DyeSignal, Noise, and Spacing.

These files adhere to a strict naming convention: **ReactionID_Pos_DNAID+PrimerID**. For example, the file name `2338715_A1_AA8117+Clon_2+ORB501` can be broken down as follows:

- `2338715`: The unique reaction ID.
- `A1`: The position of the well in the plate (row A, column 1).
- `AA8117+Clon_2`: The DNA ID, identifying the specific DNA sample.
- `ORB501`: The primer ID, which is optional as some customers provide samples with primers already included.

Thus, 6 files are generated for this specific well:

- `2338715_A1_AA8117+Clon_2+ORB501.ab1`
- `2338715_A1_AA8117+Clon_2+ORB501.phd.1`
- `2338715_A1_AA8117+Clon_2+ORB501.scf`

- `2338715_A1_AA8117+Clon_2+ORB501.seq`
- `2338715_A1_AA8117+Clon_2+ORB501.qual`
- `2338715_A1_AA8117+Clon_2+ORB501.qualtrace`

Other relevant and important information about each plate well, such as the associated **Client's Name**, **Sequence Length**, **Primer DNA Sequence** (If applied by the laboratory staff), and the **Control Well Status** is also available.
It is also important to note that while DNA sequences are primarily composed of the four canonical bases: adenine (A), thymine (T), cytosine (C), and guanine (G), some sequences contain additional IUPAC ambiguity codes such as `N` (any base) or `Y` (C or T). These ambiguous bases typically arise from basecalling uncertainty in low-quality signal regions. Their presence has implications for downstream processing, especially for sequence encoding in machine learning models, which would need to be addressed during downstream sequence encoding.

3.1.4 Manual Analysis

After the data is generated, the laboratory staff manually handles quality checks on each plate individually, particularly for identifying possible fluorescence contamination and mismatches in sequencing results. This is a lengthy and thorough process that currently induces a bottleneck in the laboratory workflow. The main steps for quarantining wells are as follows:

- **1. Control Well Check**: The laboratory staff begins by checking the `control_well` for each plate. If the control well's `ok_flag` is `NOT OK`, the entire plate is quarantined, as this indicates a problem with the sequencing process. If the control well passes the check, the process moves to the next step.
- **2. Sequence Clustering**: The sequencing software extracts the `ab1` files from each well (96 wells per plate) and clusters the sequences based on a minimum similarity of 66%. This clustering is performed to group together DNA samples that are likely to belong to the same biological entity. Each cluster represents a set of related DNA sequences.
- **3. Cluster Verification**: The next step involves verifying these clusters for potential contamination or sequencing errors. Laboratory staff checks for two main conditions:
  - *Similar Sequences from Different Clients*: If samples from different clients have very similar DNA sequences, it may suggest fluorescence contamination, triggering a quarantine for further investigation.

  - *Same Primers with Different Sequences*: If different primers are associated with very similar DNA sequences, this could indicate contamination and thus requires quarantining.

- **4. Forward and Reverse Sequence Matching**: DNA samples are analyzed based on forward and reverse sequencing. If a single base mismatch is detected in the same DNA sample (with matching `DNAID` and `primer_id`), it is flagged for quarantine. In the case where a sample is processed with different primers, a variation in sequence is expected.

Sometimes certain samples do not appear to be contaminated, but the overall suspicion of such, or the poor quality presented, is enough to justify a quarantine, due to the importance of delivering accurate results. After the anomalies are quarantined, these possibly contaminated samples are re-sequenced for an attempt at acquiring higher quality and accuracy, while the remaining samples are delivered to the client.

Labeled Anomalies: Although anomalies are not registered automatically in the laboratory's database, a small set of manually verified anomalies was made available upon request. Due to the rarity of confirmed contamination or sequencing errors in practice, the laboratory only identified **four plates** with known anomalies during the course of the data collection of this thesis. These labeled plates were used for evaluating the performance of the proposed models against the laboratory's ground truth. This scarcity of labeled anomalies further justified the adoption of machine learning strategies throughout the project.

Table 3.1: Manual workflow vs. proposed automation (proposal phase).

Manual Step	**Proposed Automation (at proposal time)**
Control-well check	Rule: if control `NOT OK` $\Rightarrow$ quarantine entire plate.
Cluster sequences ($\geq$66% similarity)	DBSCAN over k-mer vectors to form clusters; ignore outliers.
Different clients, very similar sequences	Cluster-level metadata checks: flag mixed-client clusters.
Different primers, very similar sequences	Cluster-level rule: flag unexpected primer–sequence similarity.
Forward vs. reverse mismatch (same DNAID)	Pairwise alignment and per-sample rules; escalate on single-base mismatch.

3.2 Initial Proposed Architecture

Grounded in the laboratory workflow and data described above, we formulated an initial theoretical approach to automate plate-level quality control. The initial system design

aimed to automate the manual verification of 96-well Sanger sequencing plates using artificial intelligence. The proposed approach focused on replicating the laboratory's reasoning process using a combination of clustering techniques and anomaly detection models. The intention was to streamline the pipeline from data ingestion to anomaly flagging, leveraging both sequence similarity and metadata.

This section presents the architecture that was initially envisioned, including the database schema, preprocessing pipeline, and the selection of machine learning models. This intial design served as a conceptual and experimental foundation.

3.2.1 Database Structure

At proposal time we defined a *logical per-well table schema* independent of storage technology. A NoSQL plate-as-document option was briefly considered but deferred. The essential idea was to model, for each plate, a single row per well aggregating identifiers, sequence text, quality metrics, and run metadata. The concrete mechanism to populate this schema (a unified per-plate export) was implemented later during the development phase (see Chapter 4).

Original Proposed Schema: At the proposal stage, the following schema was envisioned based on assumptions about data availability and early exploration of STAB VIDA's sequencing workflow. The table included planned fields ranging from raw fluorescence traces to primer sequences and purification metadata.

Column Name	Data Type	Description
`Plate_ID`	String	Unique identifier for each 96-well PCR plate.
`Well_POS`	String	Unique identifier for each well position (e.g., A1, B2).
`Reaction_ID`	String	Unique identifier for the reaction associated with the well.
`DNA_ID`	String	Identifier for the DNA sample.
`Primer_ID`	String	Identifier for the primer used (if applicable).
`Client_Name`	String	Name of the client associated with the sample.
`Control_Well_Status`	String	Status of the control well (e.g., "OK" or "Quarantine").
`DNA_Sequence`	Text	The DNA sequence derived from the '.seq' or '.phd.1' file.
`Base_Length`	Integer	The total number of bases of the DNA sequence.
`Quality_Scores`	Text	Quality scores for each base in the sequence (from '.qual' or '.phd.1' files).
`Fluorescence_Traces`	Text	Fluorescence intensity readings (from '.ab1' or '.scf' files).
`PCR_Size`	Integer	Size of the PCR product.
`Purification_Details`	Text	Information about the purification process.
`Primer_Sequence`	Text	Sequence of the primer used (if manually added by staff).
`Date_Processed`	Date	Date the plate was processed.
`Anomaly_Detected`	Integer	Although limited, some data is labeled with this flag.

Table 3.2: Table Structure for Sequencing Data and Metadata.

This table structure ensured that all relevant information of each well derived from different generated files, is stored in a single organized row, simplifying data retrieval and analysis. It includes fields for plate and well identification, sequencing data, quality metrics, and metadata, making it suitable for both clustering and contamination detection tasks.

Operational dependency (proposal). At the time of the proposal, a unified per-plate export did not yet exist. We specified this as an implementation requirement; the company later delivered a backend export page that produced the input needed for this schema (cf. Chapter 4).

