# PATIENT-BASED REAL-TIME QUALITY CONTROL

*A Comparison between Time Series Machine Learning Technique and Traditional Statistical Techniques*

BY: ZI EN THAM

BACHELOR OF COMPUTING SCIENCE (HONOURS) MAJOR IN ARTIFICIAL INTELLIGENCE AND DATA ANALYTICS

SUPERVISOR: DR. XU WANG

30 MAY 2024

# Table of Contents

## List of Figures



## List of Tables

# Abstract

Patient-based real-time quality control (PBRTQC) is integral to maintaining the daily accuracy of laboratory tests, ensuring patient safety by minimizing risks such as misdiagnosis and the need for sample reanalysis. Traditionally, PBRTQC relies on variants of the Average of Normal (AON) statistical technique, which, while effective, may have limitations in capturing complex patterns. This study explores the potential of machine learning techniques, particularly Long Short-Term Memory (LSTM), a type of Recurrent Neural Network renowned for its ability in time sequence analysis. Given the inherent challenge of labelling anomalies in real-world scenarios, an unsupervised learning approach is adopted to identify anomalies within the machine. The primary objective is to investigate whether LSTM can surpass or perform comparably to traditional statistical techniques, serving as a proof of concept for its applicability in PBRTQC. A simple moving average serves as a baseline model, while LSTM models with and without Moving Average preprocessing are developed to facilitate comparison. Focusing specifically on creatinine and sodium as analytes, each analyte is trained independently to assess their individual performance. Anomalies are simulated through sudden shifts in value, providing a controlled environment for evaluating model effectiveness. The results indicate that LSTM with Moving Average generally outperforms other scenarios, although challenges arise, particularly when encountering ±5% shifts in value, notably in the case of sodium. Nonetheless, it suggests promising avenues for leveraging machine learning techniques in enhancing PBRTQC processes and improving patient care outcomes.

# Acknowledgement

I would like to extend my deepest gratitude to Dr. Xu Wang for his unwavering support and encouragement throughout my honours year. I also wished to express my sincere appreciation to Yusof Rahman from WestMead Hospital for providing the essential datasets and medical expertise that were invaluable to the development of this thesis. Lastly, I am grateful for the moral support of my parents and friends, which has been a cornerstone of my entire academic journey.

# Chapter 1

This chapter describes the background of the study and sets the motivation of the study.

## Introduction

In the realm of modern healthcare, clinical laboratories play a pivotal role in the diagnostic process, providing invaluable insights through the analysis of patient blood samples. These laboratories are entrusted with the task of generating accurate and reliable results, which serve as the foundation for medical practitioners to make informed decisions regarding patient care. To ensure the accuracy of these results, quality control procedures are essential, periodically checking the performance of the blood analysis machines. Traditionally, some hospitals have adopted a time-based approach, mandating quality control checks at fixed intervals, irrespective of whether the machines produce erroneous results or not.

However, in the evolving landscape of medical technology and data-driven decision-making, clinical laboratories are increasingly integrating advanced techniques, such as Average of Normal (AON), into their diagnostic protocols. This innovative approach employs statistical analysis to monitor consecutive blood sample results, promptly raising a red flag when anomalies in the data surface. Such quality control approach is generally known as the Patient-Based Real Time Quality Control (PBRTQC). These anomalies can signify potential issues with the machine, ranging from calibration inaccuracies to mishandling of samples by practitioners. The adoption of the statistical technique has ushered in a new era of efficiency, as it empowers lab practitioners to execute quality control procedures only when the system detects irregularities. This, in turn, leads to significant cost savings and minimizes the delay in delivering crucial diagnostic information, preventing the need for revaluating previously reported results.

While the implementation of statistical technique has demonstrated its utility in enhancing the quality control process, an intriguing question arises: Can recurrent neural networks (RNNs), a class of machine learning algorithms renowned for their ability to capture patterns in sequential data, outperform traditional statistical techniques like Moving Average (MA) in this context? By leveraging the power of machine learning, this research aims to ascertain whether RNNs can indeed surpass the performance of Moving Average, ultimately contributing to the refinement of quality of blood sample analysis and, consequently, the enhancement of patient care.

## Background

As described by Badrick et al. (2020), PBRTQC uses the statistical characteristics of a particular patient population(s) served by a laboratory using specific analytical platforms. Specifically, Cembrowski et al. (1984), introduces AON as one of the earliest statistical techniques used in PBRTQC. The core idea behind AON is that the majority of patient samples typically fall within an anticipated range. Consequently, the average value obtained from a representative sample over a specific period should closely align with the expected average for that particular population. An indication of a potential issue with the machine arises when a new sample deviates significantly from this mean value. Subsequently, various adaptations and variations of the AON concept have emerged in the field.

In this research, MA, one of the variants, will be used as a baseline for comparison of the performance of RNN. Unlike AON, MA constantly recalculated the average as new patient results become available. When a new result is added, the oldest result that was produced a certain number of samples ago (known as the block size) is removed. The rules for calculating the moving average depend on specific criteria, such as which patient results to include and how many. The user can set parameters like the block size, control limits, and truncation limits (concentration levels at which results are excluded). If the moving average goes beyond the predefined control limits for a particular analyte, it triggers a flag.

However, a drawback of the moving average technique is that depending on the error's magnitude, the assay's precision, and the distribution of the patient population, several patient results may be recorded on the chart before identifying the error. Hence, in this research, it is hypothesized that RNN could outperform statistical techniques.

## Aims

The primary aim of this research is to investigate and compare the performance of RNN and traditional statistical techniques, specifically the MA in the context of PBRTQC for clinical laboratory blood sample analysis. It is hoped to learn the limitations and challenges of RNN through this study. Subsequently. the research seeks to aims to determine whether RNNs can offer superior capabilities in detecting errors or deviations in testing results and enable timely corrective actions to be taken compared to MA.

## Objectives

In this section, a set of objectives is listed out to provide a holistic assessment of the feasibility and utility of RNNs in enhancing the quality control processes within clinical laboratories. With that, comprehensive understanding of the potential advantages and limitations of RNNs in comparison to the traditional statistical technique of Moving Average (MA) can be achieved.

- To develop a time-series neural network model and fine-tuning it for a PBRTQC simulation.
  This objective involves the creation of a time-series neural network model tailored for the PBRTQC task in clinical laboratories. Fine-tuning this model ensures it reached its optimal performance for such task.

- To develop statistical based model using MA approach as the baseline for comparison in the PBRTQC simulation.
  This objective will construct a statistical model using the MA approach as a baseline for comparison within the PBRTQC task. This model serves as a benchmark against which the performance of the RNN can be evaluated, providing a reference point for our analysis.

- To evaluate the effectiveness of Moving Average (MA) as a traditional statistical technique in PBRTQC.
  This objective aims to assess the benefits and performance of MA in the PBRTQC task. This allows an understanding of how MA works in clinical laboratory.

- To assess the performance of RNN model in detecting anomalies and deviations from expected patterns in patient blood sample results.
  This assessment provides insights into the potential advantages of utilizing advanced machine learning techniques in PBRTQC.

- To compare the efficiency and accuracy of RNN-based quality control with MA-based quality control in terms of error detection and prevention.
  It aims to quantify the potential improvements that RNNs may offer over traditional statistical methods. Otherwise, an investigation of potential reasons contributing to its inferior performance compared to MA needs to be conducted.

- To analyze the factors influencing the performance of both RNN and MA techniques, including error magnitude, assay precision, and patient population distribution.
  Understanding these influences will provide a nuanced perspective on the strengths and weaknesses of each approach.

- To explore the potential benefits of adopting RNNs over traditional statistical methods in terms of cost savings and time efficiency in clinical laboratory operations.
  While RNN may have better accuracy, there is a need to look into the cost and time feasibility of employing such model in the PBRTQC task.

## Research Questions

Below is a list of research questions that are needed to be answered throughout this study. These research questions will serve as a guideline to achieve the identified objectives and aim.

1. How can a time-series neural network model be developed and fine-tuned for the PBRTQC task in clinical laboratory haematology area?
2. What is the effectiveness of MA as a traditional statistical technique in PBRTQC for clinical laboratory quality control?
3. How does the performance of the RNN model compare to that of the MA approach in detecting anomalies and deviations from expected patterns in patient blood sample results within a PBRTQC context?
4. What are the efficiency and accuracy differences between RNN-based quality control and MA-based quality control in terms of error detection and prevention in PBRTQC?
5. What factors influence the performance of both RNN and MA techniques in PBRTQC, including error magnitude, assay precision, and patient population distribution?
6. What potential benefits, such as cost savings and time efficiency, can be realized by adopting RNNs over traditional statistical methods in the context of clinical laboratory operations for PBRTQC?

## Argument of the Study

This research hypothesizes that machine learning, a field that centers on utilizing data and algorithms to build learning systems tailored to specific tasks. It mimics human learning processes and steadily enhances its accuracy over time. Moreover, in today's world, machine learning has gained extensive application across various domains and has demonstrated favourable outcomes in its implementation. Unlike traditional statistical methods, machine learning leverages acquired information to generate predictions or classifications, thereby informing decision-making processes for unforeseen challenges in the future.

Therefore, this research claims that RNN, a subset of machine learning, could potentially outperform statistical techniques like the MA in the PBRTQC task. Furthermore, the research contends that RNNs have the capacity to adapt to varying error magnitudes, assay precision levels, and patient population distributions, thereby addressing the limitations of MA.

## Future Impact/Significance

The potential outcomes of this research offer multiple benefits to healthcare practices. Should the efficacy of the RNN architecture surpass that of traditional statistical methods, the implications could be profound, leading to substantial enhancements in diagnostic accuracy and, consequently, patient care. Such a breakthrough could usher in a transformation in the operational procedures of clinical laboratories, bolstering their capacity to yield dependable and precise results.

Ultimately, the impact of the research extends to the patient level. More accurate and efficient quality control processes in clinical laboratories have the potential to directly improve patient outcomes. The implementation of more precise and efficient quality control procedures within clinical laboratories has the potential to yield tangible improvements in patient outcomes. Patients stand to gain from more reliable diagnoses, ultimately facilitating more timely and efficacious medical interventions.

Nevertheless, it is crucial to acknowledge that this research serves as an exploratory endeavour into the utilization of RNNs within the context of PBRTQC. Various considerations and factors may necessitate thorough examination before advocating for the widespread implementation of RNNs in this capacity. Nonetheless, the findings from this research carry the potential to stimulate further studies and advancements at the intersection of machine learning and healthcare. This may inspire other researchers to delve into uncharted territories, exploring novel applications of RNNs and similar machine learning techniques in the domains of healthcare quality control and diagnostics. This ripple effect could potentially trigger a surge of innovative research and practical implementations, perpetuating the ongoing enhancement of the healthcare industry.