3.2.2 Data Preprocessing (Feature Engineering)

To ensure optimal input quality for the AI-based model, multiple preprocessing steps are performed on the raw sequencing data. This includes sequence trimming, feature extraction, normalization of numerical features, and handling categorical/text-based attributes appropriately.

3.2.2.1 Sequence Trimming

Raw sequencing data contains noise, in the form of poor base quality, particularly at the start and end of sequences. At the beginning, this is primarily due to inefficient primer binding and weaker signal intensities as the primer binds and starts the reaction. At the end, the reduced availability of chain-terminating nucleotides (ddNTPs), declining polymerase efficiency, and signal saturation contribute to weaker and less accurate readings. [66]

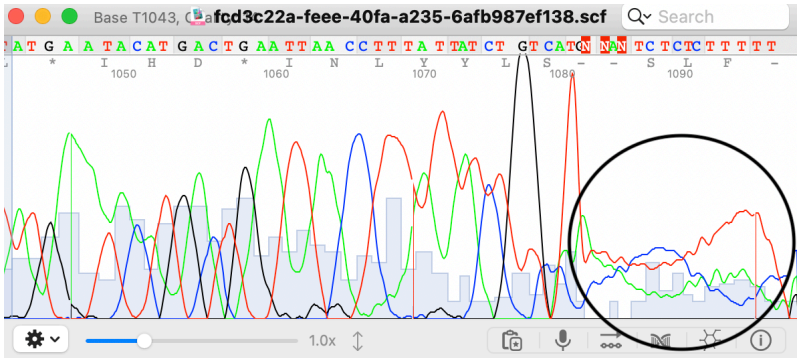

Figure 3.5: Visualization of noise in the beginning of the DNA sequence, using 4peaks.

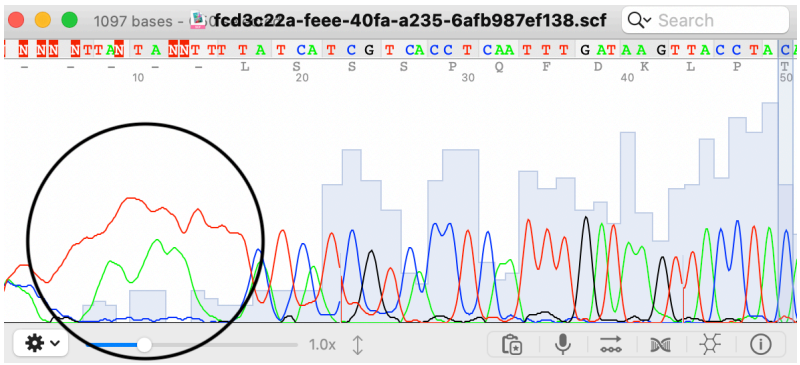

Figure 3.6: Visualization of noise in the end of the DNA sequence, using 4peaks.

To improve downstream analysis, sequences are trimmed using the following criteria:

- Bases are removed from the start and end of the sequence until **10 consecutive bases** have a **quality score** greater than 20.
- This threshold ensures high-quality regions are retained, as a quality score of 20 corresponds to a base-calling accuracy of 99% [20].

The trimmed sequences are then stored for subsequent steps.

3.2.2.2 Sequence Encoding

To represent DNA sequences in a numerical format suitable for clustering:

- **k-mer Frequency Encoding**:

- Each DNA sequence is converted into a vector of k-mer frequencies.
- A k-mer size of 4 is chosen to balance specificity and computational efficiency, as tetramers provide sufficient resolution for pattern recognition [13].
- For example, a 4-mer approach splits the sequence into overlapping subsequences of length 4 (tetramers) and counts their occurrences.
- This method captures local sequence patterns and is particularly useful for clustering algorithms like DBSCAN.

3.2.2.3 Normalization

To ensure consistent scaling of features, while also ensuring numerical features contribute equally to the model:

- **Sequence Feature Normalization**:
  - The k-mer frequency vectors and the numerical features such as quality scores, sequence lengths, and other quantitative features, are normalized using **min-max scaling** or **z-score normalization** [29].
  - This ensures all features are on a similar scale, preventing any single feature from dominating the clustering process.

3.2.2.4 Dimensionality Reduction

To handle the high dimensionality of k-mer frequency vectors:

- **Principal Component Analysis (PCA)**:
  - PCA is applied to reduce the dimensionality of the feature space while retaining the most significant variance.
  - This step reduces computational complexity and mitigates the risk of overfitting.
  - The number of principal components is selected based on the cumulative explained variance, retaining 95% of the variance for example is a common threshold to balance complexity and information retention [40].

3.2.3 Model Selection & Development

The goal of the proposed AI-based tool is to fully automate the process currently performed manually by the laboratory staff. This involves categorizing DNA samples into wells based on their similarity and identifying anomalies that could indicate contamination, sequencing errors, or other irregularities. Since the proposed work is based on AI,

there is a clear need to choose appropriate models for specific tasks. Given the complexity of the problem and the multiple steps involved in manual analysis—ranging from sequence clustering to anomaly detection—a **hybrid approach** is proposed to achieve both tasks. This approach leverages **two machine learning models**, one for clustering and another for anomaly detection inside these formed clusters:

- **Unsupervised Clustering (DBSCAN)**: Groups DNA sequences based on similarity, mimicking the laboratory's manual clustering process without requiring labeled anomalies.
- **Supervised Classification (Random Forest)**: Detects anomalies using labeled data and features derived from clustering, enhancing detection accuracy.

Thus, a **structured pipeline** seems to be a good choice and starting point for streamlining the data from one model to another, combining the strengths of unsupervised and supervised learning to address the dual challenges of sequence similarity and anomaly detection.

3.2.3.1 Unsupervised Learning for Clustering - DBSCAN

Given the lack of cluster information data in the initial stages, unsupervised learning is particularly suitable for this task, as we first need to group samples that appear to be similar to each other together, indicating they are possibly from the same client. Unlike centroid-based methods like K-means, DBSCAN does not require predefined cluster numbers, making it adaptable to diverse datasets. DBSCAN is used to group sequences into clusters based on their k-mer frequency representations. This unsupervised approach identifies natural groupings in the data, allowing for the detection of anomalies within clusters.

- **Parameter Tuning**:
  - The key parameters—*eps* (maximum distance between samples in the same cluster) and *min_samples* (minimum number of samples in a cluster)—are optimized using grid search and silhouette score evaluation.
- **Cluster Formation**:
  - Sequences are grouped into clusters based on their similarity in k-mer frequency space.
  - **Samples not assigned to any cluster** are ignored, as the focus is on detecting anomalies within clusters.
- **Prediction**:

  - The cluster labels (0, 1, etc.) now act as pseudo-labels for grouping similar samples in the following model.
  - Outliers (-1) are ignored.

3.2.3.2 Semi-Supervised Learning for Anomaly Detection - Random Forest

To leverage the limited labeled anomalies provided by the laboratory team, a semi-supervised learning approach is proposed. This combines the unsupervised clustering results from DBSCAN with a supervised classifier to identify samples that do not belong in their assigned clusters.

- **Using DBSCAN's output with the labeled data available**:
  - Manually labeled anomalies (from laboratory annotations) are combined with DBSCAN's pseudo-labels to create a semi-supervised dataset for training.
- **Classification with Random Forest**:
  - A Random Forest classifier is trained on the labeled dataset due to its robustness and interpretability.
  - Feature importance analysis is performed to identify which k-mers contribute most to detecting anomalies within clusters.
- **Prediction**:
  - The trained classifier predicts whether a sample belongs to its assigned cluster, flagging it as an anomaly if it does not.
  - This provides automated quarantine recommendations for samples that are likely to be misassigned within clusters.