# Chapter 2

This chapter encompasses a review of related literature aimed at deepening the comprehension of the present study.

## Understanding Moving Average in Laboratory Quality Control

This subsection describes one of the mainstream methodologies used in the PBRTQC field, which is the MA technique. A mainstream methodology in this context is a research method that is widely accepted and commonly used in most studies on PBRTQC.

Similar to AON, MA uses an unweighted patient average to evaluate the analytical performance of a test. However, MA is continuously recalculated when each new eligible patient result is produced. Anomaly is detected if the moving average exceeds the analyte-specific control limits. According to Van Rossum and Kemperman (2017), the advantage of the MA is that patient results are continuously released. This means that it depends on the number of affected patients' samples before the detection of the system error condition. To minimise the number of affected samples that would require amendments due to detected anomaly, the earliest result in the MA block is released only after n+1 results have been analysed and the block of results has been found to be within the analyte-specific control limits. The disadvantage of this technique is that, depending on the magnitude of the error, the analytical imprecision of the assay and the population distribution, numerous patient results may be posted to the chart before the error is detected.

However, (Fleming & Katayev, 2015) who introduced "release from the back" strategy to minimise the number of affected samples that would require corrections due to the nature of MA. This strategy states that the earliest result within the MA block is released only after the moving block passed the control limits. With that, this strategy effectively isolates patient results, ensuring they undergo MA quality control before being released.

In addition, Spies et al. (2023) mentioned that most of the data in the medical laboratory are unlabelled. The paper theorises that historical calibration data could be used to train a reinforcement learning agent. The paper also suggested that supervised machine-learning algorithms can be applied to extract hidden patterns in the data if high quality labels can be generated. These algorithms use predefined classifications and mathematical transformations to make predictions based on the labelled data. Semi-supervised learning and generative models can help bridge the gap between labelled and unlabelled data. However, their performance may only partially reflect actual patient biology and is difficult to validate.

To conclude this subsection, MA can help smooth out fluctuations in data and identify trends over time. However, MA is only sometimes effective in detecting anomalies in PBRTQC because it is not designed to capture the complex patterns and dependencies that can exist in time-series data.

Therefore, this research explores the usage of sophisticated ML algorithms, such as RNN, which is better suited to detecting anomalies in PBRTQC by learning from the historical patterns and relationships in the data. RNN is usually used for supervised learning, but it can also be used as semi-supervised or unsupervised learning, depending on the resource available and experiment settings.

## The Potential and Limitation of Long-Short Term Memory

According to (Bowie et al., 2017), anomaly detection techniques often struggle to detect complex anomalies beyond point anomalies due to the challenges posed by multivariate and temporally heterogeneous data. Labelled data primarily represents normal instances, necessitating semi-supervised or unsupervised techniques. Detecting various anomaly types—point, collective, and contextual—typically requires employing multiple techniques. Contextual anomalies, which often require domain expertise for interpretation, are particularly time-intensive. For instance, in medical data like electronic health records (EHRs), temporal context is crucial; a low sodium reading may be normal for a patient with a history of large intestine removal. Neglecting time in anomaly detection can lead to misinterpretations. To address these challenges, there is a need for improved anomaly detection methods in medical data capable of identifying all types of anomalies. Hence, the paper suggested solution involves employing neural network-based approaches, specifically LSTM networks, which could automate anomaly detection tasks effectively.

With that, the purpose of using LSTM is due to its for detecting both data quality errors and various types of anomalies, including temporal anomalies, in large datasets. For instance, an LSTM can learn to differentiate between normal and anomalous data points based on past events, such as a patient's medical history, and adjust its predictions accordingly. This ability to adapt to changing contexts makes LSTMs particularly well-suited for anomaly detection tasks in diverse datasets. However, this paper is still on-going research and the results has yet to be determined.

(Maleki et al., 2021) mentions that anomaly detection in time-series data presents a significant challenge in various industries due to the proliferation of sensor technologies. Researchers have focused on developing intelligent agents to address this challenge and enable remote monitoring. However, these agents typically rely on algorithms that requires offline training on labelled data, which is resource-intensive. Moreover, they often lack adaptability across different domains and require balancing accuracy with false alarm rates, especially in complex systems with multiple operation modes. Despite the growing volume of data, the scarcity of suitable training examples remains a major hurdle for anomaly detection. Unsupervised learning schemes, such as autoencoders, have emerged as effective solutions. Autoencoders, including variants using Long Short-Term Memory (LSTM) networks, learn latent representations of data and detect anomalies based on reconstruction errors. Yet, achieving significant improvements in detection accuracy often demands complex LSTM architectures with substantial memory requirements and computational complexity.

(Malhotra et al., 2015) investigates the potential of LSTM neural networks in anomaly detection within the context of process monitoring. Traditionally, process monitoring relies on statistical methods over fixed time windows, which may not effectively capture complex temporal patterns. LSTM networks address this limitation by enabling long-term memory storage without predefined time windows, making them suitable for modelling multivariate sequences without pre-processing. The study demonstrates that stacked LSTM networks can accurately detect deviations from normal behaviour without prior knowledge of pattern duration or pre-processing steps. Additionally, the study explores the combination of LSTM's retentive power with recurrent hierarchical processing layers for predicting time series and detecting anomalies, offering a novel approach to anomaly

detection in real-world scenarios. Results indicate LSTM's potential superiority over traditional methods, particularly in scenarios with uncertain long-term dependencies in normal behaviour, showcasing its viability for process monitoring and anomaly detection applications.

This study investigates the use of machine learning-based anomaly detection methods in vertical plant wall systems to enhance indoor climate control through predictive maintenance. The research focuses on identifying two types of anomalies: point anomalies and contextual anomalies. The dataset used in the study is a univariate dataset. Various prediction-based and pattern recognition-based methods were evaluated, with neural network-based models, specifically autoencoders (AE) and long short-term memory encoder-decoder (LSTM-ED) models, proving to be the most effective. Autoencoders excelled in detecting point anomalies, while LSTM-ED models were superior in identifying contextual anomalies. However, the study addresses several concerns such as the performance of the model heavily relies on training data, which can be affected by many factors such as the sampling rate and the size of dataset (Liu et al., 2020).

(Hu et al., 2019) focuses on predicting tunnel surface settlement to prevent disasters like tunnel collapses and landslides, which is crucial for ensuring safety in both urban and rural construction projects. It evaluates different methods for this prediction, highlighting two main categories: model-based methods and AI-enhanced methods. The study particularly examines the limitations of using deep learning neural networks (DLNN), specifically long short-term memory (LSTM) networks, for this purpose. LSTM is deemed unsuitable because tunnel construction projects often generate small and short-length datasets, and LSTM networks typically require extensive historical data to produce accurate forecasts. In the context of clinical laboratory settings, similar challenges may arise if the available data across months is not sufficiently extensive or comprehensive. LSTM networks might struggle with limited datasets, which could impact the accuracy and reliability of anomaly detection in blood sample results.

# Chapter 3

This chapter describes the datasets and main methodologies used in this study.

## Population and Sample Description

This research centers on an extensive dataset comprising patient records, specifically capturing the results of various analytes collected during the period from January 2023 to March 2023. The dataset originates from Westmead Hospital, with the patient's personal information meticulously anonymized, retaining only the Medical Record Number (MRN) for identification. This comprehensive dataset encompasses a total of 440,093 records and encompasses key analytes such as sodium, creatinine, and calcium. It is important to emphasize that data has been collected, recorded, and verified. Hence, it is assumed that the analytes' result would not consist of any error.

The patient population in this dataset reflects a wide age range, encompassing individuals from infancy to those over 100 years old. Furthermore, the blood samples of these patients have been sourced from an array of healthcare facilities and wards, including, but not limited to, cancer treatment wards, emergency wards, and various other relevant clinical settings. This diversity underscores the representativeness and richness of the dataset under examination.

Overall, the dataset can be summarised as the table below (*Table 1*):

| **Attribute Name** | **Data Type** | **Description** |
|---|---|---|
| Facility | String | The location from which the sample is collected. |
| Ward | String | The specific ward within the facility from which the sample was collected. |
| MRN | Integer | A unique identifier of a patient used in clinical settings |
| Age | Integer | The age of the patient |
| Service resource | String | The machine used for analysing the sample |
| Accessin | Integer | The unique identifier assigned to the specimen |
| DTA | String | The specific analyte that is being analysed and tested |
| Results | Float | The result of the analyte |
| Ord | DateTime (yyyy-mm-dd hh-mm-ss) | The order date for the test |
| Coll | DateTime (yyyy-mm-dd hh-mm-ss) | The sample collection date |
| Recorded | DateTime (yyyy-mm-dd hh-mm-ss) | The recorded test result date |
| Verified | DateTime (yyyy-mm-dd hh-mm-ss) | The date of the test result being verified |

*Table 1: Data Description*

As depicted in *Figure 1*, the dataset's age distribution presents some intriguing findings. Notably, the records include a substantial number of patients falling into two extreme age groups: infants and individuals aged over 100 years. Specifically, there may be record error for individuals who are aged over 120 years old, thus these records will need to be removed later. However, the majority of patients in the dataset fall within the age range of 60 to 80 years.

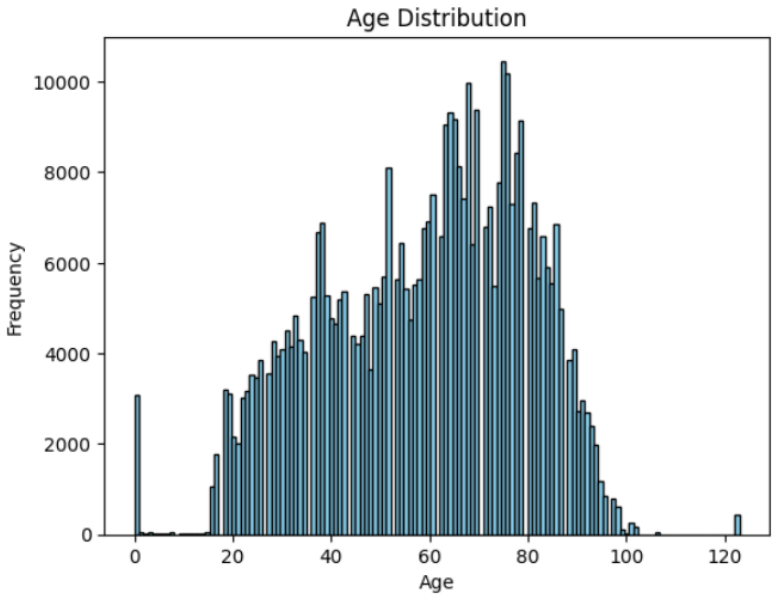


*Figure 1: Age Distribution*

This study focuses on two key analytes: sodium and creatinine. The accompanying figure (*Figure* 2) illustrates the data distribution of these analytes. Creatinine exhibits a positively skewed distribution with a short tail and a tall head, while sodium follows a normal distribution characterized by an exceptionally high peak.

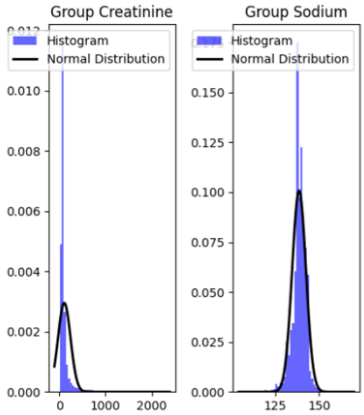


*Figure 2: Creatinine and Sodium Distribution*

*Figure 3* depicts the trends in sodium and creatinine over time. This is to provide a quick overview of the expected ranges for sodium and creatinine results.

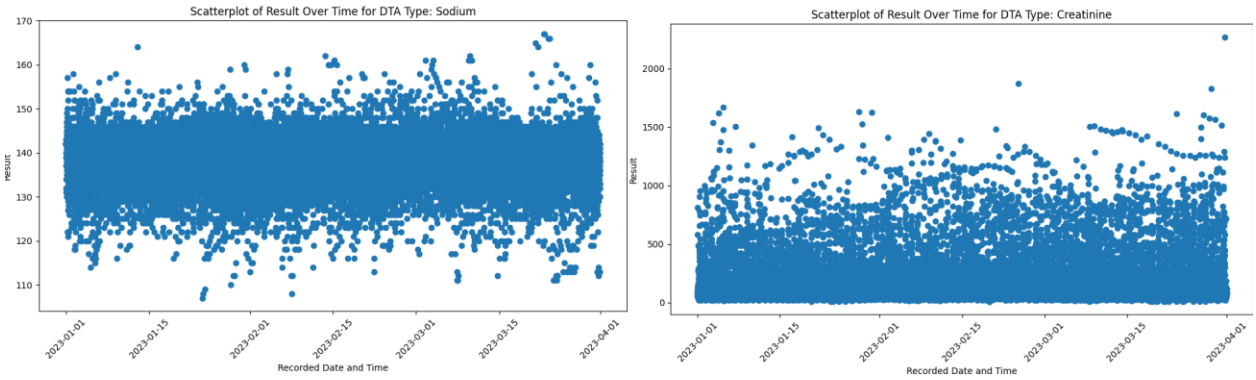


*Figure 3: Sample Patients' Sodium and Creatinine Result*

*Figure 4* illustrates the variations in sodium and creatinine levels across different age groups. Typically, patients aged between 20 and 40 tend to have sodium levels within the 120 to 150 range, while those aged over 80 often exhibit sodium levels exceeding 150. In contrast, creatinine levels are generally lower than 500 for patients under the age of 19.

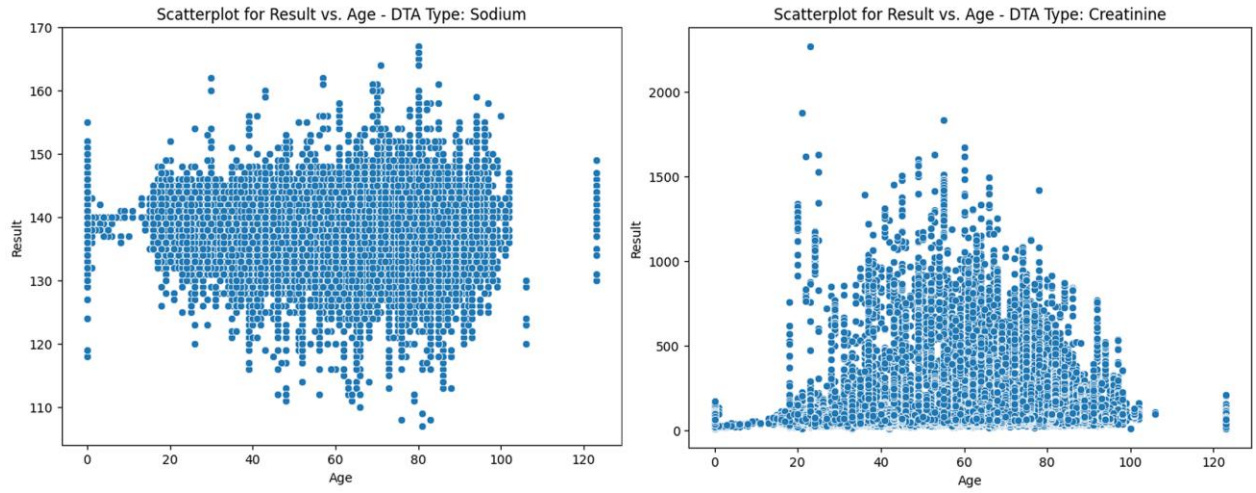


*Figure 4: Sodium and Creatinine Distribution on Different Age*

# Methodology

For the baseline model, MA is the main statistical technique that will be used. A simple MA calculation could be seen as below (Bietenbeck et al., 2020):

$$MA_t = \frac{x_{1t} + x_{2t} + \cdots + x_{nt}}{n_t}$$

$MA$ refers to the Moving Average, $t$ refers to the batch numbers, $x$ refers to the data, and $n$ refers to the block size. Each analyte will have their own MA calculations and control limits. Control limits are thresholds or bounds set to determine whether a process or measurement is within an acceptable range. They are used to monitor the variation in a process and identify when it goes out of control. Control limits help detect unusual patterns or deviations that may indicate a problem.

In the domain of laboratory settings, various control limit techniques are outlined in (Bietenbeck et al., 2020), including symmetric control limits, overall percentiles control limits, and percentiles of daily extremes. However, for the sake of simplicity, this study focuses solely on implementing symmetric control limits which makes the upper and lower control limits to have equal distance based on the mean of the datasets. The calculation process for symmetric control limits is outlined as follows:

$$Upper\ Control\ Limit\ (UCL) = \bar{x} + (threshold\ \times standard\ deviation)$$
$$Lower\ Control\ Limit\ (LCL) = \bar{x} - (threshold\ \times standard\ deviation)$$

Where $\bar{x}$ refers to the mean of a series of data, and threshold is defined through trial and error. The standard deviation is calculated relative to the mean of the data series.

A standard LSTM (Long Short-Term Memory) model derived from a RNN architecture. It is designed to process sequential data effectively while mitigating the vanishing gradient problem. Central to the LSTM structure are memory cells, which store and manage information over extended sequences. The model incorporates three gates—the input, forget, and output gates—to control the flow of information, using sigmoid and tanh activations to manage information retention and output. The cell state retains memory, while the hidden state delivers predictions. LSTM models are trained through backpropagation through time, enabling them to adapt gate weights to store and retrieve information optimally. *Figure 5* shows the basic architecture of LSTM.

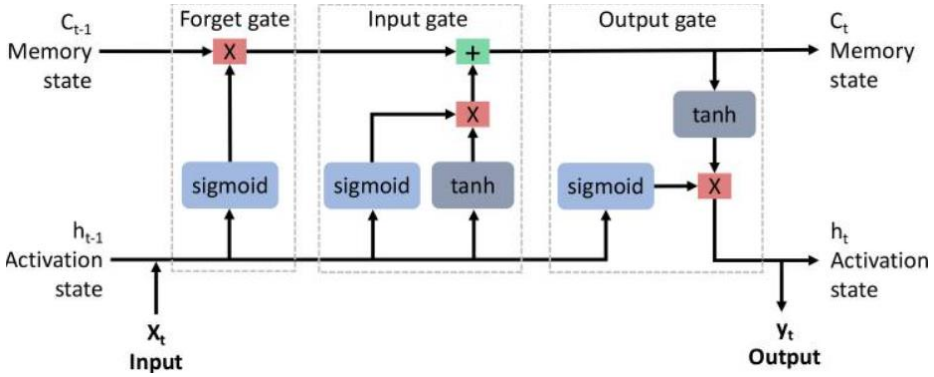


*Figure 5: LSTM Architecture (Kiser et al., 2023)*

The formula for a LSTM cell can be seen below (Staudemeyer & Morris, 2019):

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Where $x_t$ is the input at time step $t$ and $h_{t-1}$ is the hidden state from the previous time step. The cell state from the previous time step is denoted as $C_{t-1}$. $W_f, W_i, W_C, W_o$ are the weight matrices and $b_f, b_i, b_C, b_o$ are bias vectors. $\sigma$ refers to the sigmoid function and $*$ refers to the element wise multiplication.

The formula first starts of with forget gate that decides what information to throw away from the cell state. Then, the input gate decides which values to update. The third equation refers to the candidate cell state where it creates a vector of new candidate values that could be added to the state and subsequently updates the old cell state $C_{t-1}$ into the new cell state $C_t$. Finally, the output gate decides what the next hidden state $h_t$ should be. The last equation shows how the hidden state is calculated.

Based on the LSTM cell architecture, this study explores the use of stacked LSTM cells to create an autoencoder model. The amalgamation of sequential data processing capabilities, long-term memory management, gate mechanisms, and specialized training algorithms makes LSTM well-suited for unsupervised anomaly detection tasks. In this study, unlike supervised learning approaches, only normal data without anomalies will be trained. This approach enables the model to learn normal patterns and reconstruct them. Anomalies are identified by calculating the mean square error loss between the reconstructed and actual data. If the loss exceeds a predefined threshold, the data point at that time is classified as anomalous. This methodology allows for effective anomaly detection without the need for labeled anomaly data, demonstrating the versatility and effectiveness of LSTM networks in unsupervised settings. The formula for the calculation of anomalous points using mean squared error (MSE) loss and a threshold is as follows:

Let $X$ be the original sequences, $\hat{X}$ be the reconstructed sequences, and $mse\_loss$ be the mean squared error loss for each data point.

$$mse\_loss_i = \frac{1}{N} \sum_{j=1}^{N} (X_{ij} - \hat{X}_{ij})^2$$

Where $i$ indexes the sequences, $j$ indexes the time steps within each sequence, and $N$ is the total number of time steps.