3.2.4 Evaluation (Planned)

Clustering quality would be scored via silhouette/Davies–Bouldin; anomaly detection via precision, recall, F1, and confusion matrices. Thresholds would be tuned to prioritize recall (safety-first), as in this proposal stage, it was already understood that the laboratory team prioritized a higher percentage of a plate being flagged as an anomaly, given that all anomalies were successfully detected, rather than some anomalies managing to slip away detection due to a low percentage of flagged wells.

3.2.5 Frameworks, Languages & Tools

The development and implementation of the proposed AI-driven tool relied on the following technologies, selected for their suitability in data processing, machine learning, and deployment:

Table 3.3: Planned frameworks and tools (proposal stage).

Category	**Tools/Frameworks**	**Rationale / Status**
Programming	Python	Scientific stack and bioinformatics ecosystem.
Data Processing	NumPy, Pandas, Biopython	Numeric ops, tabular IO, sequence utilities.
Machine Learning	scikit-learn, imbalanced-learn	DBSCAN, RF, sampling utilities.
Visualization	Matplotlib, Seaborn	EDA and reporting.
Database	SQLite / PostgreSQL	*Considered* early; **not used**—per-plate Excel retained.
Deployment	Flask/FastAPI, Docker	*Considered* for a service wrapper; **not part** of final artifact.
Bioinformatics	Needleman–Wunsch, Smith–Waterman (Biopython)	Alignment utilities for verification/analysis.
Compute	Workstation / HPC	For batch preprocessing and experiments.

3.3 Limitations of the Proposal (Why We Pivoted)

The proposal assumed that:

1. bag-of-k-mers (order-agnostic) would suffice for similarity,
2. global clustering across plates would be informative, and
3. semi-supervised RF on cluster features would be robust under scarce labels.

During implementation we identified key gaps (quantified in Chapter 5):

- **Order sensitivity.** Bag-of-k-mers discarded base order; single-base mismatches—material to Sanger QC—were easily missed.
- **Plate locality.** Anomalies are defined *within* plates; global clusters risked cross-plate leakage and false structure.
- **Label scarcity & imbalance.** Supervised RF/MLP variants were unstable; we required stronger unsupervised signals (per-plate AE) and explicit rules.
- **Ambiguity/short reads.** Treating `N/Y` as neutral tokens and padding short reads polluted representation learning.

These findings motivated the final design in Chapter 4: plate-local, alignment-informed neighbourhoods; an order-aware *per-plate* Conv1D autoencoder (latent + reconstruction error); a calibrated LightGBM classifier on engineered features (excluding identifiers); and a post-hoc rule layer for interpretable, policy-aligned decisions.

4

Implementation

This chapter details the project's implementation. We document the concrete design, the pivots from the initial proposal, and the design choices that led to the final configuration. Quantitative validation and ablation results are reported in Chapter 5. The pipeline runs end-to-end and produces interpretable anomaly flags suitable for routine laboratory use.

Name and Rationale. We named the system Prometheus to reflect its purpose and ethos. In Greek myth, Prometheus (Προμηθεύς, "forethought") brings fire, knowledge and capability to humans, while accepting responsibility for its use. Analogously, Prometheus brings computational "fire" (automation, pattern discovery, and early warnings) into a life-science workflow, with forethought as a design principle: plate-local modeling to avoid leakage, conservative rules to prevent brittle decisions, and human-in-the-loop review for safety.

4.1 Design Evolution at a Glance

Three constraints drove the change from the proposal (DBSCAN + Random Forest) to the final pipeline:

1. **Plate locality.** Anomalies are defined *within* each 96-well plate (comparisons to sibling wells).
2. **Order sensitivity.** Base order matters (e.g., single-base mismatches), so we need a sequence model rather than bag-of-k-mers alone.
3. **Label scarcity.** Very few confirmed anomalies; we must lean on unsupervised/weakly-supervised signals plus interpretable rules.

Consequently, Prometheus adopts a *per-plate* Conv1D autoencoder (AE) to learn plate-specific normal patterns and extract latent features; a supervised gradient-boosting classifier (LightGBM) with plate-wise cross-validation; and a post-hoc rule layer. Plate-local sequence similarity at $\geq 66\%$ identity (BLAST-like) supplies contamination-style features and rule checks.

4.2 Co-Design with the Laboratory (Requirements Drift)

The project evolved through continuous discussions with STAB VIDA's biotechnology team. Several operational realities surfaced after the proposal and materially influenced the design:

- **Direction metadata (forward/reverse).** No reliable field exists in routine exports; the team often infers direction from context or ad-hoc tags ("F"/"R") that are inconsistently present. We therefore *do not model direction as a feature*. Reverse-complement checks are applied *only when* a reliable pairing is available; otherwise they are omitted to avoid false positives.

- **Naming variability.** `DNA_ID` and `Primer_ID` are user-entered and may be subjective or inconsistent (synonyms, typos, multiple conventions). To prevent shortcut learning and cross-plate leakage, these identifiers are excluded from the classifier (kept for explanations and rules only).

- **Privacy & identifiers.** Client names are replaced by pseudonymous `Client_IDs`. This reduces identifiable information and aligns with the lab's confidentiality requirements.

These realities transformed the thesis from a straightforward "build the planned system" into an *investigation* that bridged domain knowledge and robust ML/software engineering choices.

4.3 System Overview and Orchestration

A watcher process `deploy_prometheus.py` runs continuously at the project root. It polls the `input/` folder every few seconds and processes plates *sequentially* (one at a time) through Steps 1–4. **After each attempt—success or failure—the input Excel is moved to `input/processed/`.** To re-run a failed plate, the file must be re-added to `input/`. After a successful run, a background routine (Step 5) is launched asynchronously for optional historical validation.

1. **Step 1 – Preprocessing** (`preprocess_plates_wrapper.py`): Q20 trimming, ambiguity flags, position-aware 4-mer tokenization, and plate-local similarity (≥66%).

2. **Step 2 – Autoencoder (per plate)** (`auto_encoder_wrapper.py`): train seq-to-seq Conv1D AE; export per-well latent vectors and reconstruction errors.

3. **Step 3 – Features & Classifier** (`build_features_wrapper.py`, `lgbm_classifier_wrapper.py`): merge features; score with LightGBM (plate-wise CV during development, calibrated threshold).

4. **Step 4 – Post-hoc Rules & Exports** (`generate_results_wrapper.py`): fuse model scores with rules; save interpretable CSVs.

5. **Step 5 – (Async) Historical/Taxonomic Check** (`verify_anomalies_with_history.py`): launched *after* a successful run to enrich flagged wells with optional BLAST taxonomy and customer-history alignment; runs in the background.

Artifact locations. Steps 1 and 2 write staging artifacts under `Step_1/output/` and `Step_2/output/`. The orchestrator copies *and opens* the plate layout HTML at `results/{plate}/{plate}_layout.html`. Step 3 writes `results/{plate}/features_lgbm.csv` and `results/{plate}/predictions_lgbm.csv`; Step 4 writes `results/{plate}/predictions_interpretable.csv` and `results/{plate}/predictions_in`

4.4 From Proposal to Implementation: Stepwise Evolution

This section bridges Chapter 3 to the implemented pipeline by comparing, per step, what was proposed, what we observed, and the change we made. Quantitative evidence for these pivots is presented in Chapter 5.