Anomalous points are identified based on the MSE loss exceeding a predefined threshold:

$$anomaly = \{i \mid mse\_loss_i > threshold\}$$

Where $i$ represents the index of the sequence with MSE greater than the threshold. The threshold is defined through trial and error.

# Chapter 4

This chapter lists the experiment settings of this study.

## Development Environment

In this context, a development environment signifies the platform where the code for creating the models will be executed.

The project environment will be set up in iHPC from University of Technology, using Python as the main programming language. iHPC is an Interactive High-Performance Computing facility that provides an interactive high-performance computing resource for all researchers within the university.

## Introduction of Biases

The types of anomalies that is going to be simulated is a sudden shift of data. Hence, to replicate out-of-control conditions in a real-world scenario, six biases of varying magnitudes will be intentionally introduced into the datasets using the following formula:

$$x' = x \times n$$

$x'$ represents the changed data point, $x$ is the original data point value, and $n$ refers to the bias percentage. These bias percentages include ±20%, ±10%, and ±5%, totalling six different biases. However, bias will only be introduced to the rest of the instances after every 300 instances within each day. If a given day does not have at least 300 instances, no bias is introduced. Additionally, biases are only applied to the test set, with the same test repeated six times to incorporate each bias variation.

## Data Cleaning

Depending on the data's distribution and characteristics, various data transformation techniques may be necessary during the data cleaning phase.

- **Data Removal:** Columns deemed irrelevant to the model's learning process are eliminated. In this study, all columns except 'Recorded' and 'Result' are discarded. Additionally, instances with non-numeric or null values in the 'Result' column are excluded.
- **Data Type Conversion**: The 'Recorded' date is converted to integers by creating a new column named 'Day' with sequential values (e.g., 1, 2, 3), while 'Result' is converted to floats.

Following these cleaning procedures, the dataset is further segregated based on the DTA. As the study concentrates on sodium and creatinine, two new datasets are created, each containing all instances of either sodium or creatinine. These variables are selected due to their distinct distributions, with sodium exhibiting a more normal distribution and creatinine displaying positive skewness. Below is a sample dataset illustrating these transformations.

| Day | Result |
|---|---|
| 1 | 142.0 |

| | |
|---|---|
| 1 | 139.0 |
| 1 | 131.0 |
| 1 | 132.0 |
| 2 | 139.0 |

*Table 2: A Snippet of the Sodium Data Frame*

# Data Pre-processing

With that, each dataset will be divided into a training set comprising 80% of the data, and a test set containing 20%. Random shuffling of the dataset cannot be employed as it will destroy the time component.

Due to the distribution of creatinine dataset, a box-cox transformation is performed. The Box-Cox transformation is a statistical method used to stabilize the variance and make data more normally distributed. The calculation for The Box-Cox transformation is as follows:

$$y' = \begin{cases} \frac{\left(y^{\lambda} - 1\right)}{\lambda}, & if \quad \neq 0 \\ \log(y), & \lambda = 0 \end{cases}$$

Where $y$ is the original data value, and $y'$ is the transformed data. λ refers to the transformation parameter. This process will be conducted on the creatinine training set. The λ value obtained from the training set will then be utilized to perform the Box-Cox transformation on the test set. This process is omitted from the sodium dataset as it is already normally distributed.

For each train and test dataset of sodium and creatinine, a min-max normalization is applied on 'Result'. MinMax normalization is a technique used to scale numerical features to a fixed range, typically between 0 and 1. It involves subtracting the minimum value from each observation and then dividing by the difference between the maximum and minimum values of the feature. This ensures that all values fall within the specified range, making it easier for machine learning algorithms to process the data. The calculation for normalization can be seen below:

$$X_{normalized} = \frac{X - X_{minimum}}{X_{maximum} - X_{minimum}}$$

Where $X$ represents the original data value, $X_{normalized}$ is the normalized data, while $X_{maximum}$ and $X_{minimum}$ refers to the maximum and minimum value of $X$ in the dataset respectively. This normalization step can be omitted when using simple MA experiment because there is only one analyte per run.

# Data Modelling for Simple Moving Average

For each of the analyte's dataset, specific truncation limit, batch size, and control limits were individually determined for the simple MA method. The selection of truncation limits is contingent upon the specific patient population served by the laboratory. However, as a starting point, following (Lukić & Ignjatović, 2019)'s result, an upper limit of 150 μmol/L for creatinine is

considered, whereas for sodium, there is no need for truncation limit when using simple MA. The control limit technique used in this study is a symmetrical control limit. The standard deviation is calculated from the mean of the training set, with the lower and upper control limits defined by subtracting and adding a certain threshold value from the mean, respectively. (Fleming & Katayev, 2015) also recommended 50 as the block size for PBRTQC. Anomalies are identified when a data point exceeds the upper control limit or falls below the lower control limit.

## Data Modelling for LSTM Autoencoder

This study explores two types of LSTM autoencoder models. One model incorporates moving averages as part of the pre-processing step (LSTM-AE-MA), while the other does not (LSTM-AE).

The pre-processing steps begin with the application of a rolling window of size 50 to both the normalized training and test sets. This rolling average operation is applied independently to each dataset and serves to smooth out trends, reducing noise and short-term fluctuations. The goal here is to enhance the learning process of the subsequent LSTM-AE-MA model by mitigating the impact of individual data points with significant fluctuations. For the LSTM-AE model, this step can be omitted.

Subsequently, sequences are constructed for both the training and test sets. Each sequence represents a series of data points arranged in chronological order or another meaningful sequence. In this context, where data pertains to daily maintenance activities, sequences are organized to contain data points from the same day and avoid spanning into the next day. In cases where the last sequence of a day extends into the subsequent day, the last sequence is disregarded, and a new sequence begins at the start of the next day. Additionally, for days where there is insufficient data to form even one sequence, the entire day is skipped. This study will experiment with different timestep sizes for sequencing, ranging from 50 to 200 in increments of 20.

After sequence creation, an LSTM autoencoder is built and trained using the sequenced training set. As an unsupervised learning model, the sequenced training data serves as both the input features and the target for training the autoencoder.

Both LSTM models for sodium and creatinine will adopt identical architectures for comparison. These models are autoencoders crafted to reconstruct sequences of data, comprising two principal components: the encoder and the decoder. This study will explore the configuration of each encoder and decoder, ranging from two hidden layers to four hidden layers, with neuron counts ranging from 16 to 128, decrementing in steps of division by 2. A 20% dropout is also added after every hidden layer to prevent overfitting.

The model is then compiled using the Adam optimizer and mean squared error loss function. Both models will be trained with 100 epochs, batch size of 32, and a further 20% split on the training set as validation.

Upon training the model, the test set is used to check its reconstruction and anomaly detection capability. If the difference between the original input and the reconstructed output exceeded the predefined threshold, an anomaly is then identified.

## Evaluation Metrics

The purpose of developing the LSTM in this study is to perform monitoring tasks. In other words, the LSTM models or simple MA are not making explicit predictions. Essentially, the study is interested in the model's ability to detect anomalies or deviations from expected patterns. Therefore, the evaluation metrics should be designed to measure the effectiveness of the monitoring model in raising flags when there is a problem.

In the field of accuracy, there are four main components to look at:

1. **True Positives (TP):** These are instances where the model correctly raised a flag when there was a problem in the system.
2. **False Negatives (FN):** These are instances where there was a problem, but the model failed to raise a flag. These represent missed detections.
3. **True Negatives (TN):** These are instances where the model correctly did not raise a flag when there was no problem.
4. **False Positives (FP):** These are instances where the model raised a flag when there was no problem. These represent false alarms.

In this research, the goal is to minimise missed detection. However, the expense of the goal is there could be potentially more false alarms. With that, there are several evaluation metrics to consider:

- **Sensitivity (True Positive Rate, TPR):** TPR measures the proportion of actual problems that the system correctly detects. It is calculated as TPR = TP / (TP + FN).
- **Specificity (True Negative Rate, TNR):** Specificity measures the proportion of actual non-problems that the system correctly identifies as non-problems. It is calculated as TNR = TN / (TN + FP).
- **Accuracy:** Accuracy provides an overall measure of the system's correctness in both raising flags when there's a problem and not raising flags when there's no problem. It's calculated as Accuracy = (TP + TN) / (TP + TN + FP + FN).

Given the research's objective, the optimization of sensitivity and specificity is important, as these metrics underscore the model's ability in accurately detecting issues. However, it is essential to assess multiple factors to determine the model's quality. Maximizing sensitivity and recall might lead to a higher rate of false alarms, potentially incurring additional costs by triggering unnecessary quality control procedures when no errors exist. As a result, it is crucial to strike a well-considered balance between sensitivity and specificity. The study's conclusion should consider the acceptable trade-off between false alarms and missed detections, with a meticulous evaluation of the trade-offs.

# Chapter 5

This chapter presents the outcomes derived from the experiments conducted within the scope of this study.

## The Parameters of the Models

After various experiments, this study has concluded several parameter tunings for the simple MA model, LSTM-AE, and LSTM-AE-MA.

For the simple MA model, the window size used for the rolling mean calculation is 50 for creatinine and 70 for sodium. The threshold value used to define the control limit is 3 for both analytes.

When creating sequences for both the LSTM models, the study has concluded to use 150 timesteps for creatinine and 70 timesteps for sodium. This sequence length remains the same for both the LSTM-AE and the LSTM-AE-MA models. For the LSTM-AE-MA, the window size used to calculate the rolling mean is 50 for both creatinine and sodium.

The LSTM architecture is kept consistent across both models (LSTM-AE and LSTM-AE-MA) and for both analytes to ensure a fair comparison. This is because the study aims to understand how the model performs across different analytes and various preprocessing steps. The specific architecture used is depicted in the accompanying *Figure 6*.

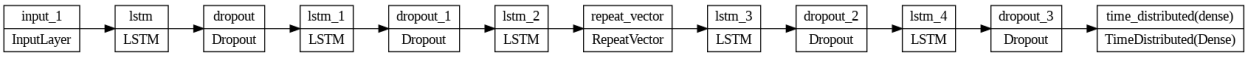


*Figure 6: Model Summary of LSTM*

The model defined is a stacked LSTM autoencoder designed to reconstruct sequences of data. It is composed of an encoder and a decoder, both utilizing LSTM layers to capture and reconstruct temporal dependencies in the input data.