4.5 Data Access & Collection (Implemented)

Although the sequencing run produces multiple file types per well (`.ab1`, `.seq`, `.phd.1`, `.qual`, `.qualtrace`, etc.), these do not arrive as a single joined artifact by default. Operationally, STAB VIDA maintains a backend with several internal pages that expose different slices of information (run metadata, quality metrics, client identifiers, primer details, control-well status). During implementation, STAB VIDA delivered a backend page that accepts a `Plate_ID` that follows the nomenclature mentioned in (chatGPT please mention the correct chapter) and returns a unified `.xlsx` with one row per well (up to 96), merging fields from multiple internal sources (run metadata, client/sample registry, primer management, basecalling/quality, trace metrics, control-well status). This *per-plate Excel* became the canonical, read-only input to PROMETHEUS.

Unified Export. A backend page accepts a `Plate_ID` and returns a downloadable `.xlsx` containing one row per well (up to 96 rows), with fields merged from multiple backend sources. This *plate-level Excel* became the canonical interface for the project: it preserved well granularity, matched the laboratory's plate-by-plate workflow, and avoided direct coupling to internal databases or proprietary UIs.

Confidentiality. Client names are replaced by pseudonymous `Client_IDs` in the export. Each file is saved as `{PlateName}.xlsx` and logged with export timestamp and, when available, the triggering user and backend version.

Rationale. (i) The laboratory team already reviews data plate-by-plate; (ii) model training and evaluation are defined *within* plates; and (iii) several required attributes originate from different backend pages. A unified export ensured consistent joins, minimized manual handling, and enabled reproducible experiments.

Table 4.1: Backend sources and example fields included in the unified per-plate export.

Backend Source (internal)	**Example Fields**	**Notes**
Run metadata / plate view	`Plate_ID`, `Well_POS`, `Reaction_ID`, date	Primary keys, scheduling
Client/sample registry	`Client_ID`, `DNA_ID`, `Barcode`	May be missing for some samples
Primer management	`Primer_ID`, `Primer_Sequence`	Primer can be lab-added
Basecalling/quality	`DNA_Sequence`, `Base_Length`, `Quality_Scores`	Derived from `.phd.1/.qual`
Trace metrics	`DyeSignal`, `Noise`, `Spacing`, `TRACE_SCORE`	From `.ab1/.qualtrace`

Consistency Checks (at ingestion). We collaborated with STAB VIDA's backend engineering team to validate the export. Multiple test exports were cross-checked to confirm field completeness and consistency with expectations.

Scope Note. This subsection documents the operational access path realized during implementation. The internal backend itself is out of scope; our artifacts and experiments depend only on the exported per-plate Excel files.

Step 1 — Data Processing & Similarity

Proposal (Ch. 3). Global clustering with DBSCAN on PCA-reduced k-mer frequencies; basic end-trim.

Observed issues. (i) Bag-of-k-mers lost order (single-base mismatches slipped through). (ii) Global cluster IDs encouraged cross-plate leakage; clusters are inherently plate-local. (iii) Tail noise (`N/Y`) polluted features; short reads distorted representations.

Implemented change. (i) Plate-local BLAST-like pairwise identity ($\geq$ 66%) to form in-plate neighbourhoods (no cross-plate reuse). (ii) Order-sensitive 4-mer windows (stride = 1 for encoding; exported as 150-token windows with stride = 50). (iii) Central Q20×10 trimming; ambiguous bases *flagged* (not encoded); reads < 150 bases bypass AE.

Takeaway. Treat plates as isolated universes; keep order; do not encode ambiguity.

Step 2 — Representation Learning (Autoencoder)

Proposal (Ch. 3). No sequence model; PCA on k-mers; Random Forest downstream.

Observed issues. Missing order sensitivity; global models underfit plate-specific distributions; padding short reads injected noise; mapping `N/Y` to neutral tokens corrupted latents.

Implemented change. Per-plate Conv1D autoencoder over 4-mer tokens; skip AE when <150 bases or ambiguity remains (`Short_Read/Has_Non_ACTG_Bases`); export 32-d latent + reconstruction error per well.

Takeaway. Small per-plate encoders capture local regularities that global models miss.

Step 3 — Classifier & Leakage Control

Proposal (Ch. 3). Random Forest on cluster-derived features; per-well splits.

Observed issues. Per-well splits leak plate context; RF/MLP unstable under class imbalance; metadata shortcuts (`Client_ID/Primer_ID`) suppress sequence features.

Implemented change. LightGBM with GroupKFold by plate and calibrated threshold; remove `Client_ID/Primer_ID` from the model (kept for explanations/rules); positive class weighting; permutation-importance checks. Quantitative comparisons across RF/XGB/LGBM are in Chapter 5.

Takeaway. Grouped splits, calibrated boosting, and careful feature hygiene prevent illusory gains.

Step 4 — Post-hoc Rules & Reporting

Proposal (Ch. 3). Purely model-driven decisions.

Observed issues. Heterogeneous anomaly types (contamination vs. forward/reverse mismatch vs. short/ambiguous reads) benefit from explicit logic; the lab needs traceable reasons and a compact export.

Implemented change. Weighted rules (cross-client homology, cross-primer consistency, ambiguity/short-read handling, forward/reverse reverse-complement check when available), fused with model probability; two exports: an interpretable CSV and a LabVIEW-ready compact CSV. Precision–recall changes due to fusion are presented in Chapter 5.

4.6 Data and Final Per-Well Schema

Each plate arrives as a single Excel file with one row per well (sequence, quality/trace metrics, metadata). The schema used in implementation is:

Table 4.2: Final schema used for per-well data (Excel input).

Column	**Descr**
Plate_ID, Well_Position	Plate
Client_ID, Barcode, DNA_ID, Primer_ID	Subm
Purification_Status, PCR_Size	Purifi
DNA_Sequence, DNA_Sequence_Quality	Base s
TRACE_SCORE, Noise_Level, QtClass, PEAK_WIDTH, artefact flags	Instru
TOTAL_BASES, GOOD_BASES, INDEL, EARLY/MID/LATE_SIGNAL, EARLY_MIX, LATEMIX	Deriv
Anomaly	Manu

Feature usage note. Although `TOTAL_BASES` and `GOOD_BASES` may be present in the raw Excel, they are *not* used by the classifier because they correlate with trimming outcomes and introduced redundancy and noise in early trials.

4.7 Preprocessing

4.7.1 Quality Trimming and Ambiguity Handling

Sequences are trimmed from both ends until **10 consecutive bases** have Phred $\geq$ 20 (Q20 $\approx$99% accuracy). Ambiguous IUPAC bases (`N`, `Y`, ...) are *not* fed to the sequence model; wells that still contain ambiguity after trimming are flagged (`Has_Non_ACTG_Bases`=1). Trimming is anchored on the highest-quality central segment and expands outwards until the Q20$\times$10 criterion is satisfied, reducing tail noise.

4.7.2 Minimum Length and Short-Read Path

We build 4-mer windows only for reads with $\geq$ 150 trimmed bases. Wells shorter than this threshold skip the AE path (latents/reconstruction become `NaN`); they are handled by rules and by the tabular classifier using metadata/trace features plus `Short_Read`/`NoLatent` flags.

4.7.3 Sequence Representation and Single-Base Sensitivity

We use overlapping, position-aware 4-mer tokenization into fixed-length **150-token** windows (4^{mer} vocabulary size = 256). A single-base difference perturbs local 4-mer composition, which makes mismatches separable in both the AE latent space and the alignment-derived features.

4.7.4 Plate-Local Similarity and Clustering Signals

For each plate we compute pairwise sequence identity using a local BLAST-like alignment; wells with identity ≥ 66% form neighbourhoods. These neighbourhoods are *plate-scoped* (IDs not reused across plates) and serve to engineer:

- `Mixed_Client` / `Mixed_Primer` flags within a neighbourhood,
- nearest-neighbour identity features,
- "same DNA_ID, different primer" consistency checks (and reverse-complement checks when direction is available).

4.7.5 Curation Notes

A falsely labelled well (`F3`) was removed after manual verification. `PCR_Size` is normalised and uses `-1` when absent.

4.7.6 Per-Step Inputs and Outputs (I/O Contracts)

Table 4.3: Step 1 – Preprocessing (`preprocess_plates_wrapper.py`)

Input	Excel plate (`input/{plate}.xlsx`)
Ops	Q20 trimming (10 consecutive bases), ambiguity flags (N/Y), 4-mer tokenization into fixed-length 150-token windows for reads ≥ 150, plate-local identity (≥ 66%).
Output	`Step_1/output/{plate}_kmer_windows.npy`; `Step_1/output/{plate}_window_map.npy`; `Step_1/output/{plate}_metadata.xlsx`; and a BLAST neighbourhood layout HTML under `Step_1/output/visualizations/`. The orchestrator copies and opens the HTML at `results/{plate}/{plate}_layout.html`.

Table 4.4: Step 2 – Autoencoder (`auto_encoder_wrapper.py`)

Input	`{plate}_kmer_windows.npy`, `{plate}_window_map.npy`, metadata (from Step 1 outputs).
Ops	Train per-plate Embedding+Conv1D seq-to-seq AE over 150-token inputs; export latent (mean-pooled) and reconstruction error per well.
Output	`Step_2/output/{plate}_autoencoder.h5`, `Step_2/output/{plate}_latent_vectors.npy`, `Step_2/output/{plate}_with_errors.xlsx`, `Step_2/output/{plate}_loss_curve.png`.

Table 4.5: Step 3 – Features & Classifier

Input	AE exports + Step 1 metadata + similarity features.
Ops	Build tabular features (drops `Client_ID/Primer_ID` for modeling). LightGBM scoring with calibrated threshold.
Output	`results/{plate}/features_lgbm.csv`, `results/{plate}/predictions_lgbm.csv`.

Table 4.6: Step 4 — Post-hoc Rules & Exports (`generate_results_wrapper.py`).

Input	`predictions_lgbm.csv` + rule features.
Ops	Weighted rule scoring and fusion with $\hat{p}$. Threshold values were selected empirically during validation (Chapter 5).
Output	`predictions_interpretable.csv`, `predictions_interpretable_labview.csv`.

4.8 Per-Plate Conv1D Autoencoder

Rationale. Plates are independent operational units; training one AE per plate lets the model learn only that plate's distribution ("small agents"). A global AE was tested and underperformed; the plate AE was kept (a configuration flag allows switching).

Model. A sequence-to-sequence Conv1D autoencoder over 4-mer tokens:

- **Tokenization:** 4-mer vocabulary ($|\mathcal{V}| = 256$), inputs are **150-token** windows derived from trimmed reads.
- **Encoder/Decoder:** `Embedding`$(256, 32)$ with Conv1D blocks to a *bottleneck* of 32; symmetric decoding with a `TimeDistributed(Dense(256, softmax))`.
- **Loss & training:** sparse categorical cross-entropy; `BATCH_SIZE=64`, `EPOCHS=40`, early stopping (`patience=5`).

Outputs per well: (i) mean reconstruction error across windows, (ii) a 32-dim latent (mean-pooled). Reads < 150 bases or with post-trim ambiguity produce `Short_Read/Has_Non_ACTG_Bases` flags and skip AE.

4.9 Classifier and Evaluation Protocol

4.9.1 Retired Configurations

We trialled (i) a global AE trained across plates, (ii) an MLP classifier with binary cross-entropy and class weights, and (iii) XGBoost. The global AE underperformed due to loss of plate context; the MLP was more sensitive to class imbalance and data scarcity; XGBoost was comparable to LightGBM but less stable under plate-wise cross-validation. LightGBM with calibrated thresholding was therefore adopted.

4.9.2 Tabular Classifier

We compared MLP, XGBoost, and LightGBM; LightGBM was selected for stability under plate-wise cross-validation and strong performance with mixed dense/flag features. Class imbalance is addressed via positive class weighting and decision-threshold tuning (recall-priority). The deployment decision threshold is $\tau = 0.47$, calibrated on validation folds (see Chapter 5); Step 4 later fuses $\hat{p}$ with rules for deployment decisions.

4.9.3 Train/Test Split and Leakage Control

All splits are **by plate** (plate-wise GroupKFold): entire plates are held out. This prevents leakage that would occur if wells from the same plate appeared in both train and test.[1] During training, each fold produces a model; deployment trains a single final model on all training plates after CV hyperparameter selection, avoiding fold-specific $\hat{y}$ discrepancies.

4.9.4 Feature Engineering and Leakage Mitigation

Features are grouped as:

- **AE features:** reconstruction error; 32-dim latent.
- **Similarity/rule features:** nearest-neighbour identity; `Mixed_Client`/`Mixed_Primer`; "same DNA_ID, different primer" checks; forward/reverse reverse-complement similarity.
- **Trace/instrument:** TRACE_SCORE, Noise_Level, PEAK_WIDTH, artefact flags, region qualities.
- **Meta flags:** `Short_Read`, `Has_Non_ACTG_Bases`.

We explicitly *remove* `Client_ID` and `Primer_ID` from classifier inputs to avoid shortcut learning and cross-plate leakage; both remain available to the rule layer and for explanations. Permutation importance checks verified that sequence-derived features retained predictive weight after these removals (see Chapter 5).

4.10 Post-hoc Rule Layer

Rules are applied after model scoring; they can *escalate* or *override* based on lab policy. We compute a weighted rule score S (sum of rule weights) and then fuse it with the model probability $\hat{p}$. Threshold values were selected empirically during validation (Chapter 5).

[1] Earlier 70/30 per-plate splits were discarded due to implicit leakage.

4.10.1 Rule inventory (excerpt)

- Reverse-complement mismatch between likely forward/reverse pair (when available).
- Same `DNA_ID` with different primers but high mismatch.
- Nearest neighbour is a different client with identity $\geq$ 66%; or a neighbourhood mixing multiple clients.
- High AE reconstruction error (per-well).
- Singleton well highly similar to a different client.
- High model confidence only (e.g., $\hat{p} > 0.90$).

4.10.2 Fusion With Model Probability

A well is flagged when any of the following holds:

$$\boxed{S \geq 1.0 \quad \vee \quad (\text{Recon_Error} > 7.5 \times 10^{-3} \wedge \hat{p} > 0.50) \quad \vee \quad (S \geq 0.5 \wedge \hat{p} > 0.75) \quad \vee \quad \hat{p} > 0.95}$$

Outputs. For plate P, Step 4 writes:

- `results/`P`/predictions_interpretable.csv` (full reasons and intermediate fields),
- `results/`P`/predictions_interpretable_labview.csv` (compact export with `Plate_Name, Well_Position, Client_ID, DNA_ID, Primer_ID, DNA_Sequence, Anomaly_Probability, Weighted_Rule_Score, Num_Reasons, Anomaly_Reason, Explanation, Anomaly_Flagged`).

We also assign an `Anomaly_Confidence` label (*High* if $S \geq 1.5$, *Medium* if $S \geq 1.0$, *Low* if $S > 0$, else *None*).

4.11 Limitations and Failure Modes

Labels. Only four labelled plates; some anomaly types are rare. **Short/ambiguous reads.** Wells < 150 bases or with `N/Y` skip AE; detection then hinges on rules and trace features. **Direction metadata.** Forward/reverse indicators are not always available; when present, reverse-complement checks are applied; otherwise omitted. **Stochasticity.** Seeds are not fixed; minor run-to-run variation is expected (variance ranges in Chapter 5 and Appendix).