The encoder begins with an input layer that accepts sequences of shape `(TIMESTEP, 1)`. The input sequences are passed through a series of LSTM layers. The first LSTM layer has 64 units and returns sequences, which means it outputs the full sequence for the next layer. This allows the model to capture temporal patterns across all time steps. Following the first LSTM layer, a dropout layer with a rate of 0.2 is applied to prevent overfitting by randomly dropping 20% of the units during training.

The output from the first LSTM and dropout layer is then fed into a second LSTM layer with 32 units, which also returns sequences to preserve the temporal structure of the data. Another dropout layer with the same dropout rate is applied. Finally, the output is passed through a third LSTM layer with 16 units, which returns only the last output in its sequence, effectively summarizing the information captured from the input sequence into a fixed-size context vector.

The decoder reconstructs the input sequence from the context vector provided by the encoder. It begins with a `RepeatVector` layer that repeats the context vector `TIMESTEP` times to match the original sequence length. This repeated sequence is then passed through two LSTM layers,

each mirroring the structure of the encoder but in reverse order. The first LSTM layer in the decoder has 16 units and returns sequences, followed by a dropout layer with a 0.2 dropout rate. The second LSTM layer has 32 units and also returns sequences, followed by another dropout layer.

The final output of the decoder is produced by a `TimeDistributed` dense layer with one unit. This layer applies a fully connected network to each time step independently, producing the reconstructed sequences.

The autoencoder is combined into a single model, with the encoder inputs and decoder outputs defined as the model's inputs and outputs, respectively. The model is compiled using the Adam optimizer, which is an adaptive learning rate optimization algorithm, and mean squared error (MSE) loss function, which is suitable for regression problems involving continuous values.

The threshold value used to calculate the reconstruction error is 0.03 for both LSTM models and analytes.

# Model Results

| Analyte | Bias | LSTM-AE-MA | | | | | LSTM-AE | | | | | Simple MA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Specificity | Sensitivity | FPR | FNR | Accuracy | Specificity | Sensitivity | FPR | FNR | Accuracy | Specificity | Sensitivity | FPR | FNR |
| **Creatinine** | 20% | **95.4753%** | 89.7910% | 99.1539% | 10.2090% | 0.8461% | 93.7609% | 87.0442% | 98.1480% | 12.9558% | 1.8520% | 93.9680% | 89.9123% | 96.5932% | 10.0877% | 3.4068% |
| | 10% | **95.9870%** | 93.9930% | 97.2774% | 6.0070% | 2.7226% | 93.5116% | 87.0819% | 97.7112% | 12.9181% | 2.2888% | 93.9749% | 95.3333% | 93.0956% | 4.6667% | 6.9044% |
| | 5% | **94.1970%** | 99.7749% | 90.5873% | 0.2251% | 9.4127% | 72.3474% | 88.1155% | 62.0483% | 11.8845% | 37.9517% | 39.2941% | 100.0000% | 0.0000% | 0.0000% | 100.0000% |
| | -5% | **94.0606%** | 98.5935% | 91.1272% | 1.4065% | 8.8728% | 78.7284% | 87.8895% | 72.7448% | 12.1105% | 27.2552% | 39.2941% | 100.0000% | 0.0000% | 0.0000% | 100.0000% |
| | -10% | **95.3572%** | 92.7693% | 97.0319% | 7.2307% | 2.9681% | 93.8135% | 87.0458% | 98.2338% | 12.9542% | 1.7662% | 93.4096% | 94.4035% | 92.7663% | 5.5965% | 7.2337% |
| | -20% | **95.2540%** | 89.2924% | 99.1120% | 10.7076% | 0.8880% | 93.8140% | 12.9542% | 98.2338% | 12.9529% | 1.7662% | 93.6509% | 89.3333% | 96.4456% | 10.6667% | 3.5544% |
| | Average | 95.0552% | 94.0357% | 95.7149% | 5.9643% | 4.2851% | 87.6626% | 75.0219% | 87.8533% | 12.6293% | 12.1467% | 75.5986% | 94.8304% | 63.1501% | 5.1696% | 36.8499% |
| **Sodium** | 20% | **95.7895%** | 92.9492% | 97.6555% | 7.0508% | 2.3445% | 96.5153% | 94.9008% | 97.5858% | 5.0992% | 2.4142% | 91.2234% | 88.0702% | 93.2889% | 11.9298% | 6.7111% |
| | 10% | **95.2814%** | 98.4829% | 93.1782% | 1.5171% | 6.8218% | 87.3394% | 95.6612% | 81.8220% | 4.3388% | 18.1780% | 90.5569% | 98.0000% | 85.6815% | 2.0000% | 14.3185% |
| | 5% | 39.6487% | 100.0000% | 0.0000% | 0.0000% | 100.0000% | **48.8299%** | 97.3433% | 16.6645% | 2.6567% | 83.3355% | 39.5778% | 100.0000% | 0.0000% | 0.0000% | 100.0000% |
| | -5% | 39.7559% | 100.0000% | 0.1776% | 0.0000% | 99.8224% | **48.6628%** | 96.9030% | 16.6786% | 3.0970% | 83.3214% | 39.5778% | 100.0000% | 0.0000% | 0.0000% | 100.0000% |
| | -10% | **95.5185%** | 97.5443% | 94.1876% | 2.4557% | 5.8124% | 94.0124% | 94.8736% | 93.4414% | 5.1264% | 6.5586% | 90.9318% | 96.3333% | 87.3937% | 3.6667% | 12.6063% |
| | -20% | 95.7097% | 92.2137% | 98.0064% | 7.7863% | 1.9936% | **96.8156%** | 94.8661% | 98.1081% | 5.1339% | 1.8919% | 91.1193% | 87.1053% | 93.7486% | 12.8947% | 6.2514% |
| | Average | 76.9506% | 96.8650% | 63.8675% | 3.1350% | 36.1325% | 78.6959% | 95.7580% | 67.3834% | 4.2420% | 32.6166% | 73.8312% | 94.9181% | 60.0188% | 5.0819% | 39.9812% |

*Table 3:Overview of Experiment Results (Red-Bold text highlights the best model accuracy for each bias and analyte, blue text highlights the FPR that is below 10%)*

Based on *Table 3*, the study has found several findings. The LSTM-AE-MA demonstrated the best overall performance. This model exhibited high accuracy across different conditions, achieving over 90% accuracy for both analytes while maintaining an overall FPR below 6%. Despite its strong performance, the model did not perform as well for sodium, where it showed a bias of approximately ±5%. When the sodium had a ±5% bias change, this model achieved an FPR of 0%. However, the FNR was 100% when the bias change was +5%, and 99.8224% when the bias change was -5%.

The LSTM-AE achieved moderately good performance in comparison. It also demonstrated moderately high accuracy but had a higher FPR, particularly for creatinine, where the FPR exceeded 10% on average. Interestingly, unlike the LSTM with Moving Average, this model achieved the highest accuracy for sodium with a bias of ±5%. When the sodium had a ±5% bias change, the FPR remained below 5%, and the FNR was around 83%.

The simple MA also showed moderately good performance. It achieved moderately high accuracy, and its average FPR remained below 10% for both sodium and creatinine. This consistency in FPR indicates that while the simple MA model may not outperform the both the LSTM models in all aspects, it provides a reliable baseline with reasonably good accuracy and controlled FPR across different analytes. Nonetheless, similar to the LSTM-AE-MA, it had rather low accuracy for sodium at ±5% bias change, with an FNR of 100% and an FPR of 0%.

# Chapter 6

This chapter delves into the discoveries made during the course of this study, as well as outlining potential avenues for future research.

## Discussions

Based on the results, several concerns and interesting findings emerge.

### Balancing False Positive Rate and False Negative Rate

A high FNR can potentially result in incorrect patient results being released without correction, or a need to reanalyze the results, causing either inaccurate or delayed diagnoses. Conversely, a high False Positive Rate (FPR) will necessitate frequent QC checks, which can be costly for the clinical laboratory. Ideally, a robust model should achieve high accuracy with both low FPR and FNR. However, this study prioritizes maintaining a low FPR (below 10%) while ensuring the FNR does not exceed 20%. This is because this study has concluded that it is medically more costly to run unnecessary QC checks.

The results indicate that both the LSTM-AE-MA and simple MA models successfully maintained an FPR below 10% across both analytes and most biases. In contrast, the LSTM-AE only achieved an FPR below 10% for sodium. When examining the FNR, the LSTM-AE-MA generally outperformed the other two models, maintaining an FNR below 10% except for sodium with a ±5% bias change.

### The Impacts of Moving Average on LSTM

As mentioned earlier, the moving average is effective at smoothing noise and providing a better generalization of the data. While noise is known to negatively impact machine learning performance, applying too much MA (by increasing the window size of the rolling mean, resulting in fewer fluctuations) can lead to a loss of valuable information. This is because certain significant data points may be smoothed out, even if they are not outliers. Consequently, the LSTM may need to learn those normal fluctuations effectively. Therefore, it is crucial to be cautious when tuning the window size for the moving average in LSTM models.

### The Effects of ±5% Bias

As shown in the results, both analytes experienced a significant drop in accuracy at a ±5% bias change, except for the LSTM-AE-MA for creatinine. While the FPR generally remained below 5%, the FNR ranged between 80% and 100%. This effect was particularly pronounced for sodium, regardless of the model used to identify anomalies. During experiments, the accuracy for sodium actually dropped at a bias change of 6% onwards. This study suspects this may be due to the shorter range of ‘Result’ values for sodium compared to creatinine. The value for sodium typically ranges between 120 mmol/L to 150 mmol/L, while creatinine ranges from 0 mmol/L to 500 mmol/L, according to the datasets used in this study. This suggests that a 5% change in sodium levels may be too subtle to detect, as the anomaly could still fall within the threshold of the “normal” pattern range. For the LSTM to detect such changes finely, the size of the time step when generating sequences needs to be smaller. However, this mitigation could increase the FPR instead.

### Threshold Value Determination

This study primarily determined the threshold value through trial and error. The final threshold value was identified through various experiments and detailed in *Chapter 5*. Any threshold value below this defined point showed no significant improvement or impact on performance. This is because a smaller threshold value would only be effective in detecting an even smaller bias, thereby making the models more sensitive. However, since this study only experimented with a ±5% bias change, which typically still falls within the normal range, a smaller threshold would only potentially increase the FPR.