4.12 Reproducibility

Processing is modular and scripted; the watcher processes plates serially and never blocks on failures:

- **Preprocessing** (trimming, ambiguity, BLAST-like neighbourhoods; 150-token windows, stride 50).
- **Autoencoder** (per-plate training; latent & reconstruction export).
- **Feature build** (merge AE, similarity, trace, and flags).
- **Classifier** (GroupKFold by `Plate_ID` during development, $\tau = 0.47$; final model retrained on all training plates).
- **Results & rules** (interpretable CSVs, LabVIEW export).

Artifacts. Each run writes to `results/{plate}/`:
`{plate}_kmer_windows.npy`, `{plate}_window_map.npy`, `{plate}_metadata.xlsx`, `{plate}_autoencoder.`
`{plate}_latent_vectors.npy`, `{plate}_with_errors.xlsx`, `{plate}_loss_curve.png`,
`features_lgbm.csv`, `predictions_lgbm.csv`, `predictions_interpretable.csv`, `predictions_interpret`
and `{plate}_layout.html`.

5

Validation & Evaluation

This chapter reports the experimental setup, metrics, baselines, and results used to validate Prometheus. We quantify performance under plate-wise cross-validation, study design trade-offs via ablations, and present per-plate case studies. It also consolidates the quantitative *evidence* for the design pivots described in Chapter 4.

5.1 Experimental Setup

Data. We evaluate on four labeled plates provided by the laboratory: `January_30_2025_p6`, `January_31_2025_p1`, `March_12_2025_p1_siso`, `March_13_2025_p4_siso`. Each plate is processed independently from Excel input (cf. Chapter 4).

Preprocessing constants. Q20 trimming (10 consecutive bases), position-aware 4-mer tokenization into 150-token windows (stride configurable, default 50), BLAST-like in-plate neighbourhoods at $\geq$ 66% identity, ambiguity flags for non-{A,C,T,G} bases. Short reads ($<$ 150) skip the AE path.

Models. Per-plate Conv1D autoencoder (sequence-to-sequence over 4-mer tokens) exports per-well latent vectors and reconstruction errors. The tabular classifier is LightGBM (features from AE, similarity, trace, and meta flags; `Client_ID`/`Primer_ID` are excluded from model inputs to avoid leakage).

Hardware. (Describe machine/CPU/GPU; include training/inference times if measured.)

5.2 Validation Protocol

Plate-wise cross-validation. We use GroupKFold by `Plate_ID` so that entire plates are held out. No wells from a test plate appear in training. After CV and hyperparameter selection, a final model is retrained on all training plates for deployment (Chapter 4).

Threshold calibration. The deployment decision threshold is set to $\tau = 0.47$ using validation folds to prioritize recall (fewer false negatives) while maintaining acceptable precision. Post-hoc fusion with rules follows Chapter 4 (fusion equation therein).

Baselines. For context, we compare against the original proposal's direction: (i) k-mer clustering + Random Forest, and (ii) a global (cross-plate) AE + MLP/XGBoost variants. (Where applicable, we report their results under the same CV protocol.)

5.3 Metrics

Given label imbalance, we report:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}$$

We also include per-plate confusion matrices, PR-AUC (preferred over ROC-AUC under imbalance), and calibration curves (reliability diagrams; Brier score).

5.4 Evidence for Design Choices

This section provides the quantitative evidence that motivated the architectural pivots in Chapter 4.

5.4.1 Step 1: BLAST-like neighbourhoods vs. DBSCAN@k-mer

Table 5.1: Neighbourhood purity (% same client within neighbourhoods), proposal vs. implemented.

Plate	DBSCAN@k-mer (proposal)	BLAST-like (implemented)
January_30_2025_p6	–	–
January_31_2025_p1	–	–
March_12_2025_p1_siso	–	–
March_13_2025_p4_siso	–	–

5.4.2 Step 2: Utility of AE latents

Table 5.2: Impact of AE latents on anomaly recall (LightGBM; plate-wise CV).

Setting	Recall	Precision
Tabular (no latents)	–	–
Tabular + AE latents	–	–

5.4.3 Step 3: Classifier comparison and leakage control

Table 5.3: Classifier comparison (plate-wise CV).

Model	**AUROC**	**AUPRC**	**F1**
Random Forest	–	–	–
XGBoost	–	–	–
LightGBM	–	–	–

5.4.4 Step 4: Rule fusion

Table 5.4: Examples from `predictions_interpretable.csv` (abridged).

Plate/Well	$\hat{p}$, S	**Reasons (excerpt)**
Jan30_p6 / D5	–, –	High similarity (≥ 66%) to different client; cross-primer consistent; AE recon error high.
Jan31_p1 / A11	–, –	Cross-client identity; neighbourhood singleton; latent anomaly.

5.5 Overall Results

Table 5.5 summarizes mean ± std across held-out plates. Detailed per-plate results follow.

Table 5.5: Overall performance under plate-wise CV (mean ± std).

Model	Precision	Recall	F1	PR-AUC
PROMETHEUS (LGBM + AE + rules)	–	–	–	–
Proposal baseline (DBSCAN + RF)	–	–	–	–
Global AE + MLP/XGB (abl.)	–	–	–	–

5.6 Per-Plate Results

For each plate we report metrics, a confusion matrix, model score distributions, and qualitative notes.

5.6.1 `January_30_2025_p6`

(Insert metrics; note large cross-client homology cluster flagged by rules + similarity features.)

5.6.2 `January_31_2025_p1`

(Include notes on cross-client identical sequences; AE recon-error histogram if helpful.)

5.6.3 `March_12_2025_p1_siso`

(Notes on multiple cross-client homologies; add plate layout screenshot if desired.)

5.6.4 `March_13_2025_p4_siso`

(Notes on multiple clusters; show PR curve or confusion matrix.)

5.7 Ablation Study

We ablate components to quantify their contributions. Table 5.6 reports per-plate metrics; qualitative effects matched those in Chapter 4 (design section).

Table 5.6: Ablations (per-plate Precision / Recall / F1).

Configuration	J30_p6	J31_p1	M12_p1_siso	M13_p4_siso
Full model (AE per plate + LGBM + rules)	–	–	–	–
No AE latents (tabular only)	–	–	–	–
No BLAST neighbourhoods	–	–	–	–
Add Client_ID/Primer_ID to model (leakage)	–	–	–	–
Global AE (not per-plate)	–	–	–	–

5.8 Error Analysis and Interpretability

We analyze false positives/negatives by inspecting: (i) AE reconstruction error vs. similarity features, (ii) post-hoc rule contributions (weighted score S), (iii) feature permutation importance and (optionally) SHAP values for LGBM. Case studies (Appendix **??**) show representative chromatograms, plate layouts, and reasons in `predictions_interpretable.csv`.

5.9 Robustness Checks

We probe sensitivity to: (a) BLAST identity threshold (e.g., 60–75%); (b) window length and stride; (c) ambiguity handling; (d) anomaly class weight; and (e) short-read prevalence. We report any notable swings in PR-AUC and recall.

5.10 Runtime & Throughput

We measure runtime per step and per plate (median ± IQR). The watcher processes plates sequentially; Step 5 runs asynchronously post-success.

5.11 Summary

PROMETHEUS achieves [insert headline result], with strongest gains from AE latents + BLAST neighbourhood features and the post-hoc rule fusion. Leakage-prone features (`Client_ID`, `Primer_ID`) are excluded from modeling to preserve generalization.

6

Planning

This chapter outlines the timeline and milestones for the development of the proposed AI-driven tool. The planning process is structured to ensure efficient progress across all stages, from initial data collection and preprocessing to model development, evaluation, and deployment. A detailed Gantt chart is provided to visualize the schedule.

6.1 Timeline & Milestones

The timeline is designed to accommodate iterative refinement and unforeseen challenges, ensuring a robust and reliable outcome.