### The Reliability of LSTM in PBRTQC

Based on the results, the LSTM-AE-MA generally performed the best in all situations simulated in this study. However, it may struggle to identify anomalies when there are subtle changes in the data, as the moving average can smooth out these minor fluctuations. While increasing the complexity of the LSTM by adding more neurons or hidden layers could potentially enhance the model's performance, it also makes the training process more computationally intensive. Training an LSTM model without GPU support during these experiments could take up to 10 hours. Even with GPU acceleration, which reduces the average training time to about 30 minutes, the GPU memory can still be quickly exhausted, leading to internal errors during training. Additionally, LSTM models require a substantial amount of data to train effectively. It is also important to note that PBRTQC performance depends on many factors specific to the individual laboratory, such as the order of samples and the patient population. Optimal parameters for one laboratory might not work for another laboratory, even when dealing with the same analyte (Bietenbeck et al., 2020). Hence, the tunings done for the LSTM in this study may not work well for other settings. Despite these challenges, the LSTM-AE-MA can be considered successful and shows potential for tackling more complex anomaly scenarios and different types of analytes.

## Future Works

This study holds several assumptions when performing these experiments. One key assumption is that the laboratory machine setup is maintained daily. This assumption significantly affects the technique used for generating sequences for the LSTM and the calculation of the rolling mean.

Additionally, many attributes of the datasets are omitted, which can impact the training of the models. For instance, the experiment ignores variables such as the different machines used to generate the patient results, the age of the patients, their potential illnesses based on the wards they are in, and the different dates of the results. All of these factors can influence the training and overall model performance. Furthermore, the analytes are trained separately in this study, which means the performance of the LSTM is uncertain if the analytes are concatenated and trained together.

As this study is a proof of concept, several scenarios have yet to be simulated. These include more anomaly scenarios such as a gradual shift or those stated in the Westgard rules (University of Texas Medical Branch, 1992). Additionally, different variants of the AON as a baseline comparison, such as the exponentially weighted moving average (EWMA), moving median, or autoregressive

integrated moving average (ARIMA), may work better than the simple moving average. Exploring these alternatives could affect whether the LSTM can be seen as the better solution when dealing with similar scenarios. Moreover, more analytes need to be experimented on. Due to the time limitations of this study, only two analytes have been focused on. Other commonly found analytes in PBRTQC, such as potassium, glucose, phosphorus, and more, should be considered for future research.

# Conclusion

The study conducted a comparison of unsupervised LSTM autoencoder models, both with and without MA as preprocessing, alongside a simple MA model, to detect anomalies in laboratory data. LSTM-AE-MA demonstrated superior performance overall, boasting high accuracy and low FPR. However, it faced challenges in identifying subtle changes, particularly in sodium levels. Conversely, LSTM-AE showed moderately good performance, while the simple MA model served as a reliable baseline. Notably, the study highlighted the importance of understanding the smoothing effects of MA, which can inadvertently lead to information loss during LSTM training. Despite these challenges, LSTM-AE-MA showed promise in handling sudden shift anomaly scenario. Moreover, LSTM's ability to function effectively without the need for labeled data suggests its potential to tackle complex scenarios in the future, indicating promising prospects for further research and application.

# References

Badrick, T., Bietenbeck, A., Katayev, A., van Rossum, H. H., Loh, T. P., & Cervinski, M. A. (2020). Implementation of patient-based real-time quality control. *Critical Reviews in Clinical Laboratory Sciences*, 1–16. https://doi.org/10.1080/10408363.2020.1765731

Bietenbeck, A., Cervinski, M. A., Katayev, A., Loh, T. P., van Rossum, H. H., & Badrick, T. (2020). Understanding Patient-Based Real-Time Quality Control Using Simulation Modeling. *Clinical Chemistry*, *66*(8), 1072–1083. https://doi.org/10.1093/clinchem/hvaa094

Bowie, M., Begoli, E., Park, B., & Bopaiah, J. (2017). Towards an LSTM-based Approach for Detection of Temporally Anomalous Data in Medical Datasets. In *University of Arkansas at Little Rock*. https://ualr.edu/informationquality/files/2023/02/P29-ICIQ2017-LSTM-Based-Approach.pdf

Cembrowski, G. S., Chandler, E. P., & Westgard, J. O. (1984). Assessment of "Average of Normals" Quality Control Procedures and Guidelines for Implementation. *American Journal of Clinical Pathology*, *81*(4), 492–499. https://doi.org/10.1093/ajcp/81.4.492

Fleming, J. K., & Katayev, A. (2015). Changing the paradigm of laboratory quality control through implementation of real-time test results monitoring: For patients by patients. *Clinical Biochemistry*, *48*(7-8), 508–513. https://doi.org/10.1016/j.clinbiochem.2014.12.016

Hu, M., Li, W., Yan, K., Ji, Z., & Hu, H. (2019). Modern Machine Learning Techniques for Univariate Tunnel Settlement Forecasting: A Comparative Study. *Mathematical Problems in Engineering*, *2019*(7057612), 1–12. https://doi.org/10.1155/2019/7057612

Kiser, A. C., Eilbeck, K., & Bucher, B. T. (2023). Developing an LSTM Model to Identify Surgical Site Infections using Electronic Healthcare Records. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, *2023*, 330–339. (Kiser et al., 2023)

Liu, Y., Pang, Z., Karlsson, M., & Gong, S. (2020). Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control. *Building and Environment*, *183*(107212). https://doi.org/10.1016/j.buildenv.2020.107212

Lukić, V., & Ignjatović, S. (2019). Optimizing moving average control procedures for small-volume laboratories. *Biochemia Medica*, *29*(3), 587–599. https://doi.org/10.11613/bm.2019.030710

Maleki, S., Maleki, S., & Jennings, N. R. (2021). Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering. *Applied Soft Computing*, *108*(107443), 107443. https://doi.org/10.1016/j.asoc.2021.107443

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). *Long Short Term Memory Networks for Anomaly Detection in Time Series Long Short Term Memory Networks for Anomaly Detection in Time Series*. ESANN.

Spies, N. C., Farnsworth, C. W., & Jackups, R., Jr. (2023). Data-Driven Anomaly Detection in Laboratory Medicine: Past, Present, and Future. The Journal of Applied Laboratory Medicine, 8(1), 162-179.

Staudemeyer, R., & Morris, E. (2019). -*Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks*. https://arxiv.org/pdf/1909.09586

University of Texas Medical Branch. (1992). *Westgard Rules*. Westgard Rules; University of Texas Medical Branch. https://www.utmb.edu/policies_and_procedures/Non-IHOP/Respiratory/Pulmonary_Function_Laboratory/04-10%20ABG%20-%20Westgard%20Rules.pdf

# APPENDIX A: Important Source Code Snippet

The following attachments is the important source code snippets for the experiment documented in this document.

```
def calculate_control_limits(data, num_std_dev = 1.5):
    mean = np.mean(data)
    std_dev = np.std(data)

    # Calculate the upper and lower control limits
    ucl = mean + (num_std_dev * std_dev)
    lcl = mean - (num_std_dev * std_dev)

    return ucl, lcl
```

```
def test_simulation (bias_value, cl_threshold, window_size, train, test):
  ucl, lcl = calculate_control_limits(train['Result'], cl_threshold)


  test['Result'] = test.groupby('Day')['Result'].transform(lambda x: simulation(x, bias_value))

  final = pd.concat([train, test], axis=0)

  final['MA'] = final['Result'].rolling(window = window_size).mean()
  final['STD'] = final['Result'].rolling(window = window_size).std()
  final['UCL'] = ucl
  final['LCL'] = lcl

  final['Anomaly'] = (final['MA'] > final['UCL']) | (final['MA'] < final['LCL'])
  anomalies = final[final['Anomaly'] == True]

  return ucl, lcl, final, anomalies
```

*Figure 7: Main Source Code for Simple Moving Average*

```
TIMESTEP = 70

def create_sequences_with_null(data, sequence_length):
    sequences = []
    # Find the index where 'Result' is not null
    start_index = np.argmax(~np.isnan(data['Result']))

    # Initialize variables to track the current day and the start index of the current day
    current_day = data['Day'].iloc[start_index]
    current_day_start = start_index

    for i in range(start_index, len(data) - sequence_length + 1):
        # Check if any NaN values are present in the sequence
        if data['Result'].iloc[i:i+sequence_length].isnull().any():
            # Move to the next day
            current_day += 1
            current_day_start = data[data['Day'] == current_day].index[0]
            continue  # Skip sequences with NaN values

        # Check if the sequence spans across multiple days
        if data['Day'].iloc[i] != current_day:
            # Move to the next day
            current_day += 1
            current_day_start = data[data['Day'] == current_day].index[0]
            continue  # Skip sequences spanning multiple days

        sequence = data['Result'].iloc[i:i+sequence_length]  # Select only the 'Result' column
        sequences.append(sequence)

    return np.array(sequences)
```

*Figure 8: Source Code of Sequence Generator for LSTM*

```
# Define the input shape
input_shape = (TIMESTEP, 1)

# Define the encoder layers
encoder_inputs = Input(shape=input_shape)
encoder = LSTM(units=64, return_sequences=True)(encoder_inputs)
encoder = Dropout(0.2)(encoder)
encoder = LSTM(units=32, return_sequences=True)(encoder)
encoder = Dropout(0.2)(encoder)
encoder = LSTM(units=16, return_sequences=False)(encoder)

# Define the decoder layers
decoder = RepeatVector(TIMESTEP)(encoder)
decoder = LSTM(units=16, return_sequences=True)(decoder)
decoder = Dropout(0.2)(decoder)
decoder = LSTM(units=32, return_sequences=True)(decoder)
decoder = Dropout(0.2)(decoder)
decoder_outputs = TimeDistributed(Dense(units=1))(decoder)

# Combine encoder and decoder into an autoencoder model
model = Model(inputs=encoder_inputs, outputs=decoder_outputs)

# Compile the autoencoder model
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mse')
```

*Figure 9: Stacked LSTM Autoencoder Model*

# APPENDIX B: Simple Moving Average Result Snippet

The below images show the sample result visualisation form simple moving average

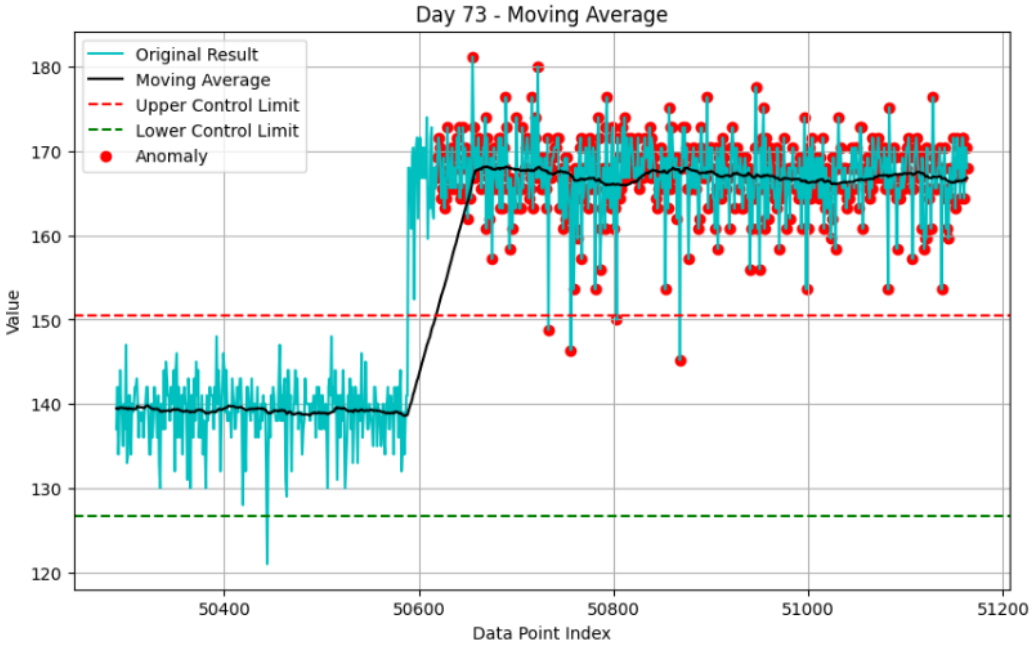


*Figure 10: Sodium (+20% Bias) at Day 73*


*Figure 11: Sodium (+20% Bias) in Train and Test Set*

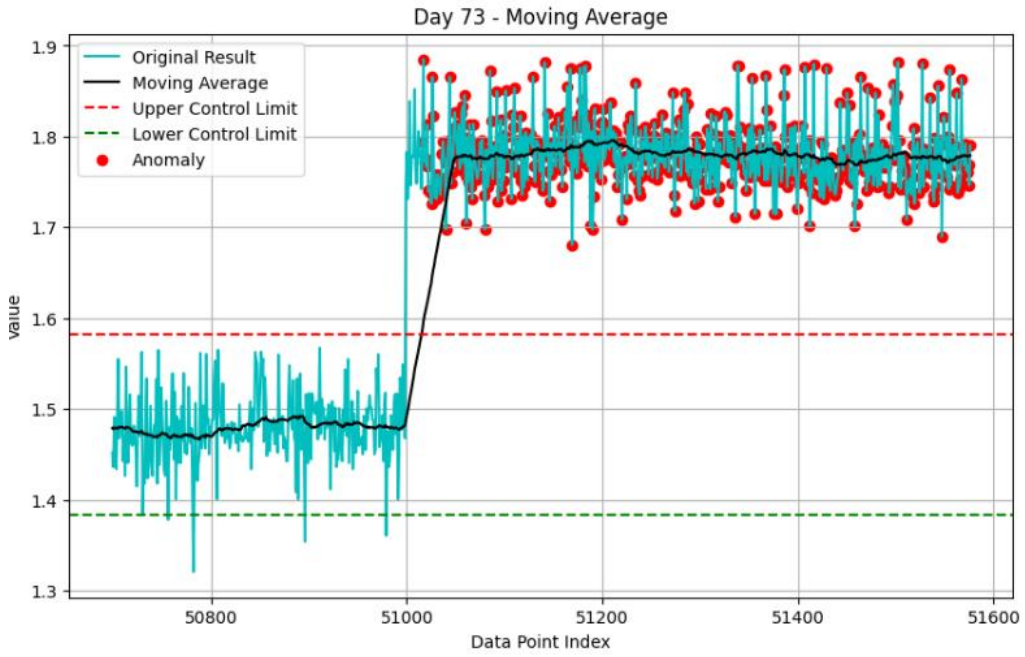


*Figure 12: Creatinine (+20% Bias) at Day 73*

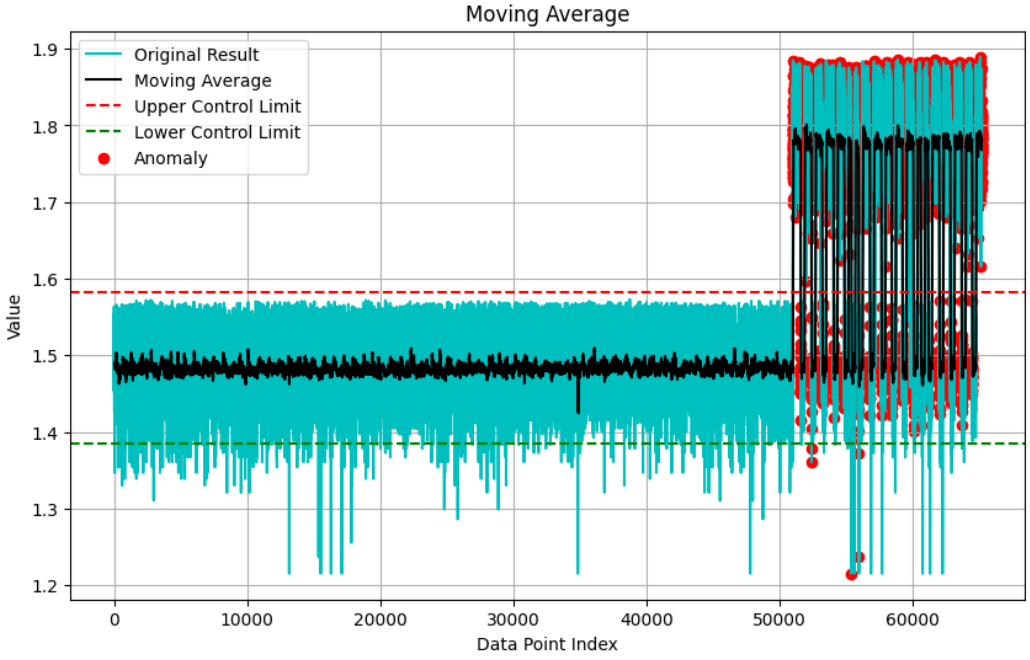


*Figure 13: Creatinine (+20% Bias) in Train and Test Set*

# APPENDIX C: LSTM-AE Result Snippet

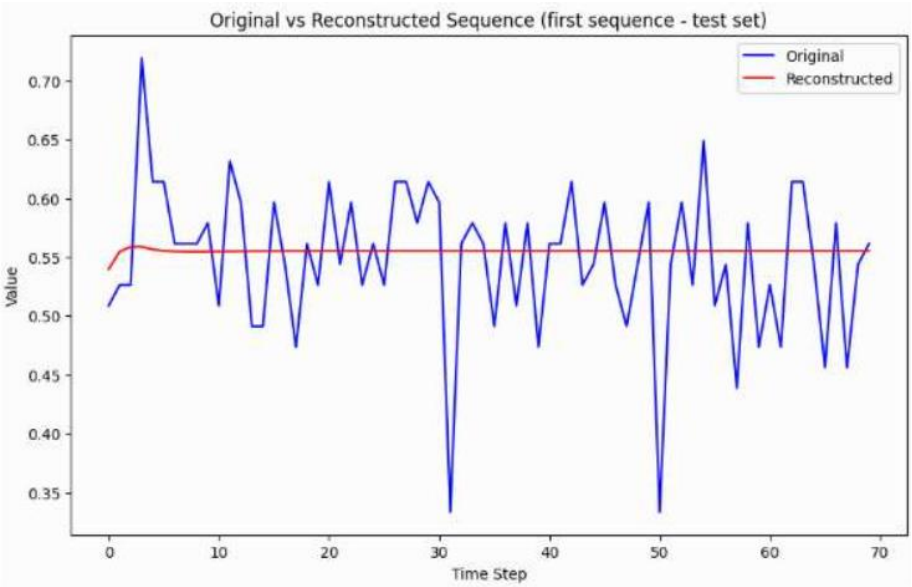


*Figure 14: Sodium Test Set (+20% Bias) Original vs Reconstructed First Sequence Only [No Moving Average]*

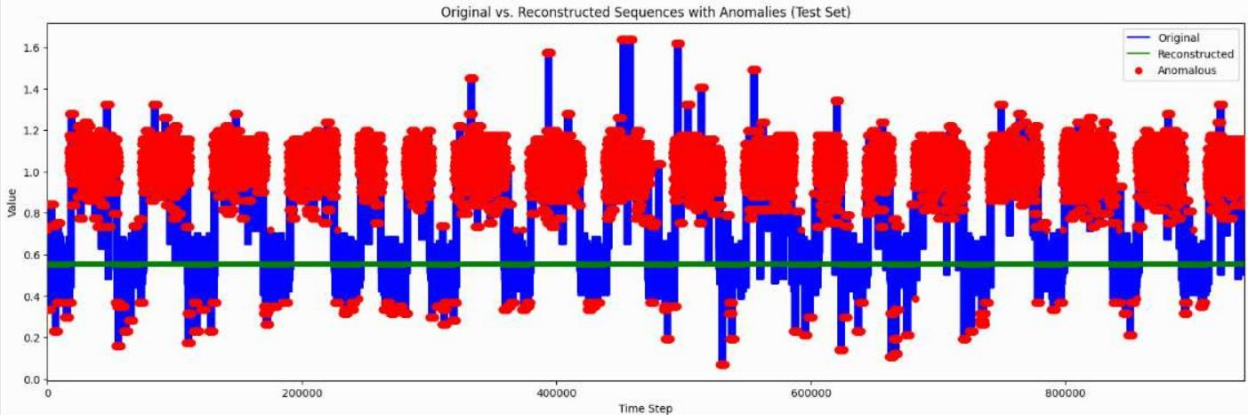


*Figure 15: Sodium Test Set (+20% Bias) with Anomalies Result [No Moving Average]*