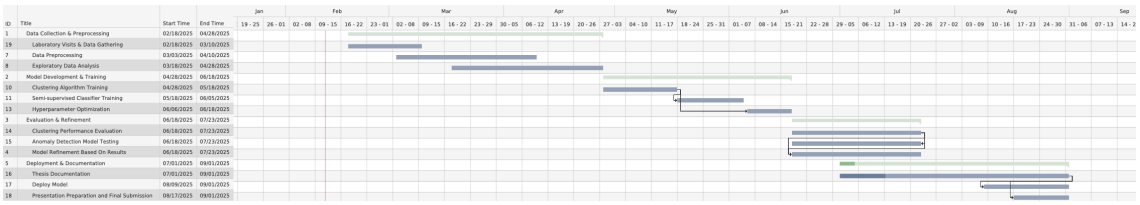

Figure 6.1: Gantt Chart of the Project Timeline

The Gantt chart (Figure 6.1) illustrates the following phases:

- **Phase 1:** Data Collection & Preprocessing (Weeks 1–10)
- **Phase 2:** Model Development & Training (Weeks 10–17)
- **Phase 3:** Evaluation & Refinement (Weeks 17–22)
- **Phase 4:** Deployment & Documentation (Weeks 19–27)

Bibliography

[1] K. Academy. *Discovery of the Structure of DNA*. Available at https://www.khanacademy.org/science/biology/dna-as-the-genetic-material/dna-discovery-and-structure/a/discovery-of-the-structure-of-dna. 2024 (cit. on p. 8).

[2] AIMultiple. *What is Overfitting in Machine Learning & How to Prevent It?* Accessed: 2025-01-29. 2024. URL: https://research.aimultiple.com/ai-overfitting/ (cit. on p. 17).

[3] M. Batista and J. Lim. "Automation in high-throughput sequencing pipelines: A review of current solutions". In: *Frontiers in Genetics* 12 (2021), p. 672568. DOI: 10.3389/fgene.2021.672568 (cit. on p. 3).

[4] A. Biosystems. *ABI 3730xl DNA Analyzer User Guide*. Accessed online. 2008. URL: https://www.thermofisher.com/document-connect/document-connect.html?url=https://assets.thermofisher.com/TFS-Assets/LSG/manuals/MAN0014293_ABI3730xl_UserGuide.pdf (cit. on p. 25).

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. on pp. 11, 12, 15, 17, 18).

[6] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324 (cit. on p. 22).

[7] E. Britannica. *Occam's Razor*. Accessed: 2025-01-28. 2025. URL: https://www.britannica.com/topic/Occams-razor (cit. on p. 18).

[8] E. Brynjolfsson and A. McAfee. "The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies". In: *W. W. Norton & Company* (2014) (cit. on p. 11).

[9] A. Chakravarthy and S. Sharma. "Optimizing Sanger sequencing workflows in genomics laboratories". In: *Bioinformatics* 36 (2020), pp. 405–412. DOI: 10.1093/bioinformatics/btz802 (cit. on p. 3).

[10] S. Chakravarthy et al. "Challenges and opportunities in DNA sequencing". In: *Genomics* 112.1 (2020), pp. 1–10 (cit. on p. 2).

[11] X. e. a. Chen. "Improving the throughput of genomic sequencing with AI-driven techniques". In: *Genomics* 111 (2019), pp. 89–96. DOI: 10.1016/j.ygeno.2018.11.009 (cit. on p. 4).

[12] Y. e. a. Chen. "Integrating AI into existing genomic workflows: A review and case study". In: *Trends in Biotechnology* 39.8 (2021), pp. 857–870. DOI: 10.1016/j.tibtech.2021.04.010 (cit. on p. 4).

[13] R. Chikhi and P. Medvedev. "Optimal K-mer Length Selection for DNA Sequence Analysis". In: *Bioinformatics* 30.9 (2014), pp. 1211–1217. DOI: 10.1093/bioinformatics/btu031 (cit. on p. 32).

[14] K. Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *arXiv preprint* (2014). arXiv: 1406.1078 (cit. on p. 13).

[15] H. Dalla-Torre et al. "Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics". In: *bioRxiv* (2023) (cit. on p. 22).

[16] J. Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 14).

[17] P. Domingos. "A Few Useful Things to Know About Machine Learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87 (cit. on pp. 11, 15, 16, 18).

[18] M. Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*. 1996, pp. 226–231 (cit. on p. 20).

[19] A. e. a. Esteva. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.7639 (2017), pp. 115–118. DOI: 10.1038/nature21056 (cit. on p. 2).

[20] B. Ewing and P. Green. "A Standard for Base Calling Accuracy in DNA Sequencing". In: *Genome Research* 8.3 (1998), pp. 186–194. DOI: 10.1101/gr.8.3.186 (cit. on p. 31).

[21] T. Fawcett. "An Introduction to ROC Analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874 (cit. on p. 11).

[22] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Springer, 2001 (cit. on p. 17).

[23] R. Fujimura and P. Schifano. "Exploring the potential of genomics for precision medicine". In: *Nature Genetics* 47 (2015), pp. 323–329. DOI: 10.1038/ng.3212 (cit. on p. 3).

[24] S. Geman, E. Bienenstock, and R. Doursat. "Neural networks and the bias/variance dilemma". In: *Neural Computation* 4.1 (1992), pp. 1–58 (cit. on pp. 15, 17).

[25] K. e. a. Gokcay. "Leveraging machine learning to enhance sequencing anomaly detection". In: *Biotechnology Advances* 37.1 (2019), pp. 95–102. DOI: 10.1016/j.biotechadv.2018.11.004 (cit. on p. 4).

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613 (cit. on p. 11).

[27] E. D. Green, J. D. Watson, and F. S. Collins. "The Human Genome Project: Lessons from Large-Scale Biology". In: *Science* 348.6236 (2015), pp. 36–37 (cit. on p. 1).

[28] I. Guyon and A. Elisseeff. "An Introduction to Variable and Feature Selection". In: *Journal of Machine Learning Research* 3 (2003), pp. 1157–1182 (cit. on p. 11).

[29] J. Han and M. Kamber. "A Comparative Study of Normalization Techniques for Machine Learning". In: *Data Mining: Concepts and Techniques* 3 (2011), pp. 2–4 (cit. on p. 32).

[30] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Elsevier, 2011 (cit. on p. 11).

[31] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009 (cit. on p. 15).

[32] K. He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778 (cit. on p. 15).

[33] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313 (2006), pp. 504–507. DOI: 10.1126/science.1127647 (cit. on p. 14).

[34] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 13).

[35] N. H. G. R. Institute. *The Human Genome Project*. https://www.genome.gov/human-genome-project. Accessed: 2025-01-28. 2023 (cit. on p. 7).

[36] G. James et al. *An Introduction to Statistical Learning*. Springer, 2013. ISBN: 9781461471370 (cit. on pp. 11, 13).

[37] Y. Ji et al. "DNABERT: pre-trained bidirectional encoder representations from transformers for DNA-language in genome". In: *Bioinformatics* 37.15 (2021), pp. 2112–2120 (cit. on p. 22).

[38] L. e. a. Jiang. "AI in genomic anomaly detection". In: *Journal of Bioinformatics and Computational Biology* 19.5 (2021), p. 1750012. DOI: 10.1142/S0219720021500127 (cit. on p. 3).

[39] I. Jolliffe. *Principal component analysis*. Springer, 2002 (cit. on p. 16).

[40] I. T. Jolliffe and J. Cadima. "Principal Component Analysis in Genomics: Theory and Application". In: *Philosophical Transactions of the Royal Society A* 374.2065 (2016), p. 20150202. DOI: 10.1098/rsta.2015.0202 (cit. on p. 32).

[41] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 22).

[42] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv preprint arXiv:1609.02907* (2016) (cit. on p. 22).

[43] R. Kohavi. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 14. 1995, pp. 1137–1145 (cit. on p. 16).

[44] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. "Data Preprocessing for Supervised Learning". In: *International Journal of Computer Science* 1.2 (2006), pp. 111–117 (cit. on p. 11).

[45] R. Kumar and H. Verma. "Reducing human error in genomic sequencing through automation". In: *Journal of Genomic Medicine* 13 (2018), pp. 38–45. DOI: 10.1007/jgm.2018.38 (cit. on p. 3).

[46] S. Kumar et al. "Sequencing technologies and their applications in evolutionary studies". In: *Nature Reviews Genetics* 19.6 (2018), pp. 385–396 (cit. on p. 2).

[47] E. S. Lander et al. "Initial sequencing and analysis of the human genome". In: *Nature* 409.6822 (2001), pp. 860–921 (cit. on p. 7).

[48] Y. LeCun, Y. Bengio, and G. Hinton. "Deep Learning". In: *Nature* 521 (2015), pp. 436–444. DOI: 10.1038/nature14539 (cit. on p. 13).

[49] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86 (1998), pp. 2278–2324 (cit. on p. 13).

[50] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation forest". In: *2008 Eighth IEEE International Conference on Data Mining* (2008), pp. 413–422 (cit. on p. 19).

[51] P. P. Mandal, R. Rezaee, and I. Emelyanova. *Graphical Illustration of Bias and Variance*. Accessed: 2025-01-29. 2021. URL: https://www.researchgate.net/figure/a-Graphical-illustration-of-bias-and-variance-The-aim-is-to-obtain-a-model-that-has_fig3_357411611 (cit. on p. 18).

[52] E. R. Mardis. "DNA sequencing technologies: 2006–2016". In: *Nature Protocols* 12.2 (2017), pp. 365–368 (cit. on pp. 7, 9).

[53] E. R. Mardis. "Clinical applications of new DNA sequencing technologies". In: *Nature Reviews Genetics* 11.10 (2010), pp. 687–696 (cit. on p. 9).

[54] A. M. Maxam and W. Gilbert. "A new method for sequencing DNA". In: *Proceedings of the National Academy of Sciences* 74 (1977), pp. 560–564. DOI: 10.1073/pnas.74.2.560 (cit. on p. 9).

[55] M. L. Metzker. "Sequencing technologies—the next generation". In: *Nature Reviews Genetics* 11.1 (2010), pp. 31–46 (cit. on pp. 7, 9).

[56] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. ISBN: 9780262018029 (cit. on p. 13).

[57] P. Neurology. *Understanding Data in AI*. https://premierneurology.org/?p=278055216. Accessed: 2025-01-27. 2025 (cit. on p. 12).

[58] C. O'Neil. *Data Science for Mathematicians*. Springer, 2017 (cit. on p. 11).

[59] L. Prechelt. "Automatic early stopping using cross validation: quantifying the criteria". In: *Neural Networks* 11.4 (1998), pp. 761–767 (cit. on p. 16).

[60] F. Provost and T. Fawcett. *Data Science for Business: What You Need to Know about Data Mining and Data-analytic Thinking*. O'Reilly Media, Inc., 2013 (cit. on p. 11).

[61] Quantib. *How Does Deep Learning Work in Radiology?* Accessed: 2025-01-27. 2023. URL: https://www.quantib.com/blog/how-does-deep-learning-work-in-radiology (visited on 2025-01-27) (cit. on p. 14).

[62] P. Rajendran and R. Gupta. "Scalable solutions for genomic sequencing with AI". In: *Nature Biotechnology* 39 (2021), pp. 1593–1601. DOI: 10.1038/s41587-021-00450-x (cit. on p. 4).

[63] B. Reality. *Sanger Sequencing Steps and Process*. Available at https://www.biotechreality.com/2024/10/sanger-sequencing-steps-and-process.html. 2024 (cit. on p. 10).

[64] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Prentice Hall, 2010 (cit. on p. 10).

[65] F. e. a. Sanger. "DNA sequencing with chain-terminating inhibitors". In: *Proceedings of the National Academy of Sciences* 74.12 (1981), pp. 5463–5467. DOI: 10.1073/pnas.78.12.5463 (cit. on p. 3).

[66] F. Sanger, S. Nicklen, and A. R. Coulson. "DNA sequencing with chain-terminating inhibitors". In: *Proceedings of the National Academy of Sciences* 74 (1977), pp. 5463–5467 (cit. on pp. 9, 31).

[67] A. e. a. Senior. "Improved protein structure prediction using potentials from deep learning". In: *Nature* 577.7792 (2020), pp. 706–710. DOI: 10.1038/s41586-019-1923-7 (cit. on p. 2).

[68] A. Sharma, A. Sharma, and S. Singh. "A brief overview of game-changing genetic engineering projects". In: *Journal of Biomedical Research* 37.1 (2023), pp. 1–7. DOI: 10.1097/JBR.0000000000000142. URL: https://spj.science.org/doi/10.1097/JBR.0000000000000142 (cit. on p. 8).

[69] J. Shendure et al. "DNA sequencing at 40: Past, present, and future". In: *Nature* 550 (2017), pp. 345–353. DOI: 10.1038/nature24286 (cit. on pp. 1, 9).

[70] J. Sheng and S. Lee. "Artificial intelligence in genomics research: Opportunities and challenges". In: *Journal of Computational Biology* 25.6 (2018), pp. 553–561. DOI: 10.1089/cmb.2018.0201 (cit. on p. 4).

[71] C. Shorten and T. M. Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data* 6.1 (2019), pp. 1–48 (cit. on pp. 15, 16).

[72] L. M. Smith, J. Z. Sanders, R. J. Kaiser, et al. "Fluorescence detection in automated DNA sequence analysis". In: *Nature* 321.6071 (1986), pp. 674–679. DOI: 10.1038/321674a0 (cit. on p. 25).

[73] N. Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on p. 16).

[74] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018. ISBN: 9780262039246 (cit. on p. 13).

[75] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977 (cit. on p. 11).

[76] R. K. Varshney, R. Terauchi, and S. R. McCouch. "Genomics-assisted breeding for crop improvement". In: *Trends in Plant Science* 19.2 (2014), pp. 98–106 (cit. on p. 1).

[77] A. Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017) (cit. on p. 14).

[78] J. C. Venter et al. "The sequence of the human genome". In: *Science* 291.5507 (2001), pp. 1304–1351 (cit. on p. 7).

[79] P. Vincent et al. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *Journal of Machine Learning Research* 11 (2010), pp. 3371–3408 (cit. on p. 14).

[80] J. D. Watson and F. H. C. Crick. "Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid". In: *Nature* 171 (1953), pp. 737–738 (cit. on p. 8).

[81] J. e. a. Wu. "AI-driven solutions for genomic anomaly detection in sequencing pipelines". In: *Computational Biology and Chemistry* 87 (2020), p. 107274. DOI: 10.1016/j.compbiolchem.2020.107274 (cit. on p. 4).

[82] Z. e. a. Yang. "Automation of genomic data analysis: A comprehensive review". In: *Bioinformatics Advances* 1.1 (2020), pp. 14–23. DOI: 10.1093/bioadv/vzab015 (cit. on p. 4).

[83] L. e. a. Zhang. "AI in high-throughput sequencing workflows: A case study in genomic anomaly detection". In: *Nature Communications* 12 (2021), p. 2567. DOI: 10.1038/s41467-021-22865-9 (cit. on p. 4).