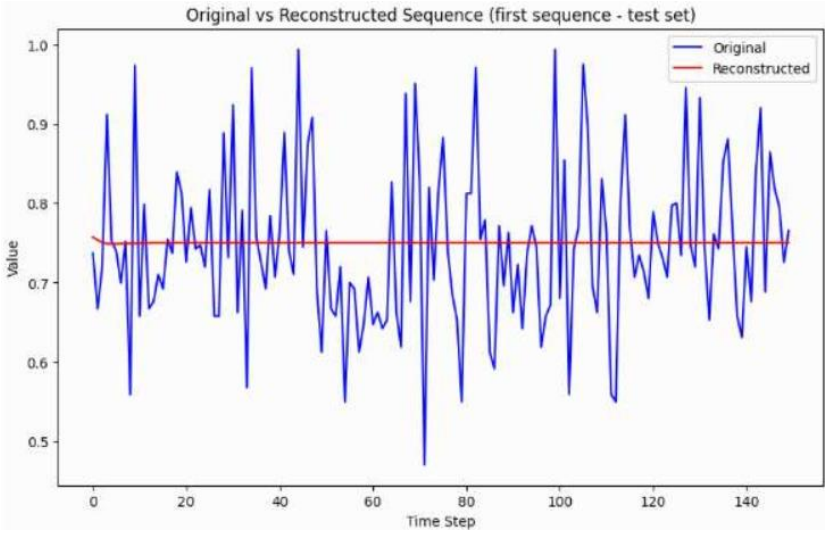


*Figure 16: Figure 14: Creatinine Test Set (+20% Bias) Original vs Reconstructed First Sequence Only [No Moving Average]*

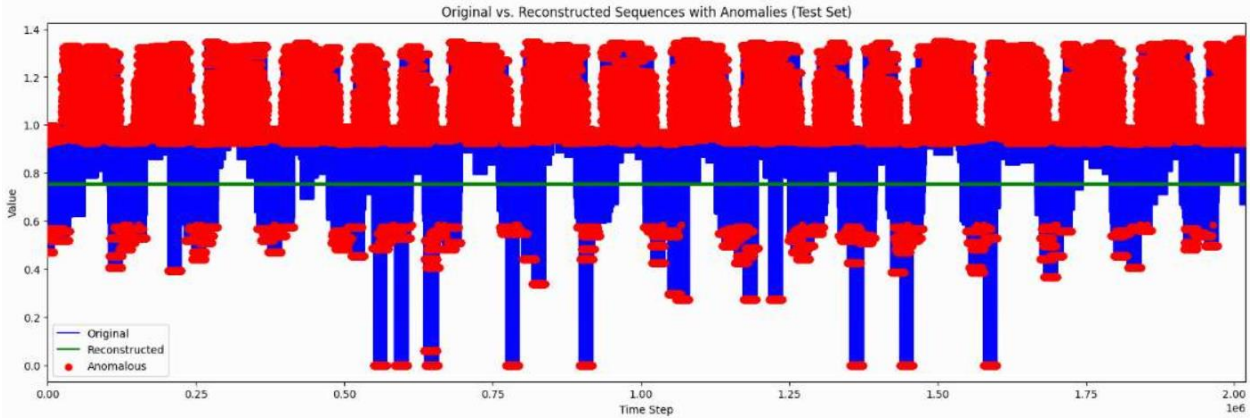


*Figure 17: Creatinine Test Set (+20% Bias) with Anomalies Result [No Moving Average]*

# APPENDIX D: LSTM-AE-MA Result Snippet

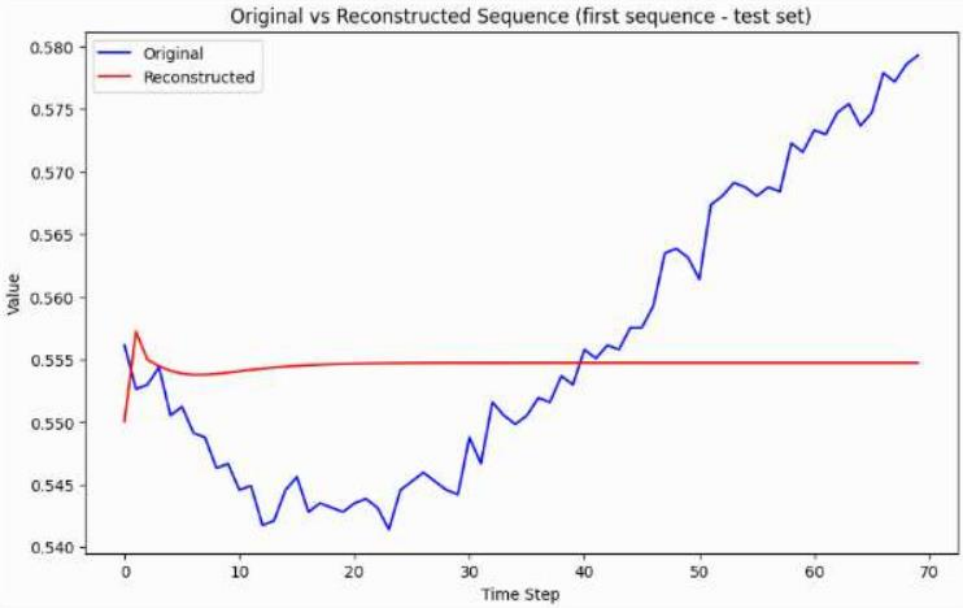


*Figure 18: Sodium Test Set (+20% Bias) Original vs Reconstructed First Sequence Only*

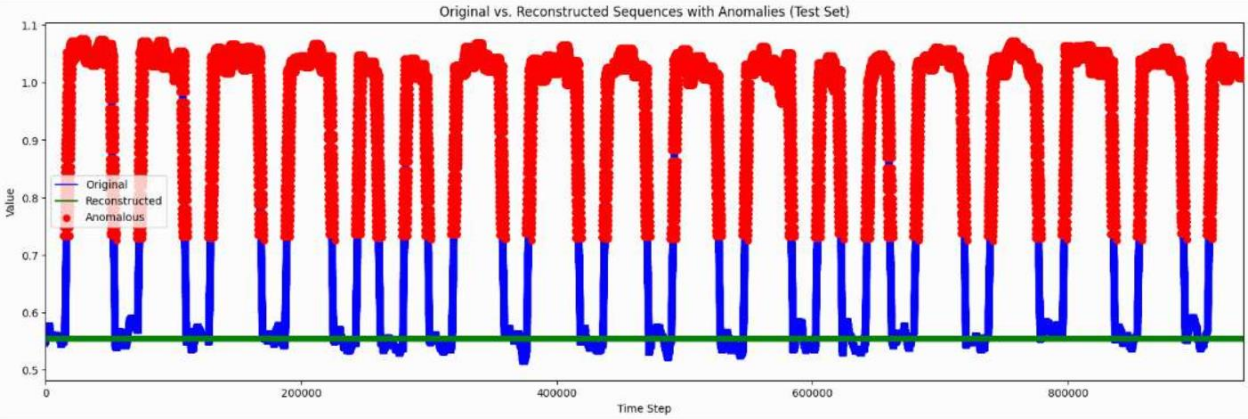


*Figure 19: Sodium Test Set (+20% Bias) with Anomalies Result*

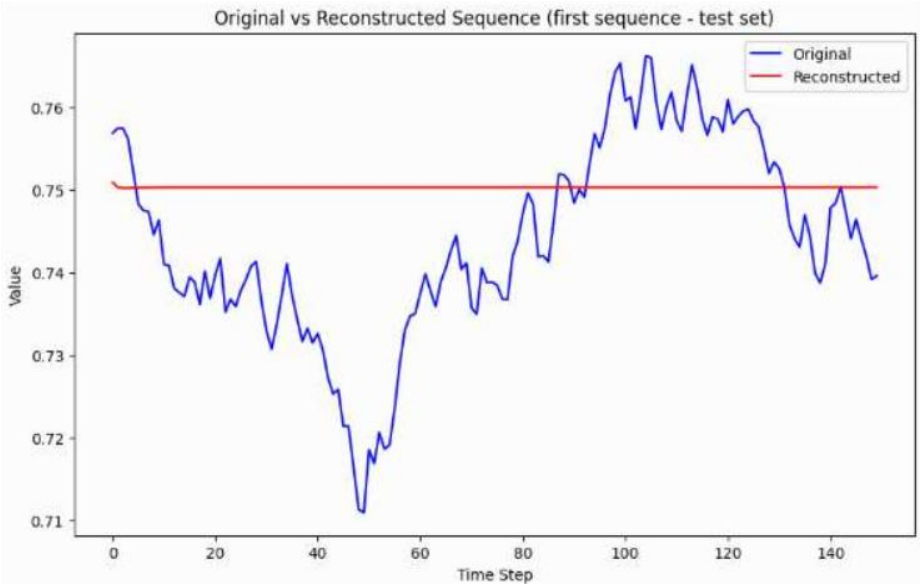


*Figure 20: Creatinine Test Set (+20% Bias) Original vs Reconstructed First Sequence Only*

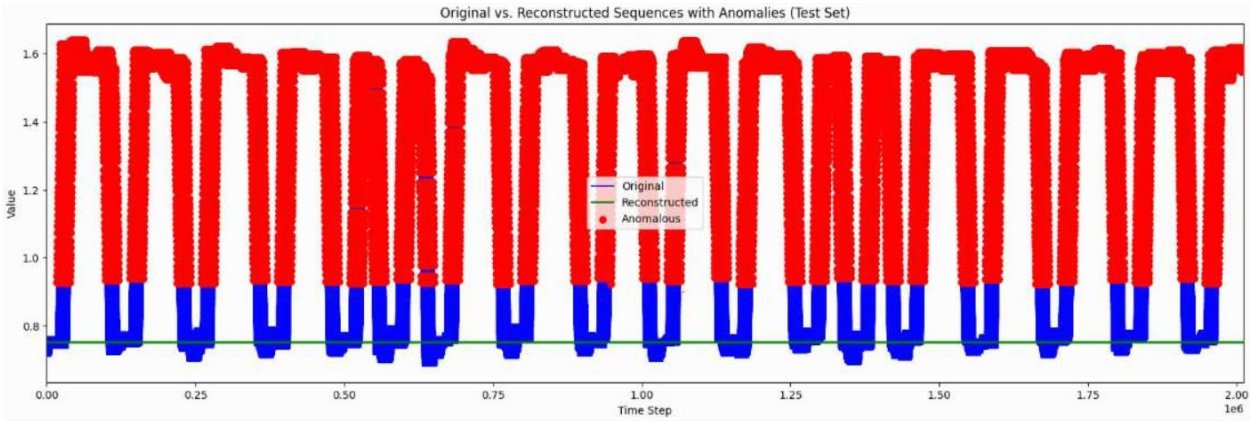


*Figure 21: Creatinine Test Set (+20% Bias) with Anomalies Result*